

OCP Java SE 7

Programmer II

CERTIFICATION GUIDE

Prepare for the 1Z0-804 exam



Mala Gupta



***OCP Java SE 7 Programmer II
Certification Guide***

by Mala Gupta

Bonus chapter 13

Copyright 2015 Manning Publications

brief contents

Introduction 1

- 1 ■ Java class design 13
- 2 ■ Advanced class design 95
- 3 ■ Object-oriented design principles 172
- 4 ■ Generics and collections 242
- 5 ■ String processing 348
- 6 ■ Exceptions and assertions 396
- 7 ■ Java I/O fundamentals 463
- 8 ■ Java file I/O (NIO.2) 512
- 9 ■ Building database applications with JDBC 577
- 10 ■ Threads 627
- 11 ■ Concurrency 679
- 12 ■ Localization 719

13

Full mock exam

This chapter contains

- A mock exam with 90 questions
- Answers to all mock exam questions with extensive explanations and the subobjective on which each exam question is based

13.1 Mock exam questions

ME-Q1. Given

```
enum Color implements AutoCloseable {           //line1
    PINK, GREEN, RED;
    public void close() {
        System.out.print("closed:");
    }
}
class Palette {
    public static void main(String args[]) {
        try (Color red = Color.RED) {           //line2
            System.out.print("try:");
        }
        finally {
            System.out.print("finally:");
        }
    }
}
```

```
    }
}
```

which statement is true? (Choose the best option.)

- ☐ a Compilation fails at //line1—enum can't implement AutoCloseable.
- ☐ b Compilation fails at //line2.
- ☐ c Code compiles and outputs try:closed:finally:.
- ☐ d None of the above

ME-Q2. Assuming that the default locale is set to `Locale.FRANCE`, which option will format and output the number literal value 9 999? (Choose the best option.)

- ☐ a `System.out.println(NumberFormat.getInstance(Locale.FRANCE).format(9999));`
- ☐ b `System.out.println(NumberFormat.getLocalInstance().format(9999));`
- ☐ c `System.out.println(NumberFormat.getDefaultInstance().format(9999));`
- ☐ d `System.out.println(NumberFormat.getDefault().format(9999));`

ME-Q3. Given

```
//insert code here
catch (FileNotFoundException e) {}
catch (IOException e) {}
```

which options, when inserted at //insert code here, will enable the preceding code to compile? (Choose two options.)

- ☐ a `try (File file1 = new File("abc.txt");
File file2 = new File("abc.txt");) {}`
- ☐ b `try (FileInputStream file1 = new File("abc.txt");
FileInputStream file2 = new File("abc.txt");) {}`
- ☐ c `try (FileInputStream file1 = new FileInputStream(new File("abc.txt"));
FileInputStream file2 = new FileInputStream(new
File("abc.txt"));) {}`
- ☐ d `try (File file1 = new File("abc.txt");
File file2 = new File("abc.txt")) {}`
- ☐ e `try (FileInputStream file1 = new File("abc.txt");
FileInputStream file2 = new File("abc.txt")) {}`
- ☐ f `try (FileInputStream file1 = new FileInputStream(new File("abc.txt"));
FileInputStream file2 = new FileInputStream(new
File("abc.txt")) {}`

ME-Q4. Given

```
public int checkFile(Path path) {
    if (Files.exists(path))
        return 1;
```

```

else {
    if (Files.notExists(path))
        return 2;
    else
        return 3;
}
}

```

for any given Path instance, which value can the preceding method return? (Choose the best option.)

- ☐ a Only 1
- ☐ b Only 1 and 2
- ☐ c Only 1 and 3
- ☐ d Only 2
- ☐ e Only 2 and 3
- ☐ f Only 3
- ☐ g 1, 2, or 3

ME-Q5. Which option from the following code would ensure that `getID()` never returns a duplicate value in a multithreaded environment? (Choose the best option.)

```

interface Counter {
    int getID();
}

```

- ☐ a class MyCounter implements Counter {
 AtomicInteger ids = new AtomicInteger();
 public int getID() {
 return ids.incrementAndGet();
 }
 }
- ☐ b class MyCounter extends Counter {
 AtomicInteger ids = new AtomicInteger();
 public int getID() {
 return ids.incrementAndGet();
 }
 }
- ☐ c class MyCounter implements Counter {
 synchronized AtomicInteger ids = new AtomicInteger();
 public int getID() {
 return ids.incrementAndGet();
 }
 }
- ☐ d class MyCounter implements Counter {
 int atomic;
 public int getID() {
 synchronized(new StringBuilder()) {
 return ids.incrementAndGet();
 }
 }
 }

ME-Q6. Given

```
abstract class Dog {  
    void bark() {}  
}  
class Beagle extends Dog {  
    //insert code here  
}
```

which methods correctly override bark() in Dog? (Choose three options.)

- ☐ a static void bark() throws Exception {}
- ☐ b void bark() {}
- ☐ c static void bark() {}
- ☐ d static void bark() throws Error {}
- ☐ e abstract static void bark() {}
- ☐ f final void bark() {}
- ☐ g protected synchronized void bark() {}

ME-Q7. Which object of the following classes can't be passed to method execute() of ForkJoinPool? (Choose four options.)

- ☐ a RecursiveTask<Integer>
- ☐ b RecursiveTask<String>
- ☐ c RecursiveAction<StringBuilder>
- ☐ d RecursiveAction<Boolean>
- ☐ e ForkTask<Long>
- ☐ f ForkTask<Double>
- ☐ g ForkJoinPoolTask<Short>
- ☐ h ForkJoinPoolTask<Byte>

ME-Q8. Given

```
class ExA extends Exception {}  
class ExB extends ExA {}  
class ExC extends ExB {}  
class App {  
    void launch() throws ExB{  
        throw new ExC();  
    }  
    void useLaunch() {  
        try {  
            launch();  
        }  
        catch (/*insert code here*/) {}  
    }  
}
```

which option when inserted at `/*insert code here*/` won't enable class App to compile? (Choose the best answer option.)

- ☐ a Exception e
- ☐ b ExA e
- ☐ c ExB e
- ☐ d ExC e
- ☐ e None of the above

ME-Q9. Given

```
enum Color {  
    RED("yellow"), YELLOW("blue"), BLUE("red");  
    String color;  
    Color(String c) {this.color = color;}  
}  
Integer qty = 991177;
```

what is the output of the following code? (Choose the best answer option.)

```
System.out.printf("I want: [%-+5d] bottles of %s color", qty, Color.RED);
```

- ☐ a I want: [991177] bottles of RED color
- ☐ b I want: [991177] bottles of yellow color
- ☐ c I want: [991177] bottles of RED color
- ☐ d I want: [991177] bottles of yellow color
- ☐ e I want: [+991177] bottles of RED color
- ☐ f I want: [+991177] bottles of yellow color
- ☐ g I want: [+99117] bottles of RED color
- ☐ h I want: [+99117] bottles of yellow color

ME-Q10. Given that the following code throws a `NullPointerException`, what could be the probable cause? (Choose two options.)

```
Console console = System.console();           //line1  
String name = console.readLine("Enter name:"); //line2  
String in = console.readLine();               //line3  
System.out.println(in.length());              //line4
```

- ☐ a The code is invoked from the command line before redirecting the standard input and output streams.
- ☐ b Code execution started automatically as a result of execution of some other program.
- ☐ c The underlying system doesn't support the Console operations.
- ☐ d When prompted (as a result of execution of `//line3`), the user didn't enter a value.
- ☐ e When prompted (as a result of execution of `//line3`), the user entered null.

ME-Q11. What is the difference when a class implements the `Runnable` interface versus extends class `Thread` to create a separate thread of execution? (Choose the best option.)

- ☐ a When a class implements `Runnable`, it can only extend `Thread`.
- ☐ b Classes that implement `Runnable` don't inherit the members of class `Thread`.
- ☐ c You can't modify the priority of threads of execution created using classes that implement `Runnable`.
- ☐ d By extending class `Thread`, a new thread of execution is created more efficiently than by implementing the `Runnable` interface.

ME-Q12. What is the iteration order of values inserted in a `HashMap`? (Choose the best option.)

- ☐ a Insertion order
- ☐ b Natural order of keys
- ☐ c Natural order of values
- ☐ d Undefined

ME-Q13. Assuming that the exceptions are handled appropriately, which option is the recommended way to load a database driver and establish a connection with the database using JDBC version 4.0? (Choose the best option.)

- ☐ a `java.sql.DriverManager.registerDriver("com.mysql.jdbc.driver");`
`Class.forName("com.mysql.jdbc.driver");`
`DriverManager.getConnection("jdbc:mysql://data.ejavaguru.com:3305/myDB");`
- ☐ b `Class.forName("com.mysql.jdbc.driver");`
`DriverManager.getConnection("jdbc:mysql://data.ejavaguru.com:3305/myDB");`
- ☐ c `DriverManager.getConnection("jdbc:mysql://data.ejavaguru.com:3305/myDB");`
- ☐ d `DriverManager.getConnection("jdbc:mysql:", "guest", "guest");`

ME-Q14. Given

- X defines methods.
- Y extends X.
- Y can overload methods defined in X.
- X can't define static methods.

Which of the following statements is true? (Choose two options.)

- ☐ a X is a class.
- ☐ b Y is a class.
- ☐ c X is an interface.
- ☐ d Y is an interface.

ME-Q15. What is the output of the following code? (Choose the best option.)

```
String[] fileNames = {"data-eng-temp.txt",
                      "dataA_sst.temp.xml",
                      "data-scc-perm.doc",
                      "Data-pra-sscu.txt",
                      "data-php-data.php" };
PathMatcher matcher = FileSystems.getDefault()
    .getPathMatcher("glob:*a?a-*x*");

int result = 0;
for (int i = 0; i < fileNames.length; i++)
    if (matcher.matches(Paths.get(fileNames[i])))
        result = result + 2;
System.out.println(result);
```

- ☐ a 2
- ☐ b 4
- ☐ c 6
- ☐ d 8
- ☐ e 10

ME-Q16. Given

```
class Color {
    final int shade;
    //insert code here
}
```

which options when inserted independently at //insert code here will enable class Color to compile successfully? (Choose two options.)

- ☐ a Color(int shade) {
 this.shade = shade;
}
- ☐ b static {
 shade = 10;
}
- ☐ c {
 shade = 10;
}
- ☐ d Color () {
 shade = 10;
}
 Color(int shade) {
 this();
 this.shade = 20;
 }

ME-Q17. Given

```
Item PEN
ID, INTEGER: PK
MODEL, VARCHAR(50)
LENGTH, INTEGER
WEIGHT, REAL
```

assuming that table PEN contains two rows, what is the output of the following code? (Choose the best option.)

```
try (
    Connection c = DriverManager.getConnection
        ("jdbc:mysql://localhost/abc", "a", "b");
    String query = "SELECT * FROM pen WHERE length>99";
    Statement stmt = c.createStatement();
    ResultSet rs = stmt.executeQuery(query) {
        while (rs.next ()) {
            System.out.println(rs.getString("MODEL"));
        }
    }
} catch (Exception se) {
    System.out.println("Exception");
}
```

- ☐ a The code outputs a value for column model twice, if its corresponding length is greater than 99.
- ☐ b The code outputs a value for column model once, irrespective of the value of column length.
- ☐ c The code outputs an Exception.
- ☐ d The code fails to compile.

ME-Q18. Which statements are correct? (Choose two options.)

- ☐ a Class Console defines methods to access the character-based console device associated with the current JVM.
- ☐ b System.console() always returns a non-null value.
- ☐ c System.getConsole() might not always return a null value.
- ☐ d Console can be used to write formatted data.
- ☐ e Console's method readPwd() reads a password or pass-phrase from the console with echoing disabled.
- ☐ f Console's method readUser() reads a String value from the console without echoing disabled.

ME-Q19. Which statement is incorrect? (Choose the best answer.)

- ☐ a A CachedRowSet is scrollable, updatable, and serializable.
- ☐ b A JdbcRowSet is always connected to its database.

- ☐ c A FilteredRowSet object applies a criterion on all rows in a RowSet object to manage a subset of rows in a RowSet object.
- ☐ d Any number of RowSet objects can be added to an instance of JoinRowSet if they can be related in a SQL JOIN.
- ☐ e An application can modify the data in a CachedRowSet object, and those modifications can then be propagated back to the source of the data.
- ☐ f A CachedRowSet object is a connected rowset.

ME-Q20. Given that contents of in.txt are

day

what is the output of the following code? (Choose the best option.)

```
import java.io.*;
class ReadWrite {
    public static void main(String args[]) throws Exception {
        BufferedReader in = new BufferedReader(new FileReader("in.txt"));
        BufferedWriter out = new BufferedWriter(new FileWriter("out.txt"));
        int i = in.read();
        int ctr = 0;
        while (i != -1) {
            out.write(++ctr + "." + (char)i);
            i = in.read();
        }
        out.flush();
        out.close();
    }
}
```

- ☐ a File out.txt contains 1.d.2.a.3.y.
- ☐ b File out.txt contains
 - 1.d
 - 2.a
 - 3.y
- ☐ c File out.txt contains 1.d2.a3.y.
- ☐ d Code fails to compile
- ☐ e None of the above

ME-Q21. Given

- Fwimmer is an interface.
- Swammer extends Fwimmer.
- JamG defines a variable of type Boolean.
- Jammer extends JamG.
- Yammer implements Swammer.
- Jammer implements Fwimmer.

Which statements are correct? (Choose two options.)

- ☐ a Fwimmer IS-A Swammer.
- ☐ b Yammer HAS-A Fwimmer.
- ☐ c Jammer HAS-A Boolean.
- ☐ d JamG IS-A Swammer.
- ☐ e Jammer IS-A Swammer.
- ☐ f Jammer IS-A Fwimmer.

ME-Q22. Given

```
class Lake {
    void waterSkiing() throws RuntimeException {
        System.out.print("Skiing:");
    }
}
class Travel {
    public static void main(String args[]) {
        Lake lake = new Lake();
        try {
            lake.waterSkiing();
        }
        catch (RuntimeException e) {
            System.out.print("RuntimeEx:");
        }
        catch (Exception e) {
            System.out.print("Ex:");
        }
    }
}
```

what is the output? (Choose the best option.)

- ☐ a Skiing:
- ☐ b Skiing:RuntimeEx:
- ☐ c Skiing:Ex:
- ☐ d Compilation error

ME-Q23. Given

```
Path path1 = new File("/home/oracle").toPath();
Path path2 = FileSystems.getDefault().getPath("/home/java/../hello.txt");
Path path3 = /* insert code here */
System.out.println(path3);
```

when inserted at /* insert code here */, which option will output /home/hello.txt?
(Choose the best option.)

- ☐ a path1.resolve(path2);
- ☐ b path2.normalize(path1);

- ☐ c `path1.relativeTo(path2);`
- ☐ d `path2.relativeTo(path1);`
- ☐ e `path2.normalize();`

ME-Q24. Given

```
class MyException extends Exception{}
```

which method will compile successfully? (Choose the best option.)

- ☐ a `void methodA() throws Exception {}`
- ☐ b `void methodB() throws RuntimeException {}`
- ☐ c `void methodC() throws FileNotFoundException {}`
- ☐ d `void methodD() throws MyException {}`
- ☐ e All of the above
- ☐ f None of the above

ME-Q25. What is the output of the following code? (Choose the best option.)

```
public class TestSplit {  
    public static void main(String[] args) {  
        String names = "Shreya;.-Selvan;.-Paul;.-Harry";  
        String[] results = names.split(";.-");  
        for(String str:results) System.out.print(str);  
    }  
}
```

- ☐ a ShreyaSelvanPaulHarry
- ☐ b Shreya-Selvan-Paul-Harry
- ☐ c Shreya.-Selvan.-Paul.-Harry
- ☐ d Shreya;.-Selvan;.-Paul;.-Harry

ME-Q26. Which option when inserted at `//insert code here` copies a source file to the destination, replacing any existing file with the same name? (Choose the best option.)

```
void move(Path src, Path dest) throws Exception {  
    //insert code here  
}
```

- ☐ a `Files.copy(src, dest);`
`Files.delete(src);`
- ☐ b `Files.copy(src, dest, true);`
`Files.deleteIfExists(src);`
- ☐ c `Files.copy(src, dest, StandardCopyOption.REPLACE_EXISTING);`
`Files.deleteIfExists(src);`
- ☐ d `Files.replace(src, dest);`
`Files.delete(src);`

ME-Q27. Given that contents of in.txt are

day

what is the output of the following code? (Choose the best option.)

```
import java.io.*;
class ReadWrite {
    public static void main(String args[]) throws Exception {
        BufferedReader in = new BufferedReader(new FileReader("in.txt"));
        BufferedWriter out = new BufferedWriter(new FileWriter("out.txt"));
        int i = in.read();
        int ctr = 0;
        while (i != -1) {
            out.write(++ctr + (char)i);
            i = in.read();
        }
        out.flush();
        out.close();
    }
}
```

- ☐ a File out.txt contains 1.d.2.a.3.y.
- ☐ b File out.txt contains
 - 1.d
 - 2.a
 - 3.y
- ☐ c File out.txt contains 1.d2.a3.y.
- ☐ d Code fails to compile
- ☐ e None of the above

ME-Q28. Assuming that c is a valid Connection object, which of the following can be used to execute the stored procedure updateExamDetails and also retrieve the updated data from the database? (Choose the best option.)

- ☐ a c.prepareCall("{call updateExamDetails()}").executeUpdate();
- ☐ b c.prepareCall(" {call updateExamDetails()}").execute();
- ☐ c c.prepareCall("{call updateExamDetails()}").executeQuery();
- ☐ d c.callableStatement(" {call updateExamDetails()}");
- ☐ e c.callProcedure(" {call updateExamDetails()}");

ME-Q29. Given

```
class Device {}
class Phone extends Device {
    void model() {}
}
class MobilePhone extends Phone {}
```

which options can be used to access method `model()` using a reference variable, `d`, defined as follows? (Choose two options.)

```
Device d = new MobilePhone();
```

- ☐ a `d.model()`;
- ☐ b `((Phone)d).model()`;
- ☐ c `((MobilePhone)d).model()`;
- ☐ d `((MobilePhone.Phone)d).model()`;

ME-Q30. Which statements for the given code are correct? (Choose two options.)

```
Integer i1 = 20;
String s1 = "Shreya";
boolean b1 = false;
char c1 = 109;
```

- ☐ a `System.out.printf("%s %b %c", i1, s1, c1, b1);` will output values without any exceptions.
- ☐ b `System.out.printf("%s %b %c", i1, s1);` won't compile.
- ☐ c `System.out.printf("%s %i %c", i1, s1);` will throw a runtime exception.
- ☐ d `System.out.printf("%sd", i1);` will throw a runtime exception.

ME-Q31. Given that code snippet at `//line1` can throw a `NullPointerException`, `FileNotFoundException`, and `SQLException`, which of the following are appropriate catch handles? (Choose two options.)

```
try {
    //line1
}
```

- ☐ a `catch (SQLException e | FileNotFoundException ex) {}`
- ☐ b `catch (SQLException e) {}`
- ☐ c `catch (SQLException e) {}`
`catch (NullPointerException | FileNotFoundException e) {}`
- ☐ d `catch (FileNotFoundException | SQLException e) {}`

ME-Q32. Assuming all directories exist, what is the result of the following code? (Choose the best option.)

```
class MyFileVisitor extends SimpleFileVisitor<Path> {
    String backupDir = "e:\\code\\backup\\";
    public FileVisitResult visitFile
        (Path file, BasicFileAttributes attrs) {
        if (file.toString().endsWith(".java")) {
            try {
                String fileName = file.getFileName().toString();
                Files.copy(file, Paths.get(backupDir + fileName),
                    StandardCopyOption.REPLACE_EXISTING);
            }
        }
    }
}
```



```

        } catch (Exception e) {}
    }
    return FileVisitResult.CONTINUE;
}
public static void main(String args[]) throws Exception {
    Files.walkFileTree(Paths.get("e:/code"), new MyFileVisitor());
}
}

```

- ☐ a It copies all .java files from e:/code to e:/code/backup.
- ☐ b It copies all .java files recursively from e:/code and its subdirectories to e:/code/backup.
- ☐ c It copies all non .java files from e:/code to e:/code/backup.
- ☐ d It throws a runtime exception for duplicate .java files found in e:/code or any of its subdirectories.
- ☐ e It modifies the type of directory from e:/code/backup.
- ☐ f It fails to compile.

