socrative

Answer Key
February 24, 2021

OCA Final Assessment pt. 1 of 2 ('Copy')

1.

```
public class Indexing {
   public static void main(String... books) {
      StringBuilder sb = new StringBuilder();
      for (String book : books)
         sb.insert(sb.indexOf("c"), book);
      System.out.println(sb);
   }
}
```

What is the output if this class is run with java Indexing cars carts?

A cars

B cars carts

C ccars arts

D The code does not compile.

✓ E The code compiles but throws an exception at runtime.

i The first time through the loop, we are calling indexOf on an empty StringBuilder. This returns -1. Since we cannot insert at index -1, the code throws a StringIndexOutOfBoundsException.

2.

The operators +=, ___, ___, ____, ____, and ++ are listed in increasing or the same level of operator precedence. (Choose two.)

A -, +, =, --

B %, *, /, +

✓ C =, +, /, *

D ^, *, -, ==

✓ E *, /, %, --

i In Option A, the assignment operator = incorrectly comes after the addition + operator. In Option B, the addition operator + incorrectly comes after the division / operator. In Option D, the subtraction operator - incorrectly comes after the multiplication * operator. This leaves Options C