ME-Q33. What is the output of the following code? (Choose the best option.)

```

Pattern p = Pattern.compile("\\Sthe\\S");
Matcher m = p.matcher("These are their,leather,,them");
System.out.println(m.replaceAll("<>"));

```

- ☐ a These are their,le<>,,<>
- ☐ b These are <>ir,leather,,them
- ☐ c These are <>ir,leather,<>m
- ☐ d These are <>ir,lea<>r<>m
- ☐ e These are their,lea<>r,<>m

ME-Q34. Given

```

class Wave extends Exception {}
class Oar extends Exception {}
interface Sail{
    void surf() throws Wave, Oar;
}
interface SuperSail {
    void surf() throws Wave;
}
class Surfer implements SuperSail {
    //INSERT CODE HERE
}

```

which option when inserted independently at //INSERT CODE HERE will enable class Surfer to compile successfully? (Choose two options.)

- ☐ a public void surf() throws Wave {}
- ☐ b public void surf() throws Wave, Oar {}

- ☐ c public void surf() throws Wave {}
public void surf() throws Wave, Oar {}
- ☐ d public void surf() {}
- ☐ e public void surf() throws Exception {}

ME-Q35. What are the subtypes of `List<?>`? (Choose two correct options.)

- ☐ a `List<String>`
- ☐ b `ArrayList<String>`
- ☐ c `LinkedList<?>`
- ☐ d `List<LinkedList>`

ME-Q36. Given

```
Table Book
ID, INTEGER: PK
TITLE VARCHAR(100)
```

what is the output of the following code? (Choose the best option.)

```
class Transact {
    public static void main(String[] args) {
        Connection con = null;
        Statement st = null;
        try {
            con = DriverManager.getConnection("jdbc:mysql://localhost/abc",
                                             "a", "b");

            con.setAutoCommit(false);
            st = con.createStatement();
            st.executeUpdate("INSERT INTO book values(1, 'Java')");
            Savepoint sv1 = con.setSavepoint("sv1");
            st.executeUpdate("INSERT INTO book values(3, 'PHP')");
            con.rollback();
            st.executeUpdate("INSERT INTO book values(4, 'Android')");
            con.setAutoCommit(true);
            st.executeUpdate("INSERT INTO book values(2, 'Oracle')");

            ResultSet rs = st.executeQuery("SELECT * from BOOK");
            int ctr = 0;
            while (rs.next()) ctr++;
            System.out.println(ctr);
        }
        catch (Exception e) {System.out.println("Exception");}
    }
}
```

- ☐ a 1
- ☐ b 2
- ☐ c 3
- ☐ d 4
- ☐ e The code fails to compile.

ME-Q37. Which of the following are valid assignments? (Choose five options.)

- ☐ a `List<Number> list = new ArrayList<Integer>();`
- ☐ b `List<?> list = new ArrayList<Integer>();`
- ☐ c `List<? extends Number> list = new ArrayList<Integer>();`
- ☐ d `List<? super Number> list = new ArrayList<>();`
- ☐ e `ArrayList<Number> list = new ArrayList<Integer>();`
- ☐ f `ArrayList<?> list = new ArrayList<Integer>();`
- ☐ g `ArrayList<? extends Number> list = new ArrayList<Integer>();`
- ☐ h `ArrayList<? implements Number> list = new ArrayList<>();`

ME-Q38. You have a file that stores your top secrets. How can you modify its attributes so that it becomes read-only and hidden? Assume you're using a DOS-based filesystem. (Choose the best option.)

```
void modifyAttr(Path file) {
    // insert code here
}
```

- ☐ a `Files.setAttribute(file, "dos:hidden", true);`
`Files.setAttribute(file, "dos:readonly", true);`
- ☐ b `try {`
 `Files.setAttribute(file, "dos:hidden", true);`
 `Files.setAttribute(file, "dos:readonly", true);`
 `}`
 `catch (IOException e) {}`
- ☐ c `try {`
 `Map<String, Object> values =`
 `Files.readAttributes(file, "dos:readonly,hidden");`
 `values.put("readonly", new Boolean(true));`
 `values.put("hidden", new Boolean(true));`
 `Files.setAttributes(file, values);`
 `}`
 `catch (IOException e) {}`
- ☐ d `try {`
 `Map<String, Object> values = Files.readAttributes(file, "dos:*");`
 `values.put("readonly", new Boolean(true));`
 `values.put("hidden", new Boolean(true));`
 `Files.setAttributes(file, values);`
 `}`

ME-Q39. What is the output of the following code? (Choose the best option.)

```
List<Integer> list = new ArrayList<>();
list.add(new Integer(30));
list.add(10);
Arrays.sort(list);
for (Integer i : list)
    System.out.println(i);
```

- ☐ a 10
30
- ☐ b 30
10
- ☐ c Compilation error
- ☐ d RuntimeException

ME-Q40. To be eligible to be declared using a try-with-resources statement, which interfaces should a class implement? (Choose two options.)

- ☐ a java.io.Resource
- ☐ b java.io.Closeable
- ☐ c java.lang.Resource
- ☐ d java.lang.AutoCloseable
- ☐ e java.lang.Closeable
- ☐ f java.io.AutoCloseable

ME-Q41. What is the output of the following code? (Choose the best option.)

```
import java.util.*;
class TestTreeSet {
    public static void main(String args[]) {
        TreeSet<StringBuilder> treeSetNames = new TreeSet<StringBuilder>();
        treeSetNames.add(new StringBuilder("Shreya"));
        treeSetNames.add(new StringBuilder("Harry"));
        treeSetNames.add(new StringBuilder("Paul"));
        treeSetNames.add(new StringBuilder("Shreya"));
        Iterator it = treeSetNames.descendingIterator();
        while (it.hasNext())
            System.out.print(it.next() + ":");
    }
}
```

- ☐ a Shreya:Harry:Paul:Shreya:
- ☐ b Shreya:Harry:Paul:
- ☐ c Shreya:Shreya:Paul:Harry:
- ☐ d Harry:Paul:Shreya:
- ☐ e Compilation error
- ☐ f RuntimeException

ME-Q42. Which option is true for the given code? (Choose the best option.)

```
assert(result):new RuntimeException();
```

- ☐ a If result is false, the code will throw a RuntimeException.
- ☐ b If result is true, the code will throw a RuntimeException.

- ☐ c If result is false, the code will throw an AssertionError.
- ☐ d If result is true, the code will throw an AssertionError.

ME-Q43. Given that a company's application, Effective Customer Relation Management (ECRM) is localized, which option is correct? (Choose the best option.)

- ☐ a ECRM is efficient.
- ☐ b ECRM is highly cohesive and loosely coupled.
- ☐ c ECRM displays text messages according to a user language and region.
- ☐ d ECRM always loads localized text messages from a .properties file.
- ☐ e ECRM defines local methods.

ME-Q44. Two threads (one and two in class EJavaGuru) can't reach completion. What is the probable cause? (Choose the best option.)

```
enum Location {COLLEGE, HOME};
class Student {
    Location loc;
    boolean friendFound = false;
    Student(Location loc) {
        this.loc = loc;
    }
    void changeLoc() {
        if (this.loc.equals(Location.COLLEGE))
            this.loc = Location.HOME;
        else
            this.loc = Location.COLLEGE;
    }
    boolean find(Student friend) {
        return (friendFound = this.loc.equals(friend.loc));
    }
}
class Find extends Thread{
    Student s, friend;
    Find (Student s, Student friend) {
        this.s = s;
        this.friend = friend;
    }
    public void run() {
        while (!s.friendFound) {
            s.friendFound = s.find(friend);
            if (!s.friendFound) s.changeLoc();
            try {Thread.sleep(1000);} catch (InterruptedException e) {}
        }
    }
}
class EJavaGuru2 {
    public static void main(String args[]) {
        Student s1 = new Student(Location.HOME);
        Student s2 = new Student(Location.COLLEGE);
        Thread one = new Find(s1, s2);
```

```

        Thread two = new Find(s2, s1);
        one.start();
        two.start();
    }
}

```

- ☐ a Deadlock
- ☐ b Livelock
- ☐ c Starvation
- ☐ d Mutual lock

ME-Q45. Which interfaces should be implemented by all JDBC drivers? (Choose two options.)

- ☐ a Driver
- ☐ b DriverManager
- ☐ c JDBCConnection
- ☐ d Connection
- ☐ e DatabaseConnection

ME-Q46. Given

```

import java.util.*;
enum Title {Mr, Ms, Dr};
class Person implements Comparable<Person> {
    String name; Title title;
    public Person(Title t, String n) {
        title = t;
        name = n;
    }
    public int compareTo(Person p) {
        int compareName = name.compareTo(p.name);
        return compareName != 0 ? compareName : title.compareTo(p.title);
    }
    public String toString() {
        return title + " " + name;
    }
}
class University {
    public static void main(String args[]) {
        ArrayList<Person> list = new ArrayList<>();
        list.add(new Person(Title.Mr, "Selvan"));
        list.add(new Person(Title.Dr, "Shreya"));
        list.add(new Person(Title.Ms, "Shreya"));
        Collections.sort(list);
        for (Person p : list) {
            System.out.println(p);
        }
    }
}

```

what is the output? (Choose the best option.)

- ☐ a Mr Selvan
Dr Shreya
Ms Shreya
- ☐ b Mr Selvan
Ms Shreya
Dr Shreya
- ☐ c Ms Shreya
Dr Shreya
Mr Selvan
- ☐ d Dr Shreya
Mr Selvan
Ms Shreya
- ☐ e Ms Shreya
Mr Selvan
Dr Shreya

ME-Q47. Which statements are incorrect? (Choose three options.)

- ☐ a “Program to an interface” means using interfaces in your code.
- ☐ b An interface can’t define static inner classes.
- ☐ c An interface can’t declare nonfinal variables.
- ☐ d An interface can declare uninitialized variables.
- ☐ e An interface can’t define static methods.

ME-Q48. Identify the classes whose instances are thread safe and don’t need external synchronization. (Choose four options.)

- ☐ a Integer
- ☐ b StringBuilder
- ☐ c ArrayList
- ☐ d Map
- ☐ e Boolean
- ☐ f String
- ☐ g Double
- ☐ h LinkedList

ME-Q49. You need to design a mining application and have identified the following metallic and nonmetallic entities: mine, copper, and gold. Assuming the natural properties of these entities, which option do you think best shows the relationship among these entities? (Choose one option.)

- ☐ a

```
abstract class Metal{}  
abstract class Mine{}  
class Copper extends Mine{}  
class Gold extends Metal{}
```

- ☐ **b**

```
abstract class Metal{
    abstract class Mine{
        class Copper extends Metal{
            abstract class Gold extends Metal{
```
- ☐ **c**

```
abstract class Mine{
    class Metal extends Mine{
        class Copper extends Metal{
            class Gold extends Metal{
```
- ☐ **d**

```
class Mine{
    abstract class Metal {
        abstract void extract();
    }
    class Copper implements Metal{
        class Gold implements Metal{
```

ME-Q50. Given

```
class Gift{
class Book extends Gift{
class Phone extends Gift{
```

which options when inserted at //INSERT CODE HERE will enable you to add elements of type Phone to books? (Choose two options.)

```
List<Book> books = new ArrayList<>();
books.add(new Book());
List anotherList = books;
//INSERT CODE HERE
```

- ☐ **a**

```
List<Gift> gifts = anotherList;
gifts.add(new Phone());
```
- ☐ **b**

```
List<? extends Gift> gifts = anotherList;
gifts.add(new Phone());
```
- ☐ **c**

```
List<? super Gift> gifts = anotherList;
gifts.add(new Phone());
```
- ☐ **d**

```
List<?> gifts = anotherList;
gifts.add(new Phone());
```

ME-Q51. Which option defines loosely coupled classes? (Choose the best option.)

- ☐ **a**

```
class Pen {
    class Ink {
```
- ☐ **b**

```
class Pen {
    public void refill(Ink ink) {
        if (ink.color.equals("red")) { /*code */ }
    }
    class Ink { String color; }
```


- ☐ c
- ```
class Pen {
 public void refill(String color) {
 if (color.equals("red")) {
 buy(new Ink("red"));
 }
 }
 public void buy(Ink ink) {
 //code
 }
}
class Ink {
 String color;
 Ink(String color) { this.color = color; }
}
```
- ☐ d
- ```
class Pen {
    Ink ink;
    Pen(Ink ink) {
        if (ink.color.equals("black") ink = new BlackInk();
        if (ink.color.equals("red") ink = new RedInk();
    }
}
interface Ink {}
class RedInk implements Ink{
    String color;
}
class BlackInk implements Ink{
    String color;
}
```

ME-Q52. Which option will output the following? (Choose the best option.)

(99,887,766)

- ☐ a `System.out.printf("%(,d", -99887766);`
- ☐ b `System.out.printf("%,d", -99887766);`
- ☐ c `System.out.printf("%(),f", -99887766);`
- ☐ d `System.out.printf("%,l", -99887766);`
- ☐ e `System.out.printf("%,i", -99887766);`
- ☐ f `System.out.printf("%(,i)", -99887766);`
- ☐ g `System.out.printf("(%d)", -99887766);`
- ☐ h `System.out.printf("(%i)", -99887766);`

ME-Q53. Which option correctly implements a Singleton pattern? (Choose the best option.)

- ☐ a
- ```
class Heart {
 private static Heart instance;
 private Heart () {}
}
```

- ```

        public static synchronized Heart getInstance() {
            if (instance == null) {
                instance = new Heart ();
            }
            return instance;
        }
    }
}

```
- ☐ **b** class Heart {
 private Heart instance;
 private Heart () {}
 public synchronized Heart getInstance() {
 if (instance == null) {
 instance = new Heart ();
 }
 return instance;
 }
}
- ☐ **c** class Heart {
 Heart () {}
 private static class HeartHolder {
 private static final Heart INSTANCE = new Heart ();
 }
 public static Heart getInstance () {
 return HeartHolder.INSTANCE;
 }
}
- ☐ **d** class Heart {
 private static Heart instance = new Heart();
 protected Heart () {}
 public static Heart getInstance () {
 return instance;
 }
}

ME-Q54. Given that method `getClass()` returns the name of an object's class, what is the output of the following code? (Choose the best option.)

```
System.out.println((new Integer(10) + new Short((short)100)).getClass());
```

- ☐ **a** class java.lang.Short
- ☐ **b** class java.lang.Integer
- ☐ **c** short
- ☐ **d** int
- ☐ **e** Compilation error
- ☐ **f** RuntimeException

ME-Q55. Given the following code, which option when inserted at //INSERT CODE HERE will enable you to compile the code? (Choose the best option.)

```
//INSERT CODE HERE
class Foo {
    public static void main(String args[]) {
        Locale locale1 = Locale.FRENCH;
        Locale locale2 = Locale.FRANCE;
        Locale locale3 = new Locale.Builder().setLanguage("fr").build();
        Locale locale4 = new Locale.Builder().setLanguage("fr")
                                                .setRegion("FR").build();

        Locale locale5 = new Locale("fr");
        Locale locale6 = new Locale("fr", "FR");
        System.out.print(locale1.equals(locale3));
        System.out.print(locale2.equals(locale6));
        System.out.print(locale4.equals(locale5));
    }
}
```

- ☐ a `import java.util.Locale*;`
- ☐ b `import java.util.locale.*;`
- ☐ c `import java.util.locatisation*;`
- ☐ d `import java.util.il8n.*;`
- ☐ e `import java.util.*;`

ME-Q56. Which options correctly define generic classes? (Choose three options.)

- ☐ a `class MyClass1 <T, A, B> { /*use only T and B*/ }`
- ☐ b `class MyClass2 <T, A, B> { /*use T, A and B*/ }`
- ☐ c `class MyClass4 <Var1, Var2> { /*...*/ }`
- ☐ d `class MyClass5 <?, ?> { /*...*/ }`
- ☐ e `class MyClass6 <U extends Number, T super Number> { /*...*/ }`

ME-Q57. Given

```
Locale locale = Locale.US;
ResourceBundle labels = null;
```

which option loads the resource bundle `MyMsgs` for `Locale.US`? (Choose the best option.)

- ☐ a `labels = ResourceBundle.getBundle("MyMsgs", locale);`
- ☐ b `labels = ResourceBundle.loadBundle("MyMsgs", locale);`
- ☐ c `labels = ResourceBundle.loadBundleFamily("MyMsgs", locale);`
- ☐ d `labels = ResourceBundle.getBundleMessages("MyMsgs", locale);`

ME-Q58. Which code examples don't initialize variables using a factory? (Choose two options.)

- ☐ **a** `Connection con = /*code to get a valid connection instance */
Statement statement = con.createStatement();`
- ☐ **b** `Statement st = /*code to get a valid Statement instance */
ResultSet rs = st.executeQuery("SELECT * FROM book");`
- ☐ **c** `NumberFormat nf = new NumberFormat();`
- ☐ **d** `NumberFormat nf = new NumberFormat(Locale.JP);`
- ☐ **e** `NumberFormat nf = NumberFormat.getCurrencyInstance();`
- ☐ **f** `ResourceBundle myBundle = ResourceBundle.getBundle("myBundle", Locale.FR);`

ME-Q59. Which method is defined in the Executor interface? (Choose one option.)

- ☐ **a** `void execute(Runnable obj)`
- ☐ **b** `void execute(Callable obj)`
- ☐ **c** `void execute(Thread obj)`
- ☐ **d** `void execute(Executable obj)`

ME-Q60. Given

```
interface Movable {
    void move(int x, int y);
}
```

which code snippet will compile successfully? (Choose one option.)

- ☐ **a** `interface Jumpable extends Movable {}`
- ☐ **b** `interface Jumpable implements Movable {}`
- ☐ **c** `class Position {
 interface Jumpable extends Movable {
 Position move(int posX, int posY);
 }
}`
- ☐ **d** `interface Jumpable extends Movable {
 private String move(long x, int y);
}`

ME-Q61. Given

```
import java.util.List;
public class Book {
    private String isbn;
    private String title;
    private String author;
    public void setISBN(String val) { isbn = val; }
    public String getISBN() { return this.isbn; }
    public void setTitle(String val) { title = val; }
    public String getTitle() { return title; }
    public void setAuthor(String val) { author = val; }
```

```

public String getAuthor() { return author; }
public void addBook(Book b) throws Exception { /* code */}
public void removeBook(String isbn) throws Exception { /* code */}
public void updateBook(Book b) throws Exception { /* code */}
public Book getBook(String isbn) throws Exception {return null;}
public List getAllBooks() throws Exception {return null;}
}

```

which set of methods should be moved to a new class to implement the DAO pattern? (Choose the best option.)

- ☐ a `getISBN(), getTitle(), getAuthor()`
- ☐ b `setISBN(), setTitle(), setAuthor()`
- ☐ c `getISBN(), getTitle(), getAuthor(), setISBN(), setTitle(), setAuthor()`
- ☐ d `addBook(), removeBook(), updateBook(), getBook(), getAllBooks()`

ME-Q62. Which option is correct for the following code? (Choose the best option.)

```

class MyPen extends Thread {
    public void run() {
        synchronized(System.out) {
            try {
                Thread.sleep(5000);           //line1
            }
            catch (InterruptedException e) {}
        }
    }
}

```

- ☐ a On //line1, the MyPen thread will give up ownership of the monitor on System.out for at least 5,000 ms.
- ☐ b On //line1, the MyPen thread will give up ownership of the monitor on System.out for exactly 5,000 ms.
- ☐ c On //line1, the MyPen thread won't give up ownership of the monitor on System.out.
- ☐ d None of the above

ME-Q63. An application needs to load its localized messages from a Java class. Which option defines the correct class definition? (Choose the best option.)

```

☐ a public class MyMessages extends ResourceBundle {
    protected Object[][] getContents() {
        return new Object[][] {
            {"day", "Day"},
            {"time", "Time"}
        };
    }
}

```

- ☐ b

```
public class MyMessages extends ListResourceBundle {
    protected Object[][] getContents() {
        return new Object[][] {
            {"day", "Day"},
            {"time", "Time"}
        };
    }
}
```
- ☐ c

```
public class MyMessages extends ClassResourceBundle {
    protected Set<String> handleKeySet() {
        return new HashSet<String>(Arrays.asList("day"));
    }
}
```
- ☐ d

```
public class MyMessages extends Bundle {
    public Object[][] getValues() {
        return new Object[][] {
            {"day", "Day"},
            {"time", "Time"}
        };
    }
}
```

ME-Q64. You coded a text editor in Java that can be used to open, edit, and save multiple text files. Your application should reload a text file, if it's modified by any other application. Assuming that your target directory is `/home/selvan`, which option can you use to register the relevant events for your directory? (Choose the best option.)

- ☐ a

```
Path dir = Paths.get("/home/selvan");
WatchService watcher = FileSystems.getDefault().newWatchService();
WatchKey key = dir.register(StandardWatchEventKinds.ENTRY_MODIFY);
```
- ☐ b

```
Path dir = Paths.get("/home/selvan");
WatchService watcher = FileSystems.getDefault().newWatchService();
WatchKey key = dir.register(watcher,
    StandardWatchEventKinds.ENTRY_MODIFY);
```
- ☐ c

```
Path dir = Paths.get("/home/selvan");
WatchService watcher = FileSystems.getDefault().newWatchService();
WatchKey key = dir.register(watcher,
    StandardWatchEventKinds.WATCH_MODIFY);
```
- ☐ d

```
Path dir = Paths.get("/home/selvan");
WatchKey key = dir.register(Watcher.getInstance(),
    StandardWatchEventKinds.WATCH_MODIFY);
```

ME-Q65. Which method can be used to replace a key-value pair if it exists in a `ConcurrentMap`? (Choose the best option.)

- ☐ a `replace(K key, V value)`
- ☐ b `replace(K key, V old, V new)`
- ☐ c `replaceIfPresent(K key, V old, V new)`
- ☐ d `modifyIfPresent(K key, V old, V new)`
- ☐ e `removeIfPresent(K key, V old, V new)`

ME-Q66. Given

```
public class Inner {  
    public class Outer {}  
}
```

which of the following instantiates class `Outer` in another class, say, `Test`? (Choose one option.)

- ☐ a `new Outer().new Inner();`
- ☐ b `new Inner().new Outer();`
- ☐ c `Outer.Inner();`
- ☐ d `Outer.new Inner();`
- ☐ e `Inner.Outer();`
- ☐ f `Inner.new Outer();`
- ☐ g `new Inner.Outer();`
- ☐ h `new Outer.Inner();`

ME-Q67. Given

```
class Laptop {  
    static class Model {}  
}
```

which option can be assigned to a variable of type `Laptop.Model`? (Choose the best option.)

- ☐ a `new Laptop.Model();`
- ☐ b `new Laptop().new Model();`
- ☐ c `Laptop.Model;`
- ☐ d `Laptop.new Model();`

ME-Q68. Given

```
1. ExecutorService pool = Executors.newFixedThreadPool(10);  
2. Job job = new Job();  
3. pool.submit(job);
```

which options define the correct definition of class `Job`, which compiles without errors or warnings? (Choose two options.)

- ☐ a `class Job implements Callable {
 public void call() {}
}`
- ☐ b `class Job implements Callable<Void> {
 public Void call() {}
}`

```

☐ c class Job implements Callable<String> {
    public String call() {
        return "Success";
    }
}

☐ d import java.io.*;
class Job implements Callable<File> {
    public File call() throws FileNotFoundException {
        return new File("abcd");
    }
}

```

ME-Q69. Given that labels refers to a valid ResourceBundle resource the contents of which are

```

day=7
time=true

```

which options can be used to read the values of the keys "day" and "time"? (Choose two options.)

```

☐ a labels.getInt("day");
    labels.getBoolean("time");

☐ b labels.getValue("day");
    labels.getValue("time");

☐ c labels.readValue("day");
    labels.readValue("time");

☐ d labels.getString("day");
    labels.getString("time");

☐ e labels.getObject("day");
    labels.getObject("time");

```

ME-Q70. Given

```

class Gift{}
class Book extends Gift{}
class Phone extends Gift{}

```

and the code

```

List<Book> books = new ArrayList<>();
books.add(new Book());
List anotherList = books;
List<? super Gift> gifts = anotherList;
gifts.add(new Phone());
//INSERT CODE HERE
System.out.println(item);

```

when inserted at //INSERT CODE HERE, which options can be used to output all values of collection gifts? (Choose two options.)

- ☐ a for (Book item : books)
- ☐ b for (Gift item : books)
- ☐ c for (Object item : books)
- ☐ d for (Phone item : books)

ME-Q71. Assuming that classes `Color` and `Room` are defined in separate packages and source files

```
package artist;
public enum Color {
    RED, YELLOW, BLUE;
}

package city;
import artist.Color;
class Room {
    Color myColor = /*insert code here*/
}
```

which options when inserted at `/*insert code here*/` will assign the enum `Color` constant `RED` to the variable `myColor`? (Choose two options.)

- ☐ a `Color.RED;`
- ☐ b `RED;`
- ☐ c `artist.Color.RED;`
- ☐ d `new Color("RED");`

ME-Q72. Given

```
Executor executor = /* assigns an Executor instance */
Runnable runnable = /* assigns a Runnable instance */
```

how many threads will execute the `Runnable` task for the following code? (Choose the best option.)

```
executor.execute(runnable);
```

- ☐ a One for each submitted task
- ☐ b Only one thread for each task
- ☐ c Depends on the `Executor` implementation
- ☐ d Depends on the property `Executor.threads` set during runtime

ME-Q73. What is the output of the following code? (Choose one option.)

```
Path path1 = Paths.get("C:/OCP/8-1.txt");
Path path2 = Paths.get("C:", "OCP", "mock", "8-1.txt");
Path path3 = path1.resolve(path2.relativize(path1));
System.out.println(path3);
```

- ☐ a ..\mock\8-1.txt
- ☐ b ../../8-1.txt
- ☐ c C:\OCP\8-1.txt\..\..\8-1.txt
- ☐ d C:\OCP\8-1.txt\..\mock\8-1.txt
- ☐ e C:\8-1.txt

ME-Q74. Given

```
enum Shade {LIGHT, DARK};
class Color {
    final int shade;
    Color(Shade val){
        //initialize
    }
}
```

which code snippet when inserted at //initialize will enable class Color to compile successfully? (Choose two options.)

- ☐ a

```
switch (val) {
    case LIGHT : shade = 11; break;
    case DARK : shade = 22; break;
    default: shade = 33;
}
```
- ☐ b

```
switch (val) {
    case LIGHT : shade = 11;
    case DARK : shade = 22;
    default: shade = 33;
}
```
- ☐ c

```
switch (val) {
    case LIGHT : shade = 11; break;
    case DARK : shade = 22; break;
}
```
- ☐ d

```
if (val.equals(Shade.LIGHT)) shade = 11;
else shade = 22;
```
- ☐ e

```
if (val.equals(Shade.LIGHT)) shade = 11;
else if (val.equals(Shade.DARK)) shade = 22;
```

ME-Q75. What is the result of the following code? (Choose the best option.)

```
class Metal {
{
    try {
        throw new RuntimeException();
    }
    finally{
        System.out.print("finally-");
    }
}
```

```
Metal() {  
    System.out.print("Metal-");  
}  
}  
class Copper extends Metal{  
    Copper(int a) {  
        System.out.print("Copper-");  
    }  
    public static void main(String args[]) {  
        new Copper(101);  
    }  
}
```

- ☐ a catch-finally-Metal-Copper-
- ☐ b Metal-catch-finally-Copper-
- ☐ c RuntimeException
- ☐ d Compilation error

ME-Q76. What is the correct option for the given code? (Choose the best option.).

```
Executors.newFixedThreadPool(5);
```

- ☐ a The code creates a thread pool that reuses five threads operating off a shared, unbounded queue.
- ☐ b The code creates five thread pools.
- ☐ c The code creates five thread pools with a predefined number of active threads.
- ☐ d The code creates a thread pool that can create multiple threads, but reuses at least five of them off a shared, unbounded queue.

ME-Q77. What is the result of the following code? (Choose one option.)

```
class A {  
    static int age = 10;  
}  
interface B {  
    int age = 20;  
}  
class C extends A implements B {  
    static int age = 30;  
}  
class MyTest {  
    public static void main(String args[]) {  
        B b = new C();  
        System.out.println(b.age);  
    }  
}
```

- ☐ a 10
- ☐ b 20

- ☐ c 30
- ☐ d Compilation error
- ☐ e RuntimeException

ME-Q78. The default locale of a JVM is `Locale.JAPAN`, and an application includes the following resource bundles:

- `AppMessages_fr.properties`
- `AppMessages_fr_FR.properties`
- `AppMessages_en.properties`
- `AppMessages_ja_JP.properties`
- `AppMessages_ja.properties`
- `AppMessages_JP.properties`

Which resource bundle will the application load for `Locale.CHINA`? (Choose one option.)

- ☐ a `AppMessages_fr.properties`
- ☐ b `AppMessages_fr_FR.properties`
- ☐ c `AppMessages_en.properties`
- ☐ d `AppMessages_ja_JP.properties`
- ☐ e `AppMessages_ja.properties`
- ☐ f `AppMessages_JP.properties`
- ☐ g Compilation error
- ☐ h RuntimeException

ME-Q79. Given

```
abstract class Wood {  
    public abstract void floats();  
}
```

which option extends class `Wood` correctly? (Choose the best option.)

- ☐ a `abstract class Boat extends Wood {}`
- ☐ b `class Boat extends Wood {
 float floats() {}
}`
- ☐ c `final class Boat extends Wood {
 abstract void floats() {}
}`
- ☐ d `class Boat extends Wood {
 void floats() {}
}`

ME-Q80. Which statement is true for the given code? (Choose the best option.)

```
class Foo extends Thread {
    String name;
    Foo(String name) {this.name = name;}
    public static int start = 10;
    public static int end = 20;
    public void run() {
        for (; start < end; start++)
            System.out.println(name + start);
    }
}

class TestFoo {
    public static void main(String args[]) {
        Thread t1 = new Foo("T1:"); t1.start();
        Thread t2 = new Foo("T2:"); t2.start();
    }
}
```

- ☐ a The code will output exactly 10 numbers.
- ☐ b The code will output exactly 20 lines.
- ☐ c Threads t1 and t2 can output the same start value.
- ☐ d Threads t1 and t2 will never output the same start values.
- ☐ e Thread t1 can never print a lower start value than that printed by thread t2.

ME-Q81. Given

```
import java.io.*;
class Factory {
    static int count = initCount();
    static int initCount() throws FileNotFoundException {
        int result = 0;
        FileInputStream fin = new FileInputStream(new File("abc.txt"));
        //read fin and initialize result
        return result;
    }
}
```

which option is correct? (Choose the best option.)

- ☐ a Class Factory will throw a FileNotFoundException if file abc.txt can't be found.
- ☐ b Class Factory will throw a FileNotFoundException irrespective of whether or not file abc.txt can't be found.
- ☐ c The code will initialize static variable count successfully.
- ☐ d Compilation error

ME-Q82. Two threads can't reach completion. What is the probable cause? (Choose the best option.)

```
class MyThread extends Thread{
    Runnable other;
    void setOther(Runnable r) {other = r;}
    public void run() {
        synchronized(this) {
            System.out.print("XYZ");
            synchronized(other) {
                System.out.print("ABC");
            }
        }
    }
}

class EJavaGuru {
    public static void main(String args[]) {
        MyThread one = new MyThread();
        MyThread two = new MyThread();
        one.setOther(two);
        two.setOther(one);
        one.start();
        two.start();
    }
}
```

- ☐ a Deadlock
- ☐ b Livelock
- ☐ c Starvation
- ☐ d Mutual lock

ME-Q83. Given

```
enum Metal {
    COPPER, GOLD;
    Metal() {
        System.out.print("constructor:");
    }
    static {
        System.out.print("static:");
    }
    public static void main(String args[]) {
        System.out.print(Metal.COPPER + ":");
    }
}
```

what is the result? (Choose the best option.)

- ☐ a COPPER:static:constructor:constructor:
- ☐ b static:constructor:constructor:COPPER:
- ☐ c constructor:constructor:static:COPPER:
- ☐ d COPPER:constructor:constructor:static:

ME-Q84. Given

```
enum Color {VIOLET, INDIGO, BLUE, GREEN, YELLOW, ORANGE, RED}
```

when started as a thread, instances of which class would output the enum values with an interval of at least one second? (Choose the best option.)

- ☐ a class Rainbow extends Thread {


```
    public void run() {
        for (Color color : Color.values()) {
            System.out.println(color);
        }
    }
}
```
- ☐ b class Rainbow implements Runnable {


```
    public void run() {
        for (Color color : Color.values()) {
            System.out.println(color);
        }
    }
}
```
- ☐ c class Rainbow extends Thread {


```
    public void run() {
        for (Color color : Color.values()) {
            Thread.sleep(1000);
            System.out.println(color);
        }
    }
}
```
- ☐ d class Rainbow extends Thread {


```
    public void run() throws InterruptedException {
        for (Color color : Color.values()) {
            Thread.sleep(1000);
            System.out.println(color);
        }
    }
}
```
- ☐ e None of the above

ME-Q85. What is the result of the following code? (Choose the best option.)

```
import java.util.*;
class Color {
    String value;
    Color(String v) {value = v;}
    public int hashCode() {return 999; }
    public String toString() {return value;}
}
class Rainbow {
    public static void main(String args[]) {
        HashSet<Color> set = new HashSet<>();
        set.add(new Color("red"));
        set.add(new Color("yellow"));
        set.add(new Color("blue"));
    }
}
```

```
        Iterator it = set.iterator();
        while (it.hasNext())
            System.out.print(it.next() + "-");
    }
}
```

- ☐ a red-yellow-blue-
- ☐ b blue-red-yellow-
- ☐ c red-
- ☐ d The retrieval order might change with each execution.

ME-Q86. Given

```
import java.util.*;
class Color {
    static int count = 0;
    public int hashCode() {
        ++count;
        return super.hashCode();
    }
}
class Rainbow2 {
    public static void main(String args[]) {
        HashMap<Color, String> map = new HashMap<>();
        Color c1 = new Color(); Color c2 = new Color();
        map.put(c1, "Red");
        map.put(c2, "Yellow");
        map.get(new Color());
        map.get(c1);
        System.out.print(Color.count);
    }
}
```

what is the output? (Choose one option.)

- ☐ a 0
- ☐ b 1
- ☐ c 2
- ☐ d 3
- ☐ e 4
- ☐ f Undefined
- ☐ g Compilation error
- ☐ h RuntimeException

ME-Q87. Assume that thread class1 doesn't own object ObjectA's monitor lock. What happens if it calls wait() or notify() on it? (Choose one option.)

- ☐ a The code won't compile.
- ☐ b The code will throw a RuntimeException.

- ☐ c class1 will try to acquire objectA's monitor lock and then execute wait() or notify().
- ☐ d class1 will execute wait() or notify() without waiting to acquire the lock on objectA's monitor.
- ☐ e None of the above

ME-Q88. Given

```
class Fibre {
    String type = "Fibre";
    String type() {
        return type;
    }
}
class Silk extends Fibre {
    String type = "Silk";
    String type() {
        return type;
    }
}
class Cloth {
    public static void main(String args[]) {
        Fibre f = new Silk();
        System.out.printf("%s : %s", f.type(), f.type);
    }
}
```

what is the output of class Cloth? (Chose one option.)

- ☐ a Fibre : Fibre
- ☐ b Fibre : Silk
- ☐ c Silk : Fibre
- ☐ d Silk : Silk
- ☐ e Compilation error
- ☐ f RuntimeException

ME-Q89. Given

```
class A {
    static int age() {return 10;}
}
interface B {
    int age();
}
class C extends A implements B {
    //INSERT CODE HERE
}
```

which option when inserted at //INSERT CODE HERE will enable class C to compile successfully? (Choose one option.)

- ☐ a `public int age() {return 20;}`
- ☐ b `public static int age() {return 20;}`
- ☐ c `public int age() {return 20;}`
`public static int age() {return 20;}`
- ☐ d None of the above

ME-Q90. Given that classes `Connection` and `SQLConnection` are defined in separate packages

```
package util;
public class Connection {
    protected String url;
    private String port;
    public String pwd;
    String username;
}
package sql;
import util.Connection;
class SQLConnection extends Connection{
    void getDefaultConnection(){
        //line1
    }
}
```

which instance variables of `Connection` can `SQLConnection` access directly (using inheritance) at `//line1`? (Choose the best option.)

- ☐ a `url`, `port`, `pwd`, and `username`
- ☐ b `url`, `pwd`, and `username`
- ☐ c `url` and `pwd`
- ☐ d `pwd`

13.2 Answers to the mock exam



[2.5] Use enumerated types



[6.3] Auto-close resources with a try-with-resources statement

ME-Q1. Given

```
enum Color implements AutoCloseable {           //line1
    PINK, GREEN, RED;
    public void close() {
        System.out.print("closed:");
    }
}
class Palette {
    public static void main(String args[]) {
```

```

        try (Color red = Color.RED) {                //line2
            System.out.print("try:");
        }
        finally {
            System.out.print("finally:");
        }
    }
}

```

which statement is true? (Choose the best option.)

- ☐ a Compilation fails at //line1—enum can't implement AutoCloseable.
- ☐ b Compilation fails at //line2.
- ☒ c **Code compiles and outputs try:closed:finally:.**
- ☐ d None of the above

ME-A1. c

Explanation: An enum implicitly extends `java.lang.Enum` and so it can't extend any other class. But it can implement one or more interfaces. The enum `Color` implements `AutoCloseable` and implements its method `close()` correctly, so it compiles successfully. The try-with-resources statement at //line2 compiles successfully. It declares a variable of type `Color` and assigns a value, `Color.RED`, to it. The try-with-resources statement need not necessarily use the operator `new` to assign a value to a variable. The variables declared using try-with-resources can be assigned constant values or values returned from methods.



[12.4] Format dates, numbers, and currencies for localization with the `NumberFormat` and `DateFormat` classes (including number format patterns)

ME-Q2. Assuming that the default locale is set to `Locale.FRANCE`, which option will format and output the number literal value 9 999? (Choose the best option.)

- ☒ a **`System.out.println(NumberFormat.getInstance(Locale.FRANCE).format(9999));`**
- ☐ b `System.out.println(NumberFormat.getLocalInstance().format(9999));`
- ☐ c `System.out.println(NumberFormat.getDefaultInstance().format(9999));`
- ☐ d `System.out.println(NumberFormat.getDefault().format(9999));`

ME-A2. a

Explanation: The method names `getLocalInstance()`, `getDefaultInstance()`, and `getDefault()` are invalid method names. They aren't defined in class `NumberFormat`. The valid factory methods to retrieve a general-purpose number format for the current default locale and the specified locale are `getInstance()` and `getInstance(Locale inLocale)`.



[6.3] Auto-close resources with a try-with-resources statement



[7.2] Use streams to read from and write to files by using classes in the java.io package including BufferedReader, BufferedWriter, File, FileReader, FileWriter, DataInputStream, DataOutputStream, ObjectOutputStream, ObjectInputStream, and PrintWriter

ME-Q3. Given

```
//insert code here
catch (FileNotFoundException e) {}
catch (IOException e) {}
```

which options, when inserted at //insert code here, will enable the preceding code to compile? (Choose two options.)

- ☐ a try (File file1 = new File("abc.txt");
File file2 = new File("abc.txt");) {}
- ☐ b try (FileInputStream file1 = new File("abc.txt");
FileInputStream file2 = new File("abc.txt");) {}
- ☒ c try (FileInputStream file1 = new FileInputStream(new File("abc.txt"));
FileInputStream file2 = new FileInputStream(new
File("abc.txt"));) {}
- ☐ d try (File file1 = new File("abc.txt");
File file2 = new File("abc.txt")) {}
- ☐ e try (FileInputStream file1 = new File("abc.txt");
FileInputStream file2 = new File("abc.txt")) {}
- ☒ f try (FileInputStream file1 = new FileInputStream(new File("abc.txt"));
FileInputStream file2 = new FileInputStream(new
File("abc.txt"))) {}

ME-A3. c, f

Explanation: This question tests you on

- The reference variables that you can use with the try-with-resources statement—that is, whether instances of File or FileInputStream can be used
- Using the correct constructor to instantiate FileInputStream
- Whether a semicolon can follow the definition of the last resource in the try-with-resources statement

Options (a) and (d) are incorrect because class File doesn't implement AutoCloseable.

Options (b) and (e) are incorrect. Though class FileInputStream implements AutoCloseable, this option doesn't initialize its variables correctly. File instances can't be assigned to FileInputStream variables.

Options (c) and (f) are correct. Class `FileInputStream` implements `AutoCloseable`. A semicolon that marks the end of a code line or code block isn't mandatory at the end of the last resource that's defined using `try-with-resources`. A semicolon can be included or excluded.



[8.2] Check, delete, copy, or move a file or directory with the `Files` class

ME-Q4. Given

```
public int checkFile(Path path) {  
    if (Files.exists(path))  
        return 1;  
    else {  
        if (Files.notExists(path))  
            return 2;  
        else  
            return 3;  
    }  
}
```

for any given `Path` instance, which value can the preceding method return? (Choose the best option.)

- ☐ a Only 1
- ☐ b Only 1 and 2
- ☐ c Only 1 and 3
- ☐ d Only 2
- ☐ e Only 2 and 3
- ☐ f Only 3
- ☒ g 1, 2, or 3

ME-A4. g

Explanation: Method `exists()` returns `true` if the file exists, or `false` if the file doesn't exist or its existence cannot be determined. Method `notExists()` returns `true` if the file does not exist, or `false` if the file exists or its existence cannot be determined.

Both `exists()` and `notExists()` return the same boolean value of `false` for a `Path` object if they can't determine the existence of the target file or directory. So the method in this question's code can return all three values: 1, 2, and 3.



[11.1] Use collections from the `java.util.concurrent` package with a focus on the advantages over and differences from the traditional `java.util` collections

ME-Q5. Which option from the following code would ensure that `getID()` never returns a duplicate value in a multithreaded environment? (Choose the best option.)

```
interface Counter {
    int getID();
}

☒ a class MyCounter implements Counter {
    AtomicInteger ids = new AtomicInteger();
    public int getID() {
        return ids.incrementAndGet();
    }
}

☐ b class MyCounter extends Counter {
    AtomicInteger ids = new AtomicInteger();
    public int getID() {
        return ids.incrementAndGet();
    }
}

☐ c class MyCounter implements Counter {
    synchronized AtomicInteger ids = new AtomicInteger();
    public int getID() {
        return ids.incrementAndGet();
    }
}

☐ d class MyCounter implements Counter {
    int atomic;
    public int getID() {
        synchronized(new StringBuilder()) {
            return ids.incrementAndGet();
        }
    }
}
```

ME-A5. a

Explanation: Option (b) is incorrect. A class implements an interface and can't extend it. Though it might seem to be a trivial point being tested on an advanced topic like concurrency, you might get to answer similar questions on the real exam. A question that seems to be defined as `synchronized` might be testing you on threads or it might be testing you on file I/O and vice versa.

Option (c) is incorrect because variables can't be defined as `synchronized` members. Only methods and code block can be defined as `synchronized` members.

Option (d) is also incorrect because it doesn't compile because `ids` is undefined. The code doesn't serve the purpose even after compilation. Each execution

of `getID()` acquires a lock on a new `StringBuilder` object. This essentially means that `getID()` doesn't block concurrent execution and can be executed concurrently by multiple threads.



[1.2] Override methods

ME-Q6. Given

```
abstract class Dog {
    void bark() {}
}
class Beagle extends Dog {
    //insert code here
}
```

which methods correctly override `bark()` in `Dog`? (Choose three options.)

- ☐ a `static void bark() throws Exception {}`
- ☒ b `void bark() {}`
- ☐ c `static void bark() {}`
- ☐ d `static void bark() throws Error {}`
- ☐ e `abstract static void bark() {}`
- ☒ f `final void bark() {}`
- ☒ g `protected synchronized void bark() {}`

ME-A6. b, f, g

Explanation: Options (a), (c), and (d) are incorrect because a static method can't override an instance method and vice versa.

Option (e) is incorrect due to several reasons:

- The combination of keywords `abstract` and `static` is invalid.
- The declaration of an abstract method is followed by a semicolon and not curly braces.
- An abstract method can't exist in a concrete or nonabstract class.



[11.4] Use the parallel fork/join framework

ME-Q7. Which object of the following classes can't be passed to method `execute()` of `ForkJoinPool`? (Choose four options.)

- ☐ a `RecursiveTask<Integer>`
- ☐ b `RecursiveTask<String>`
- ☐ c `RecursiveAction<StringBuilder>`

- ☐ d RecursiveAction<Boolean>
- ☒ e ForkTask<Long>
- ☒ f ForkTask<Double>
- ☒ g ForkJoinPoolTask<Short>
- ☒ h ForkJoinPoolTask<Byte>

ME-A7. e, f, g, h

Explanation: Classes ForkTask and ForkJoinPoolTask aren't defined in the Java API. Method execute() of class ForkJoinPool accepts objects of ForkJoinTask<T>, an abstract class. Classes RecursiveTask<T> and RecursiveAction extend ForkJoinTask<T>.



[6.1] Use the throw statement and throws clause



[6.4] Create custom exceptions

ME-Q8. Given

```
class ExA extends Exception {}
class ExB extends ExA {}
class ExC extends ExB {}
class App {
    void launch() throws ExB{
        throw new ExC();
    }
    void useLaunch() {
        try {
            launch();
        }
        catch (/*insert code here*/) {}
    }
}
```

which option when inserted at /*insert code here*/ won't enable class App to compile? (Choose the best answer option.)

- ☐ a Exception e
- ☐ b ExA e
- ☐ c ExB e
- ☒ d **ExC e**
- ☐ e None of the above

ME-A8. d

Explanation: First, note that the question asks you to choose an option that doesn't allow App to compile. Even though launch() throws an ExC instance, it declares to

throw `ExB`. Code that calls `launch()` should either handle or declare `ExB` or any of its superclasses to compile successfully. Option (d) handles `ExC`, which is a derived class of `ExB`. So it fails to compile.



[5.1] Search, parse, and build strings (including `Scanner`, `StringTokenizer`, `StringBuilder`, `String`, and `Formatter`)

ME-Q9. Given

```
enum Color {
    RED("yellow"), YELLOW("blue"), BLUE("red");
    String color;
    Color(String c) {this.color = color;}
}
Integer qty = 991177;
```

what is the output of the following code? (Choose the best answer option.)

```
System.out.printf("I want: [%-+5d] bottles of %s color", qty, Color.RED);
```

- ☐ a I want: [991177] bottles of RED color
- ☐ b I want: [991177] bottles of yellow color
- ☐ c I want: [991177] bottles of RED color
- ☐ d I want: [991177] bottles of yellow color
- ☒ e I want: [+991177] bottles of RED color
- ☐ f I want: [+991177] bottles of yellow color
- ☐ g I want: [+99117] bottles of RED color
- ☐ h I want: [+99117] bottles of yellow color

ME-A9. e

Explanation: Here are the flags and their purposes:

- -: Left justifies an argument
- +: Include a sign (+ or -) with this argument
- 5: The minimum width of the displayed argument

In this question, the flag + will include a + sign with the positive literal value 991177.

Because the minimum width specified is 5 (which exceeds the actual width of 6), using 5 doesn't make a difference. The integer value is not truncated to five digits.

Because the actual width of the argument is greater than the specified width, specifying left padding has no affect. It doesn't include additional padding.

`%s` calls `toString()` on an object to get its value. Because enum `Color` hasn't overridden its method `toString()`, calling `toString()` on an enum constant returns the constant's name.



[7.1] Read and write data from the console

ME-Q10. Given that the following code throws a `NullPointerException`, what could be the probable cause? (Choose two options.)

```
Console console = System.console();           //line1
String name = console.readLine("Enter name:"); //line2
String in = console.readLine();               //line3
System.out.println(in.length());             //line4
```

- ☐ a The code is invoked from the command line before redirecting the standard input and output streams.
- ☒ b **Code execution started automatically as a result of execution of some other program.**
- ☒ c **The underlying system doesn't support the Console operations.**
- ☐ d When prompted (as a result of execution of `//line3`), the user didn't enter a value.
- ☐ e When prompted (as a result of execution of `//line3`), the user entered `null`.

ME-A10. b, c

Explanation: If code is invoked from the command line after redirecting the standard input and output streams, you might not be able to access the console. If a user doesn't enter a value, the console returns an object with length 0. There isn't any way to enter `null` at the console. If a user keys the literal value `null`, then it's treated as the `String` value `null` and not the keyword `null`.



[3.2] Choose between interface inheritance and class inheritance

ME-Q11. What is the difference when a class implements the `Runnable` interface versus extends class `Thread` to create a separate thread of execution? (Choose the best option.)

- ☐ a When a class implements `Runnable`, it can only extend `Thread`.
- ☒ b **Classes that implement `Runnable` don't inherit the members of class `Thread`.**
- ☐ c You can't modify the priority of threads of execution created using classes that implement `Runnable`.
- ☐ d By extending class `Thread`, a new thread of execution is created more efficiently than by implementing the `Runnable` interface.

ME-A11. b

Option (a) is incorrect. When a class implements an interface, it can extend any class.

Option (c) is incorrect. A class can implement `Runnable` and modify the priority of the thread of execution using the class's instance. For example

```

class MyThread implements Runnable{
    public void run() {}
}
class Test {
    public void aMethod() {
        Thread t = new Thread(new MyThread());
        t.setPriority(7);
    }
}

```

Option (d) is incorrect. Oracle hasn't documented any efficiency issues related to creating threads of execution using instances of classes that extend `Thread` and the ones that implement `Runnable`.



[4.6] Create and use Map implementations

ME-Q12. What is the iteration order of values inserted in a `HashMap`? (Choose the best option.)

- ☐ a Insertion order
- ☐ b Natural order of keys
- ☐ c Natural order of values
- ☒ d **Undefined**

ME-A12. d

Explanation: Following is the iteration order of `Collection` classes:

- `HashSet`—undefined
- `HashMap`—undefined
- `LinkedHashSet`—insertion order
- `LinkedHashMap`—insertion order of keys (by default), or access order
- `ArrayList`—insertion order
- `LinkedList`—insertion order
- `TreeSet`—their natural order (as per `Comparable`) or by a `Comparator`
- `TreeMap`—their natural order (as per `Comparable`) or by a `Comparator`



[9.2] Identify the components required to connect to a database using class `DriverManager` (including the JDBC URL)

ME-Q13. Assuming that the exceptions are handled appropriately, which option is the recommended way to load a database driver and establish a connection with the database using JDBC version 4.0? (Choose the best option.)

- ☐ a `java.sql.DriverManager.registerDriver("com.mysql.jdbc.driver");`
`Class.forName("com.mysql.jdbc.driver");`
`DriverManager.getConnection("jdbc:mysql://data.ejavaguru.com:3305/myDB");`
- ☐ b `Class.forName("com.mysql.jdbc.driver");`
`DriverManager.getConnection("jdbc:mysql://data.ejavaguru.com:3305/myDB");`
- ☒ c `DriverManager.getConnection("jdbc:mysql://data.ejavaguru.com:3305/myDB");`
- ☐ d `DriverManager.getConnection("jdbc:mysql:", "guest", "guest");`

ME-A13. c

Explanation: JDBC 4.0 and its later versions support automatic loading and registration of all JDBC drivers accessible via an application's classpath. For all earlier versions of JDBC, you must manually load the JDBC drivers using `Class.forName()`, before you can access the driver. So options (a) and (b) are incorrect because they aren't the recommended ways to load and establish a JDBC connection.

Option (d) is incorrect because it doesn't include the database name. Here's the format of a JDBC URL: `jdbc:<dbtype>://<host>:<port>/<database_name>`.



[3.1] Write code that declares, implements, and/or extends interfaces



[3.2] Choose between interface inheritance and class inheritance

ME-Q14. Given

- X defines methods.
- Y extends X.
- Y can overload methods defined in X.
- X can't define static methods.

Which of the following statements is true? (Choose two options.)

- ☐ a X is a class.
- ☐ b Y is a class.
- ☒ c X is an interface.
- ☒ d Y is an interface.

ME-A14. c, d

Explanation: An interface can't define static methods but a class can define static methods. Because X can't define static methods, it can't be a class. Because Y extends X, Y is also an interface. An interface can only be extended by another interface.



[8.5] Find a file with the PathMatcher interface

ME-Q15. What is the output of the following code? (Choose the best option.)

```
String[] fileNames = {"data-eng-temp.txt",
                      "dataA_sst.temp.xml",
                      "data-scc-perm.doc",
                      "Data-pra-sscu.txt",
                      "data-php-data.php" };
PathMatcher matcher = FileSystems.getDefault()
                      .getPathMatcher("glob:*a?a-*x*");

int result = 0;
for (int i = 0; i < fileNames.length; i++)
    if (matcher.matches(Paths.get(fileNames[i])))
        result = result + 2;
System.out.println(result);
```

- ☐ a 2
- ☒ b 4
- ☐ c 6
- ☐ d 8
- ☐ e 10

ME-A15. b

Explanation: Similar to a regex pattern, a glob pattern is specified as a string and is matched against other strings (such as directory names or filenames). The glob pattern in this code matches the following filenames incrementing result twice, printing 4:

- data-eng-temp.txt
- Data-pra-sscu.txt

Here's a breakdown of the glob pattern used in this code, to help you understand the filenames that it matches:

- An asterisk (*) matches zero or multiple characters.
- A question mark (?) matches exactly one character.
- All other characters match themselves.

So the pattern `*a?a-*x*` matches filenames that include a followed by exactly one character and a-, which is followed by the letter x (at any position).

The value `dataA_sst.temp.xml` doesn't match the glob pattern because it doesn't include an underscore. The values `data-scc-perm.doc` and `data-php-data.php` don't include x. This glob pattern will also match the following values (matching characters in bold):

- **data**-eng-temp.**tx**
- **data**-eng-temp.**xt**

- `data-xeng-temp.doc`
- `data-eng-temxp.doc`



[2.3] Use the static and final keywords

ME-Q16. Given

```
class Color {
    final int shade;
    //insert code here
}
```

which options when inserted independently at `//insert code here` will enable class `Color` to compile successfully? (Choose two options.)

- ☒ **a** `Color(int shade) {`
 `this.shade = shade;`
 `}`
- ☐ **b** `static {`
 `shade = 10;`
 `}`
- ☒ **c** `{`
 `shade = 10;`
 `}`
- ☐ **d** `Color () {`
 `shade = 10;`
 `}`
 `Color(int shade) {`
 `this();`
 `this.shade = 20;`
 `}`

ME-A16. a, c

Explanation: Class `Color` defines a final instance variable, `shade`, which must be initialized for each instance, exactly once. A final instance variable must be initialized in one of the following ways:

- With the variable declaration
- In a constructor
- In an instance initializer block

If the compiler determines that a final instance variable isn't initialized using any of the preceding ways or assigned a value more than once, the code won't compile.

Option (b) is incorrect because a static initializer block can't access instance variables.

Option (d) is incorrect because it assigns a value to the final variable `shade` twice, which isn't allowed.



[9.3] Submit queries and read results from the database (including creating statements, returning result sets, iterating through the results, and properly closing result sets, statements, and connections)

ME-Q17. Given

```
Item PEN
ID, INTEGER: PK
MODEL, VARCHAR(50)
LENGTH, INTEGER
WEIGHT, REAL
```

assuming that table PEN contains two rows, what is the output of the following code? (Choose the best option.)

```
try (
    Connection c = DriverManager.getConnection
        ("jdbc:mysql://localhost/abc", "a", "b");
    String query = "SELECT * FROM pen WHERE length>99";
    Statement stmt = c.createStatement();
    ResultSet rs = stmt.executeQuery(query)) {
    while (rs.next ()) {
        System.out.println(rs.getString("MODEL"));
    }
}
catch (Exception se) {
    System.out.println("Exception");
}
```

- ☐ a The code outputs a value for column model twice, if its corresponding length is greater than 99.
- ☐ b The code outputs a value for column model once, irrespective of the value of column length.
- ☐ c The code outputs an Exception.
- ☒ d **The code fails to compile.**

ME-A17. d

Explanation: The code fails to compile. The resources declared in the try-with-resources statement should either implement `AutoCloseable` or any of its subinterfaces, and `String` doesn't.



[7.1] Read and write data from the console

ME-Q18. Which statements are correct? (Choose two options.)

- ☒ a **Class `Console` defines methods to access the character-based console device associated with the current JVM.**

- ☐ b `System.console()` always returns a non-null value.
- ☐ c `System.getConsole()` might not always return a null value.
- ☒ d **Console can be used to write formatted data.**
- ☐ e Console's method `readPwd()` reads a password or pass-phrase from the console with echoing disabled.
- ☐ f Console's method `readUser()` reads a `String` value from the console without echoing disabled.

ME-A18. a, d

Explanation: Option (b) is incorrect. `System.console()` might not always return a non-null value. If you invoke JVM from the command line without redirecting the standard input and output streams, then you'll be able to access its console, which will typically be connected to the keyboard and display from which the virtual machine was launched. You may not be able to access the console associated with a JVM, if it started automatically, as a result of the execution of some other program.

Option (c) is incorrect. Class `System` doesn't define method `getConsole()`.

Option (d) is correct. `Console` defines methods `printf()` and `format()`, which can be used to write formatted data.

Options (e) and (f) are incorrect because they use invalid method names.



[9.5] Construct and use `RowSet` objects using the `RowSetProvider` class and the `RowSetFactory` interface

ME-Q19. Which statement is incorrect? (Choose the best answer.)

- ☐ a A `CachedRowSet` is scrollable, updatable, and serializable.
- ☐ b A `JdbcRowSet` is always connected to its database.
- ☐ c A `FilteredRowSet` object applies a criterion on all rows in a `RowSet` object to manage a subset of rows in a `RowSet` object.
- ☐ d Any number of `RowSet` objects can be added to an instance of `JoinRowSet` if they can be related in a SQL JOIN.
- ☐ e An application can modify the data in a `CachedRowSet` object, and those modifications can then be propagated back to the source of the data.
- ☒ f **A `CachedRowSet` object is a connected rowset.**

ME-A19. f

Explanation: Options (a), (b), (c), (d), and (e) are correct statements. Only option (f) is an incorrect statement. A connected row set is a row set that's *always* connected to its database. A `CachedRowSet` caches its rows in memory (and isn't always connected to its database).



[7.2] Use streams to read from and write to files by using classes in the java.io package including BufferedReader, BufferedWriter, File, FileReader, FileWriter, DataInputStream, DataOutputStream, ObjectOutputStream, ObjectInputStream, and PrintWriter

ME-Q20. Given that contents of in.txt are

day

what is the output of the following code? (Choose the best option.)

```
import java.io.*;
class ReadWrite {
    public static void main(String args[]) throws Exception {
        BufferedReader in = new BufferedReader(new FileReader("in.txt"));
        BufferedWriter out = new BufferedWriter(new FileWriter("out.txt"));
        int i = in.read();
        int ctr = 0;
        while (i != -1) {
            out.write(++ctr + "." + (char)i);
            i = in.read();
        }
        out.flush();
        out.close();
    }
}
```

- ☐ a File out.txt contains 1.d.2.a.3.y.
- ☐ b File out.txt contains
 - 1.d
 - 2.a
 - 3.y
- ☒ c File out.txt contains 1.d2.a3.y.
- ☐ d Code fails to compile
- ☐ e None of the above

ME-A20. c

Explanation: Method read() returns a non-negative integer value for values that it reads and -1 when it reaches the end of the file. The code appends ++ctr and a dot (.) to each value that's read from in.txt and writes it to out.txt.



[3.3] Apply cohesion, low-coupling, IS-A, and HAS-A principles

ME-Q21. Given

- Fwimmer is an interface.
- Swammer extends Fwimmer.

- JamG defines a variable of type Boolean.
- Jammer extends JamG.
- Yammer implements Swammer.
- Jammer implements Fwimmer.

Which statements are correct? (Choose two options.)

- ☐ a Fwimmer IS-A Swammer.
- ☐ b Yammer HAS-A Fwimmer.
- ☒ c **Jammer HAS-A Boolean.**
- ☐ d JamG IS-A Swammer.
- ☐ e Jammer IS-A Swammer.
- ☒ f **Jammer IS-A Fwimmer.**

ME-A21. c, f

Explanation: Though the class and interface names in this question seem to be ridiculous, you might see use of similar names on the real exam.

Option (a) is incorrect. Fwimmer doesn't extend Swammer.

Option (b) is incorrect. Yammer implements Swammer, which extends Fwimmer.

Option (c) is correct. Jammer extends JamG and JamG defines a variable of type Boolean.

Option (d) is incorrect. JamG isn't connected to Swammer.

Option (e) is incorrect. Jammer implements Fwimmer, not Swammer. Because Swammer extends Fwimmer, Jammer IS-A Swammer is incorrect.

Option (f) is correct. Because Jammer implements Fwimmer, Jammer IS-A Fwimmer is correct.



[6.1] Use the throw statement and throws clause

ME-Q22. Given

```
class Lake {
    void waterSkiing() throws RuntimeException {
        System.out.print("Skiing:");
    }
}
class Travel {
    public static void main(String args[]) {
        Lake lake = new Lake();
        try {
            lake.waterSkiing();
        }
        catch (RuntimeException e) {
            System.out.print("RuntimeEx:");
        }
    }
}
```

```

        catch (Exception e) {
            System.out.print("Ex:");
        }
    }
}

```

what is the output? (Choose the best option.)

- ☒ a **Skiing:**
- ☐ b Skiing:RuntimeEx:
- ☐ c Skiing:Ex:
- ☐ d Compilation error

ME-A22. a

Explanation: Though method `waterskiing()` declares to throw a `RuntimeException`, it doesn't throw it. A method can declare to throw a checked or unchecked exception, even though it doesn't actually throw it. Because `waterskiing()` doesn't throw any exception, the control isn't passed to any of the exception handlers.



[8.1] Operate on file and directory paths with the Path class

ME-Q23. Given

```

Path path1 = new File("/home/oracle").toPath();
Path path2 = FileSystems.getDefault().getPath("/home/java/../hello.txt");
Path path3 = /* insert code here */
System.out.println(path3);

```

when inserted at `/* insert code here */`, which option will output `/home/hello.txt`? (Choose the best option.)

- ☐ a `path1.resolve(path2);`
- ☐ b `path2.normalize(path1);`
- ☐ c `path1.relativize(path2);`
- ☐ d `path2.relativize(path1);`
- ☒ e `path2.normalize();`

ME-A23. e

Explanation: Method `normalize()` returns a path that's this path with redundant name elements eliminated. `../` refers to a parent directory, so `path2.normalize()` removes `/java/..` from the path `/home/java/../hello.txt`.

Option (a) outputs `\home\java\..\hello.txt`.

Option (b) won't compile. Method `normalize()` doesn't accept any method parameter.

Option (c) outputs ..\java\..\hello.txt.

Option (d) outputs ..\..\..\oracle.



[6.1] Use the throw statement and throws clause

ME-Q24. Given

```
class MyException extends Exception{}
```

which method will compile successfully? (Choose the best option.)

- ☐ a void methodA() throws Exception {}
- ☐ b void methodB() throws RuntimeException {}
- ☐ c void methodC() throws FileNotFoundException {}
- ☐ d void methodD() throws MyException {}
- ☒ e All of the above
- ☐ f None of the above

ME-A24. e

Explanation: A method can declare to throw any checked or unchecked exception, even if it doesn't actually throw it.



[5.1] Search, parse, and build strings (including Scanner, StringTokenizer, StringBuilder, String, and Formatter)



[5.2] Search, parse, and replace strings by using regular expressions, using expression patterns for matching limited to . (dot), * (star), + (plus), ?, \d, \D, \s, \S, \w, \W, \b, \B, [], ()

ME-Q25. What is the output of the following code? (Choose the best option.)

```
public class TestSplit {
    public static void main(String[] args) {
        String names = "Shreya;-Selvan;-Paul;-Harry";
        String[] results = names.split(";..");
        for(String str:results) System.out.print(str);
    }
}
```

- ☒ a ShreyaSelvanPaulHarry
- ☐ b Shreya-Selvan-Paul-Harry
- ☐ c Shreya.-Selvan.-Paul.-Harry
- ☐ d Shreya;-Selvan;-Paul;-Harry

ME-A25. a

Explanation: In the pattern `;. .` the semicolon (`;`) matches itself and the dot (`.`) matches exactly one (any) character. So method `split()` splits the string `Shreya; .-Selvan; .-Paul; .-Harry` into tokens `Shreya`, `Selvan`, `Paul`, and `Harry`.

For a quick example, if you modify the target string value to `Shreya;Selvan;Paul;Harry`, the same regex pattern (`;. .`) will return the following tokens (the first two characters after the semicolon are included in the split pattern):

- `Shreya`
- `lvan`
- `ul`
- `rry`


[8.2] Check, delete, copy, or move a file or directory with the Files class

ME-Q26. Which option when inserted at `//insert code here` copies a source file to the destination, replacing any existing file with the same name? (Choose the best option.)

```
void move(Path src, Path dest) throws Exception {
//insert code here
}
```

- ☐ a `Files.copy(src, dest);`
`Files.delete(src);`
- ☐ b `Files.copy(src, dest, true);`
`Files.deleteIfExists(src);`
- ☒ c `Files.copy(src, dest, StandardCopyOption.REPLACE_EXISTING);`
`Files.deleteIfExists(src);`
- ☐ d `Files.replace(src, dest);`
`Files.delete(src);`

ME-A26. c

Explanation: Option (a) will throw a `RuntimeException` if the destination directory already includes a file with the same name.

Option (b) won't compile because the overloaded method `copy()` accepts a `varargs` parameter or type `CopyOption`.

Option (c) is correct. By using the option `StandardCopyOption.REPLACE_EXISTING`, method `Files.copy()` will replace an existing file with the same name.

Option (d) is incorrect. It uses the invalid method name `Files.replace()`.



[7.2] Use streams to read from and write to files by using classes in the java.io package including BufferedReader, BufferedWriter, File, FileReader, FileWriter, DataInputStream, DataOutputStream, ObjectOutputStream, ObjectInputStream, and PrintWriter

ME-Q27. Given that contents of in.txt are

day

what is the output of the following code? (Choose the best option.)

```
import java.io.*;
class ReadWrite {
    public static void main(String args[]) throws Exception {
        BufferedReader in = new BufferedReader(new FileReader("in.txt"));
        BufferedWriter out = new BufferedWriter(new FileWriter("out.txt"));
        int i = in.read();
        int ctr = 0;
        while (i != -1) {
            out.write(++ctr + (char)i);
            i = in.read();
        }
        out.flush();
        out.close();
    }
}
```

- ☐ a File out.txt contains 1.d.2.a.3.y.
- ☐ b File out.txt contains
 - 1.d
 - 2.a
 - 3.y
- ☐ c File out.txt contains 1.d2.a3.y.
- ☐ d Code fails to compile
- ☒ e None of the above

ME-A27. e

Explanation: Revisit this line of code:

```
out.write(++ctr + (char)i);
```

The preceding line of code adds ++ctr to the value i, before writing it to the output file. So the code doesn't write the value that it reads from the input file. It doesn't copy contents of in.txt to out.txt.

The first execution of in.read() reads value (d) from in.txt and assigns 100 to i (the integer value of char d is 100). The value of ctr is incremented by 1 before its value is added to i. So char e (integer value 101) is written to out.txt. Similarly, the

code reads characters from in.txt, but modifies their integer values, before writing them to out.txt.



[9.6] Create and use PreparedStatement and CallableStatement objects

ME-Q28. Assuming that `c` is a valid `Connection` object, which of the following can be used to execute the stored procedure `updateExamDetails` and also retrieve the updated data from the database? (Choose the best option.)

- ☐ a `c.prepareStatement("{call updateExamDetails()}").executeUpdate();`
- ☐ b `c.prepareStatement(" {call updateExamDetails()}").execute();`
- ☒ c `c.prepareStatement("{call updateExamDetails()}").executeQuery();`
- ☐ d `c.callableStatement(" {call updateExamDetails()}");`
- ☐ e `c.callProcedure(" {call updateExamDetails()}");`

ME-A28. c

Explanation: Option (c) is the best option because `executeQuery()` returns a `ResultSet`, and the question requires you to execute the stored procedure `updateExamDetails` and retrieve updated data from the database. Options (a) and (b) are incorrect because `executeUpdate()` returns an `int` and `execute()` returns a `boolean`. Options (d) and (e) are incorrect because they use nonexistent method names, `callableStatement()` and `callProcedure()`.



[1.4] Use the instanceof operator and casting

ME-Q29. Given

```
class Device {}
class Phone extends Device {
    void model() {}
}
class MobilePhone extends Phone {}
```

which options can be used to access method `model()` using a reference variable, `d`, defined as follows? (Choose two options.)

```
Device d = new MobilePhone();
```

- ☐ a `d.model();`
- ☒ b `((Phone)d).model();`
- ☒ c `((MobilePhone)d).model();`
- ☐ d `((MobilePhone.Phone)d).model();`

ME-A29. b, c

Explanation: Option (a) is incorrect because it won't compile. Because `model()` is defined in `Device`'s derived class (`Phone`), a reference variable of type `Device` can't access `model()`, even if it refers to an instance of a subclass that defines `model()`.

Option (b) is correct. By explicitly casting `d` to `Phone`, `((Phone)d)` can access `model()`.

Option (c) is correct. Method `model()` is defined in `Phone` and is inherited by class `MobilePhone`. By casting `d` to `MobilePhone`, `((MobilePhone)d)` can access `model()`.

Option (d) is incorrect. This option won't compile because it uses an invalid syntax to cast `d`.



[5.1] Search, parse, and build strings (including `Scanner`, `StringTokenizer`, `StringBuilder`, `String`, and `Formatter`)

ME-Q30. Which statements for the given code are correct? (Choose two options.)

```
Integer i1 = 20;
String s1 = "Shreya";
boolean b1 = false;
char c1 = 109;
```

- ☒ **a** `System.out.printf("%s %b %c", i1, s1, c1, b1);` will output values without any exceptions.
- ☐ **b** `System.out.printf("%s %b %c", i1, s1);` won't compile.
- ☒ **c** `System.out.printf("%s %i %c", i1, s1);` will throw a runtime exception.
- ☐ **d** `System.out.printf("%sd", i1);` will throw a runtime exception.

ME-A30. a, c

Explanation: Option (a) is correct. If the count of arguments exceeds the count of the format specifiers, the code compiles and executes successfully—the extra arguments are simply ignored.

Option (b) is incorrect. If the count of arguments is less than the count of the format specifiers, the code compiles but JVM throws a `MissingFormatArgumentException` at runtime.

Option (c) is correct. Code that uses invalid format specifier compiles but throws an `UnknownFormatConversionException` at runtime.

Option (d) is incorrect. This code will execute and successfully print `20d`.



[6.2] Use the try statement with multi-catch and finally clauses

ME-Q31. Given that code snippet at `//line1` can throw a `NullPointerException`, `FileNotFoundException`, and `SQLException`, which of the following are appropriate catch handles? (Choose two options.)

```
try {
    //line1
}
```

- ☐ a `catch (SQLException e | FileNotFoundException ex) {}`
- ☐ b `catch (SQLException e) {}`
- ☒ c `catch (SQLException e) {}`
 `catch (NullPointerException | FileNotFoundException e) {}`
- ☒ d `catch (FileNotFoundException | SQLException e) {}`

ME-A31. c, d

Option (a) is incorrect. The names of all the exceptions in a multicatch handler aren't followed by the exception variable. The exception variable occurs only once, at the end of the names of the exceptions.

Option (b) is incorrect. The code won't compile because it only handles the checked `SQLException`, it doesn't handle the checked `FileNotFoundException`.

Option (c) is correct. Though the compiler doesn't need you to define a catch handler for code that throws a `NullPointerException` (a `RuntimeException`), it permits you to do so.

Option (d) is correct. Though allowed, you need not catch a `RuntimeException` (`NullPointerException`) to enable your code to compile. This option defines a handler for both the checked exceptions that can be thrown by code at `//line1`, `SQLException` and `FileNotFoundException`.



[8.4] Recursively access a directory tree using the `DirectoryStream` and `FileVisitor` interfaces

ME-Q32. Assuming all directories exist, what is the result of the following code? (Choose the best option.)

```
class MyFileVisitor extends SimpleFileVisitor<Path> {
    String backupDir = "e:\\code\\backup\\";
    public FileVisitResult visitFile
        (Path file, BasicFileAttributes attrs) {
        if (file.toString().endsWith(".java")) {
            try {
                String fileName = file.getFileName().toString();
                Files.copy(file, Paths.get(backupDir + fileName),
                    StandardCopyOption.REPLACE_EXISTING);
            }
        }
    }
}
```

```

        } catch (Exception e) {}
    }
    return FileVisitResult.CONTINUE;
}
public static void main(String args[]) throws Exception {
    Files.walkFileTree(Paths.get("e:/code"), new MyFileVisitor());
}
}

```

- ☐ a It copies all .java files from e:/code to e:/code/backup.
- ☒ b **It copies all .java files recursively from e:/code and its subdirectories to e:/code/backup.**
- ☐ c It copies all non .java files from e:/code to e:/code/backup.
- ☐ d It throws a runtime exception for duplicate .java files found in e:/code or any of its subdirectories.
- ☐ e It modifies the type of directory from e:/code/backup.
- ☐ f It fails to compile.

ME-A32. b

Explanation: The following code starts recursive traversal of directory e:/code, as per the rules defined in class MyFileVisitor:

```
Files.walkFileTree(Paths.get("e:/code"), new MyFileVisitor());
```

Method visitFile() checks if the name of the visited file ends with .java and copies it to another directory. Because class MyFileVisitor extends SimpleFileVisitor, it need not implement all the methods of the FileVisitor interface.



[5.2] Search, parse, and replace strings by using regular expressions, using expression patterns for matching limited to . (dot), * (star), + (plus), ?, \d, \D, \s, \S, \w, \W, \b, \B, [], ()

ME-Q33. What is the output of the following code? (Choose the best option.)

```

Pattern p = Pattern.compile("\\Sthe\\S");
Matcher m = p.matcher("These are their,leather,,,them");
System.out.println(m.replaceAll("<>"));

```

- ☒ a **These are their,le<>,,<>**
- ☐ b These are <>ir,leather,,,them
- ☐ c These are <>ir,leather,<>m
- ☐ d These are <>ir,lea<>r<>m
- ☐ e These are their,lea<>r,<>m

ME-A33. a

Explanation: \s is a whitespace character: space, \t (tab), \n (new line), \x0B (end of line), \f (form feed), \r (carriage). \S is a nonwhitespace character: [^\s]. The pattern \Sthe\S will match any occurrence of “the” that doesn’t have a whitespace character in front or at the end, including the character in front and at the end. So \\Sthe\\S will match “ather” and “,them”.



[3.1] Write code that declares, implements, and/or extends interfaces



[6.1] Use the throw statement and throws clause



[6.4] Create custom exceptions

ME-Q34. Given

```
class Wave extends Exception {}
class Oar extends Exception {}
interface Sail{
    void surf() throws Wave, Oar;
}
interface SuperSail {
    void surf() throws Wave;
}
class Surfer implements SuperSail {
    //INSERT CODE HERE
}
```

which option when inserted independently at //INSERT CODE HERE will enable class Surfer to compile successfully? (Choose two options.)

- ☒ a public void surf() throws Wave {}
- ☐ b public void surf() throws Wave, Oar {}
- ☐ c public void surf() throws Wave {}
 - public void surf() throws Wave, Oar {}
- ☒ d public void surf() {}
- ☐ e public void surf() throws Exception {}

ME-A34. a, d

Explanation: The SuperSail interface doesn’t extend the Sail interface and overrides method surf() declaring to throw only the exception Wave. The concrete class Surfer implements the SuperSail interface and must implement its following method:

```
void surf() throws Wave;
```



```

        con.setAutoCommit(false);
        st = con.createStatement();
        st.executeUpdate("INSERT INTO book values(1, 'Java')");
        Savepoint sv1 = con.setSavepoint("sv1");
        st.executeUpdate("INSERT INTO book values(3, 'PHP')");
        con.rollback();
        st.executeUpdate("INSERT INTO book values(4, 'Android')");
        con.setAutoCommit(true);
        st.executeUpdate("INSERT INTO book values(2, 'Oracle')");

        ResultSet rs = st.executeQuery("SELECT * from BOOK");
        int ctr = 0;
        while (rs.next()) ctr++;
        System.out.println(ctr);
    }
    catch (Exception e) {System.out.println("Exception");}
}

```

- ☐ a 1
- ☒ b 2
- ☐ c 3
- ☐ d 4
- ☐ e The code fails to compile.

ME-A36. b

Explanation: By default, new connections are in auto-commit mode, but the auto-commit mode of the connection instance is initially set to false (in this question code). So no insertions will be committed, unless `commit()` is explicitly called on them. Though the code defines a Savepoint, it isn't passed as a parameter while calling `rollback()`, so *all* the transactions are rolled back. When Connection's auto-commit mode is changed during a transaction, the transaction is committed. So when auto-commit is enabled, the current transaction is committed and the data for the Android book is inserted. From now on all SQL statements will be executed and committed as individual transactions. That's why the Oracle book is inserted as well.



[4.3] Analyze the interoperability of collections that use raw types and generic types

ME-Q37. Which of the following are valid assignments? (Choose five options.)

- ☐ a `List<Number> list = new ArrayList<Integer>();`
- ☒ b `List<?> list = new ArrayList<Integer>();`
- ☒ c `List<? extends Number> list = new ArrayList<Integer>();`
- ☒ d `List<? super Number> list = new ArrayList<>();`
- ☐ e `ArrayList<Number> list = new ArrayList<Integer>();`

- ☒ f `ArrayList<?> list = new ArrayList<Integer>();`
- ☒ g `ArrayList<? extends Number> list = new ArrayList<Integer>();`
- ☐ h `ArrayList<? implements Number> list = new ArrayList<>();`

ME-A37. b, c, d, f, g

Explanation: For any type T, `List<T>` is a subtype of `List<?>`.

Option (a) is incorrect. `ArrayList<Integer>` implements `List<Integer>` and not `List<Number>`.

Option (b) is correct. The wildcard `?` represents an unknown type, and `?` can refer to `Integer`. Because `ArrayList<Integer>` implements `List<Integer>`, this assignment is valid.

Option (c) is correct. For `<? extends Number>`, `Integer extends Number` is valid.

Option (d) is correct. For `<? super Number>`, using type inference—that is, new `ArrayList<>()`—is valid.

Option (e) is incorrect. `ArrayList<Integer>` isn't a subtype of `ArrayList<Number>` even though `Integer` is a subtype of `Number`.

Option (h) fails to compile.



[8.3] Read and change file and directory attributes, focusing on the `BasicFileAttributes`, `DosFileAttributes`, and `PosixFileAttributes` interfaces

ME-Q38. You have a file that stores your top secrets. How can you modify its attributes so that it becomes read-only and hidden? Assume you're using a DOS-based filesystem. (Choose the best option.)

```
void modifyAttr(Path file) {
    // insert code here
}
```

- ☐ a `Files.setAttribute(file, "dos:hidden", true);`
`Files.setAttribute(file, "dos:readonly", true);`
- ☒ b `try {`
 `Files.setAttribute(file, "dos:hidden", true);`
 `Files.setAttribute(file, "dos:readonly", true);`
 `}`
 `catch (IOException e) {}`
- ☐ c `try {`
 `Map<String, Object> values =`
 `Files.readAttributes(file, "dos:readonly, hidden");`
 `values.put("readonly", new Boolean(true));`
 `values.put("hidden", new Boolean(true));`
 `Files.setAttributes(file, values);`
 `}`
 `catch (IOException e) {}`

```

☐ d try {
    Map<String, Object> values = Files.readAttributes(file, "dos:*");
    values.put("readonly", new Boolean(true));
    values.put("hidden", new Boolean(true));
    Files.setAttributes(file, values);
}

```

ME-A38. b

Explanation: Option (a) won't compile because it doesn't handle the `IOException` thrown by `Files.setAttribute()`.

Options (c) and (d) won't compile. Method `setAttributes()` in `Files` doesn't exist.



[4.4] Use wrapper classes, autoboxing, and unboxing



[4.8] Sort and search arrays and lists

ME-Q39. What is the output of the following code? (Choose the best option.)

```

List<Integer> list = new ArrayList<>();
list.add(new Integer(30));
list.add(10);
Arrays.sort(list);
for (Integer i : list)
    System.out.println(i);

```

- ☐ a 10
30
- ☐ b 30
10
- ☒ c **Compilation error**
- ☐ d `RuntimeException`

ME-A39. c

Explanation: The overloaded method `sort()` in class `Arrays` doesn't accept `List` instances. It only accepts arrays.



[6.3] Auto-close resources with a try-with-resources statement

ME-Q40. To be eligible to be declared using a try-with-resources statement, which interfaces should a class implement? (Choose two options.)

- ☐ a `java.io.Resource`
- ☒ b `java.io.Closeable`

- ☐ c java.lang.Resource
- ☒ d java.lang.AutoCloseable
- ☐ e java.lang.Closeable
- ☐ f java.io.AutoCloseable

ME-A40. b, d

Explanation: To be eligible to be declared using a try-with-resources statement, a class must implement `java.lang.AutoCloseable` or any of its derived interfaces. Because the `java.io.Closeable` interface extends `java.lang.AutoCloseable`, option (b) is also correct.



[4.5] Create and use List, Set, and Deque implementations

ME-Q41. What is the output of the following code? (Choose the best option.)

```
import java.util.*;
class TestTreeSet {
    public static void main(String args[]) {
        TreeSet<StringBuilder> treeSetNames = new TreeSet<StringBuilder>();
        treeSetNames.add(new StringBuilder("Shreya"));
        treeSetNames.add(new StringBuilder("Harry"));
        treeSetNames.add(new StringBuilder("Paul"));
        treeSetNames.add(new StringBuilder("Shreya"));
        Iterator it = treeSetNames.descendingIterator();
        while (it.hasNext())
            System.out.print(it.next() + ":");
    }
}
```

- ☐ a Shreya:Harry:Paul:Shreya:
- ☐ b Shreya:Harry:Paul:
- ☐ c Shreya:Shreya:Paul:Harry:
- ☐ d Harry:Paul:Shreya:
- ☐ e Compilation error
- ☒ f **RuntimeException**

ME-A41. f

Explanation: `TreeSet` stores ordered objects and it reorders itself as new objects are added to it. `TreeSet` elements are ordered or sorted using either their natural order (by implementing `Comparable`) or by using the `Comparator` instance passed to the `TreeSet` constructor. In the absence of both of these, `TreeSet` wouldn't be able to compare and add any object to itself, therefore throwing a `ClassCastException` at runtime.



[6.5] Test invariants by using assertions

ME-Q42. Which option is true for the given code? (Choose the best option.)

```
assert(result):new RuntimeException();
```

- ☐ a If result is false, the code will throw a RuntimeException.
- ☐ b If result is true, the code will throw a RuntimeException.
- ☒ c **If result is false, the code will throw an AssertionError.**
- ☐ d If result is true, the code will throw an AssertionError.

ME-A42. c

Explanation: If the expression in the assert statement evaluates to false, it throws an AssertionError. In the long form of the assert statement, the object is passed to the AssertionError's constructor. So the given code will throw an AssertionError with the following message:

```
Exception in thread "main" java.lang.AssertionError:
java.lang.RuntimeException
    at <TestAssert.main(TestAssert.java:4)
        <The name of the Java source file>
Caused by: java.lang.RuntimeException
```



[12.5] Describe the advantages of localizing an application

ME-Q43. Given that a company's application, Effective Customer Relation Management (ECRM) is localized, which option is correct? (Choose the best option.)

- ☐ a ECRM is efficient.
- ☐ b ECRM is highly cohesive and loosely coupled.
- ☒ c **ECRM displays text messages according to a user language and region.**
- ☐ d ECRM always loads localized text messages from a .properties file.
- ☐ e ECRM defines local methods.

ME-A43. c

Explanation: Options (a), (b), and (e) are incorrect. A localized application might not be efficient, highly cohesive, loosely coupled, or define local methods.

Option (d) is incorrect because a localized application might not always use a .properties file to load localized messages. The localized text messages can also be read from a Java class.



[10.4] Identify code that may not execute correctly in a multithreaded environment

ME-Q44. Two threads (one and two in class EJavaGuru) can't reach completion. What is the probable cause? (Choose the best option.)

```
enum Location {COLLEGE, HOME};
class Student {
    Location loc;
    boolean friendFound = false;
    Student(Location loc) {
        this.loc = loc;
    }
    void changeLoc() {
        if (this.loc.equals(Location.COLLEGE))
            this.loc = Location.HOME;
        else
            this.loc = Location.COLLEGE;
    }
    boolean find(Student friend) {
        return (friendFound = this.loc.equals(friend.loc));
    }
}
class Find extends Thread{
    Student s, friend;
    Find (Student s, Student friend) {
        this.s = s;
        this.friend = friend;
    }
    public void run() {
        while (!s.friendFound) {
            s.friendFound = s.find(friend);
            if (!s.friendFound) s.changeLoc();
            try {Thread.sleep(1000);} catch (InterruptedException e) {}
        }
    }
}
class EJavaGuru2 {
    public static void main(String args[]) {
        Student s1 = new Student(Location.HOME);
        Student s2 = new Student(Location.COLLEGE);
        Thread one = new Find(s1, s2);
        Thread two = new Find(s2, s1);
        one.start();
        two.start();
    }
}
```

- ☐ a Deadlock
- ☒ b Livelock
- ☐ c Starvation
- ☐ d Mutual lock

ME-A44. b

Explanation: A thread completes its execution when its method `run()` completes. Threads in a livelock aren't blocked; rather, they're responding to each other, but they aren't able to move to completion.

Compare the example code in this question to two friends who are trying to find each other at home or college. At an instance, the first friend (`s1`) checks whether her friend is at the college. When she can't find her, she proceeds to her house to find her. But at the same time, the other friend (`s2`) looks for her at the college and then home. Because the friends don't seem to communicate with each other, they're in a livelock—each one is changing her position and trying to find the other, in vain.

In this question code, threads one and two aren't able to complete execution of their `run()` method. Each thread is assigned the same set of `Student` instances, `s1` and `s2`. Both threads one and two compare the location of `s1` with `s2` (or vice versa) and if they're different, the threads change their own location until they find each other.



[9.1] Describe the interfaces that make up the core of the JDBC API (including the `Driver`, `Connection`, `Statement`, and `ResultSet` interfaces and their relationship to provider implementations)

ME-Q45. Which interfaces should be implemented by all JDBC drivers? (Choose two options.)

- ☒ a **Driver**
- ☐ b `DriverManager`
- ☐ c `JDBCConnection`
- ☒ d **Connection**
- ☐ e `DatabaseConnection`

ME-A45. a, d

Explanation: Option (b) isn't an interface—it's a class. The types used in options (c) and (e) aren't defined in the JDBC API. All the JDBC drivers must implement the following:

- `Driver`
- `Connection`
- `Statement`
- `ResultSet`



[4.7] Use java.util.Comparator and java.lang.Comparable

ME-Q46. Given

```
import java.util.*;
enum Title {Mr, Ms, Dr};
class Person implements Comparable<Person> {
    String name; Title title;
    public Person(Title t, String n) {
        title = t;
        name = n;
    }
    public int compareTo(Person p) {
        int compareName = name.compareTo(p.name);
        return compareName != 0 ? compareName : title.compareTo(p.title);
    }
    public String toString() {
        return title + " " + name;
    }
}
class University {
    public static void main(String args[]) {
        ArrayList<Person> list = new ArrayList<>();
        list.add(new Person(Title.Mr, "Selvan"));
        list.add(new Person(Title.Dr, "Shreya"));
        list.add(new Person(Title.Ms, "Shreya"));
        Collections.sort(list);
        for (Person p : list) {
            System.out.println(p);
        }
    }
}
```

what is the output? (Choose the best option.)

- ☐ a Mr Selvan
Dr Shreya
Ms Shreya
- ☒ b **Mr Selvan**
Ms Shreya
Dr Shreya
- ☐ c Ms Shreya
Dr Shreya
Mr Selvan
- ☐ d Dr Shreya
Mr Selvan
Ms Shreya
- ☐ e Ms Shreya
Mr Selvan
Dr Shreya

ME-A46. b

Explanation: The list is sorted first alphabetically on the name of a Person and then on its title. Both classes `String` and `Enum` implement the `Comparable` interface. The enum constant values are sorted on the order of their declaration, not alphabetically.



EXAM TIP Look out for questions on the exam that use the natural sort order of enum constants, like adding enum constants to a sorted collection.



[3.1] Write code that declares, implements, and/or extends interfaces

ME-Q47. Which statements are incorrect? (Choose three options.)

- ☒ a **“Program to an interface” means using interfaces in your code.**
- ☒ b **An interface can’t define static inner classes.**
- ☐ c An interface can’t declare nonfinal variables.
- ☒ d **An interface can declare uninitialized variables.**
- ☐ e An interface can’t define static methods.

ME-A47. a, b, d

Explanation: Option (a) is an incorrect statement. “Program to an interface” refers to using more generalized reference variables in your code to promote flexibility, either of base classes or implemented interfaces.

Option (b) is an incorrect statement. An interface can define an inner class and the members of an interface are implicitly static. So even if you don’t prefix the definition of an inner class with the `static` keyword, it is implicitly static.

Option (c) is a correct statement. The variables of an interface are final by default.

Option (d) is an incorrect statement. The variables of an interface are implicitly static and final. An interface with uninitialized variables won’t compile.

Option (e) is a correct statement. An interface can’t define static methods. All the methods of an interface are implicitly public and abstract.



[10.3] Synchronize thread access to shared data

ME-Q48. Identify the classes whose instances are thread safe and don’t need external synchronization. (Choose four options.)

- ☒ a **Integer**
- ☐ b `StringBuilder`
- ☐ c `ArrayList`
- ☐ d `Map`
- ☒ e **Boolean**
- ☒ f **String**

- ☒ g Double
- ☐ h LinkedList

ME-A48. a, e, f, g

Explanation: Once initialized, the values of instances of immutable classes can't be changed. So instances of these classes are thread safe and they don't require external synchronization.



[3.3] Apply cohesion, low-coupling, IS-A, and HAS-A principles

ME-Q49. You need to design a mining application and have identified the following metallic and nonmetallic entities: mine, copper, and gold. Assuming the natural properties of these entities, which option do you think best shows the relationship among these entities? (Choose one option.)

- ☐ a

```
abstract class Metal{}
abstract class Mine{}
class Copper extends Mine{}
class Gold extends Metal{}
```
- ☒ b

```
abstract class Metal{}
abstract class Mine{}
class Copper extends Metal{}
abstract class Gold extends Metal{}
```
- ☐ c

```
abstract class Mine{}
class Metal extends Mine{}
class Copper extends Metal{}
class Gold extends Metal{}
```
- ☐ d

```
class Mine{}
abstract class Metal {
    abstract void extract();
}
class Copper implements Metal{}
class Gold implements Metal{}
```

ME-A49. b

Explanation: Code for options (a), (b), and (c) compiles successfully. But the relationships in options other than (b) don't represent the correct relationship among the identified entities.

Option (a) is incorrect because class Copper extends Mine. A *metal* isn't a specialized *mine*.

Option (c) is incorrect because Metal IS-A Mine is incorrect.

Option (d) is incorrect. This option uses an incorrect keyword (`implements`) to make a class extend another class. The correct keyword is `extends`.



[4.2] Use the diamond for type inference



[4.3] Analyze the interoperability of collections that use raw types and generic types

ME-Q50. Given

```
class Gift{}
class Book extends Gift{}
class Phone extends Gift{}
```

which options when inserted at //INSERT CODE HERE will enable you to add elements of type Phone to books? (Choose two options.)

```
List<Book> books = new ArrayList<>();
books.add(new Book());
List anotherList = books;
//INSERT CODE HERE
```

- ☒ a `List<Gift> gifts = anotherList;`
`gifts.add(new Phone());`
- ☐ b `List<? extends Gift> gifts = anotherList;`
`gifts.add(new Phone());`
- ☒ c `List<? super Gift> gifts = anotherList;`
`gifts.add(new Phone());`
- ☐ d `List<?> gifts = anotherList;`
`gifts.add(new Phone());`

ME-A50. a, c

Explanation: Options (b) and (d) are incorrect. The code that adds an element to lists that are defined as follows won't compile:

```
List<?> gifts = anotherList;
List<? extends Gift> gifts = anotherList;
```

You can't add an item to the generic types with wildcard declarations `<?>` or `<? extends T>` because you don't know the actual type of objects to which the list refers. To prevent polluting the list, all additions to such lists are disallowed.

The answer options (a) and (c) allow the addition of Phone objects to lists of Book. But an attempt to assign a Phone object to a Book variable using implicit conversion will throw a `ClassCastException`.



[3.3] Apply cohesion, low-coupling, IS-A, and HAS-A principles

ME-Q51. Which option defines loosely coupled classes? (Choose the best option.)

- ☐ **a**

```
class Pen {  
    class Ink {  
    }
```
- ☐ **b**

```
class Pen {  
    public void refill(Ink ink) {  
        if (ink.color.equals("red")) { /*code */ }  
    }  
    class Ink { String color; }
```
- ☒ **c**

```
class Pen {  
    public void refill(String color) {  
        if (color.equals("red")) {  
            buy(new Ink("red"));  
        }  
    }  
    public void buy(Ink ink) {  
        //code  
    }  
    class Ink {  
        String color;  
        Ink(String color) { this.color = color; }  
    }  
}
```
- ☐ **d**

```
class Pen {  
    Ink ink;  
    Pen(Ink ink) {  
        if (ink.color.equals("black") ink = new BlackInk();  
        if (ink.color.equals("red") ink = new RedInk();  
    }  
}  
interface Ink {  
class RedInk implements Ink{  
    String color;  
}  
class BlackInk implements Ink{  
    String color;  
}
```

ME-A51. c

Explanation: Option (a) is incorrect because these classes aren't related.

Option (b) is incorrect because class Pen directly accesses the instance variable color of class Ink, making them tightly coupled classes. Tightly coupled classes are difficult to change. Option (d) doesn't compile.



[5.1] Search, parse, and build strings (including Scanner, StringTokenizer, StringBuilder, String, and Formatter)

ME-Q52. Which option will output the following? (Choose the best option.)

(99,887,766)

- ☒ a `System.out.printf("%(,d", -99887766);`
- ☐ b `System.out.printf("%,d", -99887766);`
- ☐ c `System.out.printf("%(),f", -99887766);`
- ☐ d `System.out.printf("%,l", -99887766);`
- ☐ e `System.out.printf("%,i", -99887766);`
- ☐ f `System.out.printf("%(,i)", -99887766);`
- ☐ g `System.out.printf("(%d)", -99887766);`
- ☐ h `System.out.printf("(%i)", -99887766);`

ME-A52. a

Explanation: Option (b) is incorrect. This option won't enclose the negative value -99887766 in parentheses. It outputs -99,887,766.

Option (c) is incorrect. %f is used for float or double values. When used with an int literal value, it throws an `UnknownFormatConversionException`. Also, this option uses an invalid flag—that is, (). The correct flag to enclose negative numbers within parentheses is (.

Options (d), (e), (f), and (h) are incorrect because they use invalid format specifiers—that is, %l and %i. These options throw an `UnknownFormatConversionException`.

Option (g) is incorrect because it outputs (-99,887,766).



[3.5] Design a class using the Singleton design pattern

ME-Q53. Which option correctly implements a Singleton pattern? (Choose the best option.)

- ☒ a

```
class Heart {
    private static Heart instance;
    private Heart () {}
    public static synchronized Heart getInstance() {
        if (instance == null) {
            instance = new Heart ();
        }
        return instance;
    }
}
```

- ☐ **b** class Heart {
 private Heart instance;
 private Heart () {}
 public synchronized Heart getInstance() {
 if (instance == null) {
 instance = new Heart ();
 }
 return instance;
 }
 }
- ☐ **c** class Heart {
 Heart () {}
 private static class HeartHolder {
 private static final Heart INSTANCE = new Heart ();
 }
 public static Heart getInstance () {
 return HeartHolder.INSTANCE;
 }
 }
- ☐ **d** class Heart {
 private static Heart instance = new Heart();
 protected Heart () {}
 public static Heart getInstance () {
 return instance;
 }
 }

ME-A53. a

Explanation: A Singleton design pattern limits the creation of class instances to just one. To do so

- The constructors of the class must be private so that the class has complete control of creation of its instances.
- The class defines private and static variables to refer to its sole instance, accessible using a static and public method.
- Concurrent initialization of this private and static variable is controlled using either eager initialization (initialization of a variable with its declaration) or by synchronizing the method that initializes the sole Singleton class instance.

Option (b) is incorrect. The variable `instance` and method `getInstance()` are instance members.

Options (c) and (d) are incorrect because the constructors of `Heart` are not private. This allows other classes to instantiate it. If in options (c) and (d) the constructor was private, these options would be valid implementations of the Singleton design pattern.



[4.4] Use wrapper classes, autoboxing, and unboxing

ME-Q54. Given that method `getClass()` returns the name of an object's class, what is the output of the following code? (Choose the best option.)

```
System.out.println((new Integer(10) + new Short((short)100)).getClass());
```

- ☐ a class java.lang.Short
- ☐ b class java.lang.Integer
- ☐ c short
- ☐ d int
- ☒ e **Compilation error**
- ☐ f RuntimeException

ME-A54. e

Explanation: The Integer value 10 and Short value 100 are auto-unboxed to add their value using the addition operator (+). The resultant value is of int type. Because you can't call a method on a primitive data type, the preceding code throws a compilation error:

```
error: int cannot be dereferenced
System.out.println((new Integer(10) + new Short((short)100)).getClass());
                                                                    ^
1 error
```



[1.7] Use package and import statements



[12.1] Read and set the locale by using the Locale object

ME-Q55. Given the following code, which option when inserted at `//INSERT CODE HERE` will enable you to compile the code? (Choose the best option.)

```
//INSERT CODE HERE
class Foo {
    public static void main(String args[]) {
        Locale locale1 = Locale.FRENCH;
        Locale locale2 = Locale.FRANCE;
        Locale locale3 = new Locale.Builder().setLanguage("fr").build();
        Locale locale4 = new Locale.Builder().setLanguage("fr")
                                                .setRegion("FR").build();

        Locale locale5 = new Locale("fr");
        Locale locale6 = new Locale("fr", "FR");
        System.out.print(locale1.equals(locale3));
        System.out.print(locale2.equals(locale6));
        System.out.print(locale4.equals(locale5));
    }
}
```

```

    }
}

☐ a import java.util.Locale*;
☐ b import java.util.locale.*;
☐ c import java.util.locatisation*;
☐ d import java.util.i18n.*;
☒ e import java.util.*;

```

ME-A55. e

Explanation: Class `Locale` is defined in the `java.util` package. Options (a) through (d) are invalid options.



[4.1] Create a generic class

ME-Q56. Which options correctly define generic classes? (Choose three options.)

- ☒ a `class MyClass1 <T, A, B> { /*use only T and B*/ }`
- ☒ b `class MyClass2 <T, A, B> { /*use T, A and B*/ }`
- ☒ c `class MyClass4 <Var1, Var2> { /*...*/ }`
- ☐ d `class MyClass5 <?, ?> { /*...*/ }`
- ☐ e `class MyClass6 <U extends Number, T super Number> { /*...*/ }`

ME-A56. a, b, c

Explanation: Options (a) and (b) are correct. A class might not use all its type parameters. Though this behavior isn't desired, the class will compile successfully. Compare it with a method definition that might not use all the method parameters passed to it.

Option (c) is correct. Oracle recommends the use of a single uppercase letter for type parameters. But using any other variable names—that is, `camelCase` or `ALL_CAPS`—will allow the code to compile.

Options (d) is incorrect because you can't use a wildcard character (?) to define type parameters to a class.

Option (e) is incorrect because it uses an incorrect syntax. The correct syntax is to list the type parameter's name, followed by the `extends` keyword, followed by its upper bound. But in this option, `super` is used instead of `extends`.



[12.3] Call a resource bundle from an application

ME-Q57. Given

```

Locale locale = Locale.US;
ResourceBundle labels = null;

```

which option loads the resource bundle `MyMsgs` for `Locale.US`? (Choose the best option.)

- ☒ a `labels = ResourceBundle.getBundle("MyMsgs", locale);`
- ☐ b `labels = ResourceBundle.loadBundle("MyMsgs", locale);`
- ☐ c `labels = ResourceBundle.loadBundleFamily("MyMsgs", locale);`
- ☐ d `labels = ResourceBundle.getBundleMessages("MyMsgs", locale);`

ME-A57. a

Explanation: Options (b), (c), and (d) define invalid method names to load the resource bundle.



[3.7] Design and create objects using a factory pattern

ME-Q58. Which code examples don't initialize variables using a factory? (Choose two options.)

- ☐ a `Connection con = /*code to get a valid connection instance */`
`Statement statement = con.createStatement();`
- ☐ b `Statement st = /*code to get a valid Statement instance */`
`ResultSet rs = st.executeQuery("SELECT * FROM book");`
- ☒ c `NumberFormat nf = new NumberFormat();`
- ☒ d `NumberFormat nf = new NumberFormat(Locale.JP);`
- ☐ e `NumberFormat nf = NumberFormat.getCurrencyInstance();`
- ☐ f `ResourceBundle myBundle = ResourceBundle.getBundle("myBundle", Locale.FR);`

ME-A58. c, d

Explanation: Note that the question asks you for options that *don't* use factory methods.

Options (c) and (d) are incorrect because they don't compile—both call `new` operator on `NumberFormat`, an abstract class. Also, the Factory design pattern discourages directly calling `new` to create instances of a class. It creates and returns objects depending on the parameters supplied to it. It removes the dependency of the calling class to know all the details of the exact (derived) class whose object is returned.



[11.3] Use `Executor`, `ExecutorService`, `Executors`, `Callable`, and `Future` to execute tasks using thread pools

ME-Q59. Which method is defined in the `Executor` interface? (Choose one option.)

- ☒ a `void execute(Runnable obj)`
- ☐ b `void execute(Callable obj)`

- ☐ c void execute(Thread obj)
- ☐ d void execute(Executable obj)

ME-A59. a

Explanation: Method `execute()` in the `Executor` interface doesn't accept method parameters of type `Callable`, `Thread`, or `Executable`.



[3.1] Write code that declares, implements, and/or extends interfaces

ME-Q60. Given

```
interface Movable {
    void move(int x, int y);
}
```

which code snippet will compile successfully? (Choose one option.)

- ☒ a interface Jumpable extends Movable {}
- ☐ b interface Jumpable implements Movable {}
- ☐ c class Position {
 interface Jumpable extends Movable {
 Position move(int posX, int posY);
 }
}
- ☐ d interface Jumpable extends Movable {
 private String move(long x, int y);
}

ME-A60. a

Explanation: Option (b) is incorrect. An interface can extend another interface by using the keyword `extends` and not `implements`.

Option (c) is incorrect. Method `move()` in `Jumpable` can't overload `move()` in `Movable` by only changing its return type. The method parameter names are just placeholders—changing them to `posX` and `posY` instead of using `x` and `y` won't affect the method signatures.

Option (d) is incorrect. Because the members of an interface are implicitly public, an interface can't define private members.



[3.6] Write code to implement the DAO pattern

ME-Q61. Given

```
import java.util.List;
public class Book {
    private String isbn;
    private String title;
```

```

private String author;
public void setISBN(String val) { isbn = val; }
public String getISBN() { return this.isbn; }
public void setTitle(String val) { title = val; }
public String getTitle() { return title; }
public void setAuthor(String val) { author = val; }
public String getAuthor() { return author; }
public void addBook(Book b) throws Exception { /* code */ }
public void removeBook(String isbn) throws Exception { /* code */ }
public void updateBook(Book b) throws Exception { /* code */ }
public Book getBook(String isbn) throws Exception { return null; }
public List getAllBooks() throws Exception { return null; }
}

```

which set of methods should be moved to a new class to implement the DAO pattern? (Choose the best option.)

- ☐ a `getISBN()`, `getTitle()`, `getAuthor()`
- ☐ b `setISBN()`, `setTitle()`, `setAuthor()`
- ☐ c `getISBN()`, `getTitle()`, `getAuthor()`, `setISBN()`, `setTitle()`, `setAuthor()`
- ☒ d `addBook()`, `removeBook()`, `updateBook()`, `getBook()`, `getAllBooks()`

ME-A61. d

Explanation: In the DAO design pattern, methods that access and manipulate a complete entity are defined in a separate class. It makes the classes highly cohesive. For example, by moving methods mentioned in option (d) to a new class—say, `BookDAO`—`Book` doesn't need to be bothered about how `Book` data is stored and accessed from a data store.



[10.2] Manage and control thread lifecycle

ME-Q62. Which option is correct for the following code? (Choose the best option.)

```

class MyPen extends Thread {
    public void run() {
        synchronized(System.out) {
            try {
                Thread.sleep(5000); //line1
            }
            catch (InterruptedException e) {}
        }
    }
}

```

- ☐ a On `//line1`, the `MyPen` thread will give up ownership of the monitor on `System.out` for at least 5,000 ms.
- ☐ b On `//line1`, the `MyPen` thread will give up ownership of the monitor on `System.out` for exactly 5,000 ms.

- ☒ c On //line1, the MyPen thread won't give up ownership of the monitor on System.out.
- ☐ d None of the above

ME-A62. c

Explanation: Calling `sleep()` doesn't make a thread give up the ownership of monitors.



[12.2] Build a resource bundle for each locale

ME-Q63. An application needs to load its localized messages from a Java class. Which option defines the correct class definition? (Choose the best option.)

- ☐ a

```
public class MyMessages extends ResourceBundle {
    protected Object[][] getContents() {
        return new Object[][] {
            {"day", "Day"},
            {"time", "Time"}
        };
    }
}
```
- ☒ b

```
public class MyMessages extends ListResourceBundle {
    protected Object[][] getContents() {
        return new Object[][] {
            {"day", "Day"},
            {"time", "Time"}
        };
    }
}
```
- ☐ c

```
public class MyMessages extends ClassResourceBundle {
    protected Set<String> handleKeySet() {
        return new HashSet<String>(Arrays.asList("day"));
    }
}
```
- ☐ d

```
public class MyMessages extends Bundle {
    public Object[][] getValues() {
        return new Object[][] {
            {"day", "Day"},
            {"time", "Time"}
        };
    }
}
```

ME-A63. b

Explanation: Option (a) is incorrect because class `MyMessages` doesn't implement the abstract methods of `ResourceBundle`—`getKeys` and `handleGetObject`. Options (c) and (d) extend nonexistent classes in the Java API: `ClassResourceBundle` and `Bundle`.



[8.6] Watch a directory for changes by using the WatchService interface

ME-Q64. You coded a text editor in Java that can be used to open, edit, and save multiple text files. Your application should reload a text file, if it's modified by any other application. Assuming that your target directory is /home/selvan, which option can you use to register the relevant events for your directory? (Choose the best option.)

- ☐ a `Path dir = Paths.get("/home/selvan");`
`WatchService watcher = FileSystems.getDefault().newWatchService();`
`WatchKey key = dir.register(StandardWatchEventKinds.ENTRY_MODIFY);`
- ☒ b `Path dir = Paths.get("/home/selvan");`
`WatchService watcher = FileSystems.getDefault().newWatchService();`
`WatchKey key = dir.register(watcher,`
`StandardWatchEventKinds.ENTRY_MODIFY);`
- ☐ c `Path dir = Paths.get("/home/selvan");`
`WatchService watcher = FileSystems.getDefault().newWatchService();`
`WatchKey key = dir.register(watcher,`
`StandardWatchEventKinds.WATCH_MODIFY);`
- ☐ d `Path dir = Paths.get("/home/selvan");`
`WatchKey key = dir.register(Watcher.getInstance(),`
`StandardWatchEventKinds.WATCH_MODIFY);`

ME-A64. b

Explanation: Option (a) is incorrect because it doesn't pass a Watcher instance to register().

Option (c) is incorrect because it uses an invalid StandardWatchEventKinds constant, WATCH_MODIFY.

Option (d) is incorrect because Watcher.getInstance() won't compile.



[11.1] Use collections from the java.util.concurrent package with a focus on the advantages over and differences from the traditional java.util collections

ME-Q65. Which method can be used to replace a key-value pair if it exists in a ConcurrentMap? (Choose the best option.)

- ☐ a `replace(K key, V value)`
- ☒ b `replace(K key, V old, V new)`
- ☐ c `replaceIfPresent(K key, V old, V new)`
- ☐ d `modifyIfPresent(K key, V old, V new)`
- ☐ e `removeIfPresent(K key, V old, V new)`

ME-A65. b

Explanation: Option (a) is incorrect. This option will replace the value corresponding to the key, if it's mapped to any value and not the key-value pair specified.

Options (c), (d), and (e) are incorrect because they refer to invalid method names.



[2.4] Create top-level and nested classes

ME-Q66. Given

```
public class Inner {
    public class Outer {}
}
```

which of the following instantiates class Outer in another class, say, Test? (Choose one option.)

- ☐ a new Outer().new Inner();
- ☒ b new Inner().new Outer();
- ☐ c Outer.Inner();
- ☐ d Outer.new Inner();
- ☐ e Inner.Outer();
- ☐ f Inner.new Outer();
- ☐ g new Inner.Outer();
- ☐ h new Outer.Inner();

ME-A66. b

Explanation: Take note of the names of the inner and outer classes in the code snippet. The name of the *inner* class is Outer and that of the *outer* class is Inner. *All the code snippets used in the exam questions don't use meaningful names.* You're sure to see names like ClassA, aMethod, A, B, and even Sobber!

Option (b) is correct. It uses the operator new to instantiate the outer class Inner and then again uses new to instantiate its inner class Outer.

None of the options but (b) will compile. Option (g) will instantiate class Outer if it's declared as a static inner class.



[2.4] Create top-level and nested classes

ME-Q67. Given

```
class Laptop {
    static class Model {}
}
```

which option can be assigned to a variable of type Laptop.Model? (Choose the best option.)

- ☒ a `new Laptop.Model();`
- ☐ b `new Laptop().new Model();`
- ☐ c `Laptop.Model;`
- ☐ d `Laptop.new Model();`

ME-A67. a

Explanation: Option (b) instantiates class `Model` as a nonstatic inner class.



[4.1] Create a generic class



[11.3] Use `Executor`, `ExecutorService`, `Executors`, `Callable`, and `Future` to execute tasks using thread pools

ME-Q68. Given

```
1. ExecutorService pool = Executors.newFixedThreadPool(10);
2. Job job = new Job();
3. pool.submit(job);
```

which options define the correct definition of class `Job`, which compiles without errors or warnings? (Choose two options.)

- ☐ a `class Job implements Callable {`
`public void call() {}`
`}`
- ☐ b `class Job implements Callable<Void> {`
`public Void call() {}`
`}`
- ☒ c `class Job implements Callable<String> {`
`public String call() {`
`return "Success";`
`}`
`}`
- ☒ d `import java.io.*;`
`class Job implements Callable<File> {`
`public File call() throws FileNotFoundException {`
`return new File("abcd");`
`}`
`}`

ME-A68. c, d

Explanation: You can pass instances of either `Runnable` or `Callable` to `submit()` of `ExecutorService`. Overloaded `submit()` methods include

```
Future<?> submit(Runnable task);
<T> Future<T> submit(Runnable task, T result);
<T> Future<T> submit(Callable<T> task);
```

All the options in this question implement the `Callable` interface. Here's the definition of `Callable`:

```
public interface Callable<V> {
    V call() throws Exception;
}
```

A class that implements `Callable` must pass a type parameter to the class's declaration and use the same type as a return value for its method call. Because `call()` in `Callable` throws an `Exception`, the overriding method can choose not to throw any exception or throw any checked exception. Any method can throw an unchecked exception (`RuntimeException` and errors), even if the overridden method doesn't.

Option (a) is incorrect because it doesn't pass the type argument to class `Job`, generating a compiler warning. Also its method `call()` doesn't implement a `call()` from `Callable` correctly.

Option (b) is incorrect because it misses the return statement in `call()`.



[12.3] Call a resource bundle from an application

ME-Q69. Given that `labels` refers to a valid `ResourceBundle` resource the contents of which are

```
day=7
time=true
```

which options can be used to read the values of the keys "day" and "time"? (Choose two options.)

- ☐ a `labels.getInt("day");`
`labels.getBoolean("time");`
- ☐ b `labels.getValue("day");`
`labels.getValue("time");`
- ☐ c `labels.readValue("day");`
`labels.readValue("time");`
- ☒ d `labels.getString("day");`
`labels.getString("time");`
- ☒ e `labels.getObject("day");`
`labels.getObject("time");`

ME-A69. d, e

Explanation: `ResourceBundle` defines methods `getString()` and `getObject()` that return values corresponding to the keys as a `String` or an `Object`. Options (a), (b), and (c) define invalid method names.



[4.3] Analyze the interoperability of collections that use raw types and generic types

ME-Q70. Given

```
class Gift{}
class Book extends Gift{}
class Phone extends Gift{}
```

and the code

```
List<Book> books = new ArrayList<>();
books.add(new Book());
List anotherList = books;
List<? super Gift> gifts = anotherList;
gifts.add(new Phone());
//INSERT CODE HERE
System.out.println(item);
```

when inserted at //INSERT CODE HERE, which options can be used to output all values of collection gifts? (Choose two options.)

- ☐ a for (Book item : books)
- ☒ b for (Gift item : books)
- ☒ c for (Object item : books)
- ☐ d for (Phone item : books)

ME-A70. b, c

Explanation: Option (a) throws a `ClassCastException`: Phone cannot be cast to Book.

Option (d) fails compilation with the following message:

```
error: incompatible types
    for (Phone item : books)
        ^
required: Phone
found:      Book
```



[1.7] Use package and import statements



[2.5] Use enumerated types

ME-Q71. Assuming that classes `Color` and `Room` are defined in separate packages and source files

```
package artist;
public enum Color {
    RED, YELLOW, BLUE;
}
```

```
package city;
import artist.Color;
class Room {
    Color myColor = /*insert code here*/
}
```

which options when inserted at `/*insert code here*/` will assign the enum `Color` constant `RED` to the variable `myColor`? (Choose two options.)

- ☒ a `Color.RED;`
- ☐ b `RED;`
- ☒ c `artist.Color.RED;`
- ☐ d `new Color("RED");`

ME-A71. a, c

Explanation: Option (a) is correct. The enum constants are implicitly static and public. Because enum `Color` is declared public, its constant `Color.RED` is accessible in all classes across all packages.

Option (b) is incorrect. `RED` should be prefixed with its enum's name.

Option (c) is correct. A class name can be referred to using its fully qualified name: `packagename.classname`.

Option (d) is incorrect because no other class or interface can call an enum's constructor and create new enum instances.



[11.3] Use `Executor`, `ExecutorService`, `Executors`, `Callable`, and `Future` to execute tasks using thread pools

ME-Q72. Given

```
Executor executor = /* assigns an Executor instance */
Runnable runnable = /* assigns a Runnable instance */
```

how many threads will execute the `Runnable` task for the following code? (Choose the best option.)

```
executor.execute(runnable);
```

- ☐ a One for each submitted task
- ☐ b Only one thread for each task
- ☒ c **Depends on the `Executor` implementation**
- ☐ d Depends on the property `Executor.threads` set during runtime

ME-A72. c

Explanation: `Executor` is an interface. Its instances execute submitted `Runnable` tasks. This interface provides a way of decoupling task submission from the mechanics of

how each task will be run, including details of thread use, scheduling, and so on. An `Executor` is normally used instead of explicitly creating threads. Method `execute()` executes the given command at some time in the future. The command may execute in a new thread, use a pooled thread, or execute in the calling thread, at the discretion of the `Executor` implementation. For example, class `SeparateThreadExecutor` would execute `Runnable` objects in a separate thread of execution but class `SameThreadExecutor` doesn't:

```
class SeparateThreadExecutor implements Executor {
    public void execute(Runnable r) {
        new Thread(r).start();
    }
}
class SameThreadExecutor implements Executor {
    public void execute(Runnable r) {
        r.run();
    }
}
```



[8.1] Operate on file and directory paths with the `Path` class

ME-Q73. What is the output of the following code? (Choose one option.)

```
Path path1 = Paths.get("C:/OCP/8-1.txt");
Path path2 = Paths.get("C:", "OCP", "mock", "8-1.txt");
Path path3 = path1.resolve(path2.relativeTo(path1));
System.out.println(path3);
```

- ☐ a `..\mock\8-1.txt`
- ☐ b `..\..\8-1.txt`
- ☒ c `C:\OCP\8-1.txt\..\..\8-1.txt`
- ☐ d `C:\OCP\8-1.txt\..\mock\8-1.txt`
- ☐ e `C:\8-1.txt`

ME-A73. c

Explanation: `path2.relativeTo(path1)` will construct a relative path between `path2` and `path1`, which returns

```
..\..\8-1.txt
```

`resolve()` resolves the given path against the path on which it's called. `path1.resolve(path2.relativeTo(path1))` will resolve `C:/OCP/8-1.txt` against `..\..\8-1.txt`. Because `..\..\8-1.txt` doesn't have a root component, `resolve()` simply joins the given path to `path1` and returns `C:\OCP\8-1.txt\..\..\8-1.txt`.

If the given path has a root component, then resolution is highly implementation dependent and therefore unspecified.



[2.5] Use enumerated types



[2.3] Use the static and final keywords

ME-Q74. Given

```
enum Shade {LIGHT, DARK};
class Color {
    final int shade;
    Color(Shade val){
        //initialize
    }
}
```

which code snippet when inserted at `//initialize` will enable class `Color` to compile successfully? (Choose two options.)

- ☒ **a** `switch (val) {`
 `case LIGHT : shade = 11; break;`
 `case DARK : shade = 22; break;`
 `default: shade = 33;`
 `}`
- ☐ **b** `switch (val) {`
 `case LIGHT : shade = 11;`
 `case DARK : shade = 22;`
 `default: shade = 33;`
 `}`
- ☐ **c** `switch (val) {`
 `case LIGHT : shade = 11; break;`
 `case DARK : shade = 22; break;`
 `}`
- ☒ **d** `if (val.equals(Shade.LIGHT)) shade = 11;`
 `else shade = 22;`
- ☐ **e** `if (val.equals(Shade.LIGHT)) shade = 11;`
 `else if (val.equals(Shade.DARK)) shade = 22;`

ME-A74. a, d

Explanation: A final variable must be initialized exactly once. If you use conditional constructs like if-else or switch, the compiler must ascertain that the final variable is initialized exactly once in *every* condition.

Option (b) is incorrect. In absence of break statements, the code execution will fall through the switch cases, resulting in multiple assignments to final variable `shade`.

Options (c) and (e) are incorrect. Though the enum `Shade` defines only two constants, `LIGHT` and `DARK`, the compiler must ascertain that the final variable `shade` is assigned a value in all cases. What happens if the enum `Shade` is modified and new

constant values are added to it? So the code needs default (for switch) or else (for if-else if) statements.



[1.3] Overload constructors and methods



[6.1] Use the throw statement and throws clause

ME-Q75. What is the result of the following code? (Choose the best option.)

```
class Metal {
    {
        try {
            throw new RuntimeException();
        }
        finally{
            System.out.print("finally-");
        }
    }
    Metal() {
        System.out.print("Metal-");
    }
}
class Copper extends Metal{
    Copper(int a) {
        System.out.print("Copper-");
    }
    public static void main(String args[]) {
        new Copper(101);
    }
}
```

- ☐ a catch-finally-Metal-Copper-
- ☐ b Metal-catch-finally-Copper-
- ☐ c RuntimeException
- ☒ d **Compilation error**

ME-A75. d

Explanation: Instance initializer(s) of a class must complete normally to enable the class to compile. The code in class Metal's instance initializer throws a RuntimeException, which isn't handled. So compilation of class Metal fails with the following message:

```
Copper.java:2: error: initializer must be able to complete normally
```



[11.3] Use `Executor`, `ExecutorService`, `Executors`, `Callable`, and `Future` to execute tasks using thread pools

ME-Q76. What is the correct option for the given code? (Choose the best option.).

```
Executors.newFixedThreadPool(5);
```

- ☒ **a** The code creates a thread pool that reuses five threads operating off a shared, unbounded queue.
- ☐ **b** The code creates five thread pools.
- ☐ **c** The code creates five thread pools with a predefined number of active threads.
- ☐ **d** The code creates a thread pool that can create multiple threads, but reuses at least five of them off a shared, unbounded queue.

ME-A76. a

Explanation: Here's the method signature of `newFixedThreadPool`:

```
public static ExecutorService newFixedThreadPool(int nThreads)
```

According to the Java API, this method creates a thread pool that reuses a fixed number of threads operating off a shared, unbounded queue. At any point, at most `nThreads` will be active processing tasks. If additional tasks are submitted when all threads are active, they will wait in the queue until a thread is available. If any thread terminates due to a failure during execution prior to shutdown, a new one will take its place if needed to execute subsequent tasks. The threads in the pool will exist until it is explicitly shut down.



[2.3] Use the `static` and `final` keywords



[2.2] Construct abstract Java classes and subclasses



[3.1] Write code that declares, implements, and/or extends interfaces

ME-Q77. What is the result of the following code? (Choose one option.)

```
class A {
    static int age = 10;
}
interface B {
    int age = 20;
}
```

```
class C extends A implements B {
    static int age = 30;
}
class MyTest {
    public static void main(String args[]) {
        B b = new C();
        System.out.println(b.age);
    }
}
```

- ☐ a 10
- ☒ b 20
- ☐ c 30
- ☐ d Compilation error
- ☐ e RuntimeException

ME-A77. b

Explanation: The variables of an interface are implicitly static.

Class A, interface B, and class C all define a static variable age. Because the static variables don't participate in runtime polymorphism, the type of the reference variable will determine which of these static variables is referred to by a variable. In the example code, the type of variable b is B, so b.age refers to the (implicitly) static variable age defined in interface B.

A quick question: Did successful compilation of class C puzzle you? The static variable age defined in C hides

- The static variable age defined in its base class A
- The static variable age accessible using the interface B

But the following code won't compile because the reference to variable age in class C is ambiguous (both variable age in A and variable age in B match):

```
class A {
    static int age = 10;
}
interface B {
    int age = 20;
}
class C extends A implements B {
    static void print() {
        System.out.println(age);
    }
}
```

**[12.2] Build a resource bundle for each locale**

ME-Q78. The default locale of a JVM is `Locale.JAPAN`, and an application includes the following resource bundles:

- `AppMessages_fr.properties`
- `AppMessages_fr_FR.properties`
- `AppMessages_en.properties`
- `AppMessages_ja_JP.properties`
- `AppMessages_ja.properties`
- `AppMessages_JP.properties`

Which resource bundle will the application load for `Locale.CHINA`? (Choose one option.)

- ☐ a `AppMessages_fr.properties`
- ☐ b `AppMessages_fr_FR.properties`
- ☐ c `AppMessages_en.properties`
- ☒ d **`AppMessages_ja_JP.properties`**
- ☐ e `AppMessages_ja.properties`
- ☐ f `AppMessages_JP.properties`
- ☐ g Compilation error
- ☐ h `RuntimeException`

ME-A78. d

Explanation: Following is the search order for a resource bundle for a specified locale:

- 1 `bundleName_localeLanguage_localeCountry_localeVariant`
- 2 `bundleName_localeLanguage_localeCountry`
- 3 `bundleName_localeLanguage`
- 4 `bundleName_defaultLanguage_defaultCountry_defaultVariant`
- 5 `bundleName_defaultLanguage_defaultCountry`
- 6 `bundleName_defaultLanguage`
- 7 `bundleName`

Because there isn't any matching resource bundle for `Locale.CHINA`, the application loads the resource bundle for the default locale `Locale.JAPAN`—that is, `AppMessages_ja_JP.properties`.

If no matching resource bundle is found, Java Runtime tries to load the base resource bundle—that is, one that doesn't include any additional name components, `bundleName`. If even this can't be found or loaded, Java Runtime throws a `MissingResourceException`.



[1.2] Override methods



[1.3] Overload constructors and methods



[2.2] Construct abstract Java classes and subclasses

ME-Q79. Given

```
abstract class Wood {
    public abstract void floats();
}
```

which option extends class Wood correctly? (Choose the best option.)

- ☒ **a** `abstract class Boat extends Wood {}`
- ☐ **b** `class Boat extends Wood {
 float floats() {}
}`
- ☐ **c** `final class Boat extends Wood {
 abstract void floats() {}
}`
- ☐ **d** `class Boat extends Wood {
 void floats() {}
}`

ME-A79. a

Explanation: Option (b) is incorrect. Class Boat tries to implement the abstract method `floats()` by only modifying its return type, which isn't allowed. It also tries to assign a weaker access (*default*) to `floats()`, which is defined as a public method in Wood, which is also not allowed.

Option (c) is incorrect. In this option, class Boat extends Wood, defining `floats()` as an abstract method with a body and a weaker access. Class Boat is also marked `final` even though it defines abstract method `floats()`. A class that defines abstract methods can't be marked as a final class.

Option (d) is incorrect. Method `floats()` in Boat tries to implement `floats()` using a weaker access, which isn't allowed.



[10.4] Identify code that may not execute correctly in a multithreaded environment

ME-Q80. Which statement is true for the given code? (Choose the best option.)

```
class Foo extends Thread {
    String name;
```

```

    Foo(String name) {this.name = name;}
    public static int start = 10;
    public static int end = 20;
    public void run() {
        for (; start < end; start++)
            System.out.println(name + start);
    }
}
class TestFoo {
    public static void main(String args[]) {
        Thread t1 = new Foo("T1:"); t1.start();
        Thread t2 = new Foo("T2:"); t2.start();
    }
}

```

- ☐ a The code will output exactly 10 numbers.
- ☐ b The code will output exactly 20 lines.
- ☒ c **Threads t1 and t2 can output the same start value.**
- ☐ d Threads t1 and t2 will never output the same start values.
- ☐ e Thread t1 can never print a lower start value than that printed by thread t2.

ME-A80. c

Explanation: Options (a) and (b) are incorrect. Method `run()` uses static variables in its `for` loop. Because the static variables are shared across objects and threads, the exact count of the values that the code outputs can vary for each program execution.

Option (c) is correct and (d) is incorrect. Because the values of the static variables are shared across threads, it's possible that multiple threads output the same value.

Option (e) is incorrect. Here's one of the probable outputs of the code:

```

T1:10
T2:10
T1:11
T2:12
T1:13
T2:14
T2:16
T2:17
T2:18
T2:19
T1:15

```

By the time thread `t1` outputs the value by sending it to `System.out`, the other thread might have already incremented and printed the latter values.



[2.3] Use the static and final keywords



[6.1] Use the throw statement and throws clause

ME-Q81. Given

```
import java.io.*;
class Factory {
    static int count = initCount();
    static int initCount() throws FileNotFoundException {
        int result = 0;
        FileInputStream fin = new FileInputStream(new File("abc.txt"));
        //read fin and initialize result
        return result;
    }
}
```

which option is correct? (Choose the best option.)

- ☐ a Class Factory will throw a FileNotFoundException if file abc.txt can't be found.
- ☐ b Class Factory will throw a FileNotFoundException irrespective of whether or not file abc.txt can't be found.
- ☐ c The code will initialize static variable count successfully.
- ☒ d **Compilation error**

ME-A81. d

Explanation: The code that declares and initializes the variable count using initCount() won't compile. Method initCount() declares to throw a checked exception, FileNotFoundException. To use initCount() the calling code should either

- 1 Enclose its use within a try block and catch FileNotFoundException or its superclass.
- 2 Declare FileNotFoundException or its superclass to be rethrown.
- 3 Do both options (1) and (2).

For example, the following code implements the solution suggested in (1):

```
class Factory {
    static int count;
    static {
        try {
            count = initCount();
        }
        catch (FileNotFoundException e) { /* code */ }
    }
    static int initCount() throws FileNotFoundException {
        int result = 0;
        FileInputStream fin = new FileInputStream(new File("abc.txt"));
    }
}
```

```

        //read fin and initialize result
        return result;
    }
}

```



[10.4] Identify code that may not execute correctly in a multithreaded environment

ME-Q82. Two threads can't reach completion. What is the probable cause? (Choose the best option.)

```

class MyThread extends Thread{
    Runnable other;
    void setOther(Runnable r) {other = r;}
    public void run() {
        synchronized(this) {
            System.out.print("XYZ");
            synchronized(other) {
                System.out.print("ABC");
            }
        }
    }
}

class EJavaGuru {
    public static void main(String args[]) {
        MyThread one = new MyThread();
        MyThread two = new MyThread();
        one.setOther(two);
        two.setOther(one);
        one.start();
        two.start();
    }
}

```

- ☒ **a Deadlock**
- ☐ **b Livelock**
- ☐ **c Starvation**
- ☐ **d Mutual lock**

ME-A82. a

Explanation: The question states that the two threads can't reach completion and asks you to identify the cause. Method `run()` in `MyThread` acquires two locks, the first on itself and the other on its instance member `other`. Method `main()` in class `EJavaGuru` creates two threads and assigns each other as its other instance member. So both threads acquire a lock on them and then try to acquire a lock on the other thread. Because the threads can't reach completion, the most obvious reason is that they're deadlocked, where each thread is waiting to acquire a lock on the other thread, while holding a lock on them.



[2.5] Use enumerated types

ME-Q83. Given

```
enum Metal {
    COPPER, GOLD;
    Metal() {
        System.out.print("constructor:");
    }
    static {
        System.out.print("static:");
    }
    public static void main(String args[]) {
        System.out.print(Metal.COPPER + ":");
    }
}
```

what is the result? (Choose the best option.)

- ☐ a COPPER:static:constructor:constructor:
- ☐ b static:constructor:constructor:COPPER:
- ☒ c **constructor:constructor:static:COPPER:**
- ☐ d COPPER:constructor:constructor:static:

ME-A83. c

Explanation: The creation of an enum constant occurs in the enum's static initializer, before execution of the rest of the code (explicitly) defined in the static initializer block. This explains why the code outputs constructor, twice, before static.



[10.1] Create and use the Thread class and the Runnable interface

ME-Q84. Given

```
enum Color {VIOLET, INDIGO, BLUE, GREEN, YELLOW, ORANGE, RED}
```

when started as a thread, instances of which class would output the enum values with an interval of at least one second? (Choose the best option.)

- ☒ a **class Rainbow extends Thread {**
 public void run() {
 for (Color color : Color.values()) {
 System.out.println(color);
 }
 }
 }

- ☐ **b** class Rainbow implements Runnable {
 public void run() {
 for (Color color : Color.values()) {
 System.out.println(color);
 }
 }
 }
- ☐ **c** class Rainbow extends Thread {
 public void run() {
 for (Color color : Color.values()) {
 Thread.sleep(1000);
 System.out.println(color);
 }
 }
 }
- ☐ **d** class Rainbow extends Thread {
 public void run() throws InterruptedException {
 for (Color color : Color.values()) {
 Thread.sleep(1000);
 System.out.println(color);
 }
 }
 }
- ☒ **e** None of the above

ME-A84. e

Explanation: Options (a) and (b) are incorrect because they won't pause the current thread so that the enum values output at an interval of at least one second.

Option (c) won't compile because it doesn't handle the checked exception, `InterruptedException`, thrown by `Thread.sleep()`.

Option (d) won't compile due to incorrect method overriding. Because the overridden method `run()` in `Thread` doesn't throw any checked exception, the overriding method `run()` in `Rainbow` can't throw any checked exception. One correct code is

```
class Rainbow extends Thread {
    public void run() {
        for (Color color : Color.values()) {
            try {
                Thread.sleep(1000);
            }
            catch (InterruptedException e) {}
            System.out.println(color);
        }
    }
}
```



[1.6] Override methods from the `Object` class to improve the functionality of your class

ME-Q85. What is the result of the following code? (Choose the best option.)

```
import java.util.*;
class Color {
    String value;
    Color(String v) {value = v;}
    public int hashCode() {return 999; }
    public String toString() {return value;}
}
class Rainbow {
    public static void main(String args[]) {
        HashSet<Color> set = new HashSet<>();
        set.add(new Color("red"));
        set.add(new Color("yellow"));
        set.add(new Color("blue"));
        Iterator it = set.iterator();
        while (it.hasNext())
            System.out.print(it.next() + "-");
    }
}
```

- ☐ a red-yellow-blue-
- ☐ b blue-red-yellow-
- ☐ c red-
- ☒ d **The retrieval order might change with each execution.**

ME-A85. d

Explanation: A `HashSet` makes no guarantee as to the iteration order of the set; in particular, it doesn't guarantee that the order will remain constant over time.

To trick and confuse you, note that the class `Color` includes a definition of method `hashCode()`. Java is case sensitive and `hashCode()` isn't the same as `hashcode()`. Even when this is fixed (`hashCode()` instead of `hashcode()`) and thus all have the same hash code, the retrieval order is *still* unknown and not guaranteed, because `hashCode()` has nothing to do with the retrieval order. It's only used to get a uniform distribution of objects in the different buckets to improve performance. To return an object or to add one (and preserve uniqueness) you only need to check the bucket with the same hash code (and not all elements in your set).



[1.6] Override methods from the Object class to improve the functionality of your class



[4.6] Create and use Map implementations

ME-Q86. Given

```
import java.util.*;
class Color {
    static int count = 0;
    public int hashCode() {
        ++count;
        return super.hashCode();
    }
}
class Rainbow2 {
    public static void main(String args[]) {
        HashMap<Color, String> map = new HashMap<>();
        Color c1 = new Color(); Color c2 = new Color();
        map.put(c1, "Red");
        map.put(c2, "Yellow");
        map.get(new Color());
        map.get(c1);
        System.out.print(Color.count);
    }
}
```

what is the output? (Choose one option.)

- ☐ a 0
- ☐ b 1
- ☐ c 2
- ☐ d 3
- ☒ e 4
- ☐ f Undefined
- ☐ g Compilation error
- ☐ h RuntimeException

ME-A86. e

Explanation: The HashMap retrieves the hash-code value of its key object (by calling its method `hashCode()`), when

- It adds a key-value pair by using `put()`.
- It retrieves an object corresponding to a key by using `get()`.

Class Rainbow2 adds two objects and retrieves two—outputting four.



[10.2] Manage and control thread lifecycle

ME-Q87. Assume that thread `class1` doesn't own object `ObjectA`'s monitor lock. What happens if it calls `wait()` or `notify()` on it? (Choose one option.)

- ☐ a The code won't compile.
- ☒ b **The code will throw a `RuntimeException`.**
- ☐ c `class1` will try to acquire `objectA`'s monitor lock and then execute `wait()` or `notify()`.
- ☐ d `class1` will execute `wait()` or `notify()` without waiting to acquire the lock on `objectA`'s monitor.
- ☐ e None of the above

ME-A87. b

Explanation: Option (a) is incorrect because the compiler can't determine if a class owns the object monitor or not. If a class calls `wait()` or `notify()` on an object without owning a lock on its monitor, the JVM will throw an `IllegalMonitorStateException`.



[1.2] Override methods



[1.5] Use virtual method invocation



[5.3] Format strings using the formatting parameters `%b`, `%c`, `%d`, `%f`, and `%s` in format strings

ME-Q88. Given

```
class Fibre {
    String type = "Fibre";
    String type() {
        return type;
    }
}
class Silk extends Fibre {
    String type = "Silk";
    String type() {
        return type;
    }
}
class Cloth {
    public static void main(String args[]) {
        Fibre f = new Silk();
```

```

        System.out.printf("%s : %s", f.type(), f.type());
    }
}

```

what is the output of class Cloth? (Chose one option.)

- ☐ a Fibre : Fibre
- ☐ b Fibre : Silk
- ☒ c **Silk : Fibre**
- ☐ d Silk : Silk
- ☐ e Compilation error
- ☐ f RuntimeException

ME-A88. c

Explanation: The instance variables are bound to a reference variable during compilation, but the methods are bound at runtime.

Class Silk extends class Fibre, so a reference variable of type Fibre can be used to refer to an instance of class Silk. A reference variable determines which instance variables can be accessed and which instance methods can be invoked. Because both classes Silk and Fibre define an instance variable type and an instance method type(), type and type() can be called on reference variable f.

The type of reference variable f is Fibre and it refers to an instance of class Silk. Unlike methods, there's no polymorphism with instance variables—they're bound to a reference variable during the compilation process. So f.type accesses the variable type defined in class Fibre. But the method execution is polymorphic and it depends on the type of the instance on which it's called at runtime. So f.type() calls method type() defined in the derived class Silk.



[2.3] Use the static and final keywords



[3.1] Write code that declares, implements, and/or extends interfaces

ME-Q89. Given

```

class A {
    static int age() {return 10;}
}
interface B {
    int age();
}
class C extends A implements B {
    //INSERT CODE HERE
}

```

which option when inserted at //INSERT CODE HERE will enable class C to compile successfully? (Choose one option.)

- ☐ a `public int age() {return 20;}`
- ☐ b `public static int age() {return 20;}`
- ☐ c `public int age() {return 20;}`
`public static int age() {return 20;}`
- ☒ d **None of the above**

ME-A89. d

Explanation: Class A defines a static method `age`, and interface B defines a nonstatic method `age` with the same set of method parameters. Because a class can't define methods that only differ in their nonaccess modifiers (static and nonstatic), class C can't inherit class A and implement interface B.



[1.1] Use access modifiers: private, protected, and public



[1.7] Use package and import statements

ME-Q90. Given that classes `Connection` and `SQLConnection` are defined in separate packages

```
package util;
public class Connection {
    protected String url;
    private String port;
    public String pwd;
    String username;
}
package sql;
import util.Connection;
class SQLConnection extends Connection{
    void getDefaultConnection(){
        //line1
    }
}
```

which instance variables of `Connection` can `SQLConnection` access directly (using inheritance) at //line1? (Choose the best option.)

- ☐ a `url`, `port`, `pwd`, and `username`
- ☐ b `url`, `pwd`, and `username`
- ☒ c **`url` and `pwd`**
- ☐ d `pwd`

ME-A90. c

Explanation: The private variable `port` can't be accessed outside class `Connection`. The variable `username`, defined with *default* access, can only be accessed within the package `util`. It can't be accessed by its derived class `SQLConnection`, because they're defined in separate packages. The variable `url` with protected access can be accessed by all derived classes of class `Connection`, irrespective of the package in which the derived classes are defined. So `url` is accessible in class `SQLConnection`. The variable `pwd` with public access can be accessed by all classes in all packages.

OCF Java SE 7

Programmer II Certification Guide

Mala Gupta



The OCP Java 7 certification tells potential employers that you've mastered the language skills you need to design and build professional-quality Java software. Passing the OCP isn't just about knowing your Java, though. You have to also know what to expect on the exam and how to beat the built-in tricks and traps.

OCF Java SE 7 Programmer II Certification Guide is a comprehensive, focused study guide that prepares you to pass the OCP exam the first time you take it. It systematically guides you through each exam objective, reinforcing the Java skills you need through examples, exercises, and cleverly constructed visual aids. In every chapter you'll find questions just like the ones you'll face on the real exam. Tips, diagrams, and review notes give structure to the learning process to improve your retention.

What's Inside

- 100% coverage of the OCP Java SE 7 Programmer II exam (1Z0-804)
- Flowcharts, UML diagrams, and other visual aids
- Hands-on coding exercises
- Focuses on passing the exam, not the Java language itself

Designed for readers with intermediate-level Java skills.

Mala Gupta is a trainer of programmers who plan to pass Java certification exams. She holds the OCP Java SE 7 Programmer, SCWCD, and SCJP certifications and is the author of *OCA Java SE 7 Programmer I Certification Guide* (Manning 2013).

To download their free eBook in PDF, ePub, and Kindle formats, owners of this book should visit
manning.com/ocf-java-se-7-programmer-ii-certification-guide

“A good read for all who want to deepen their Java knowledge, even if not preparing for the exam.”

—Simon Joseph Aquilina
KPMG Crimsonwing

“Makes the certification objectives clear and easy to understand.”

—Mikael Strand, Capgemini

“An excellent resource for the OCP certification exam.”

—Ashutosh Sharma
Discover Financial Services, LLC

“With a conversational style of writing, detailed code examples, and self-test questions, this book will successfully lead you to your OCP certification.”

—Mikalai Zaikin, IBA IT Park

ISBN 13: 978-1-617291-48-7
ISBN 10: 1-617291-48-X
5 4 4 9 9



9 781617 129148 7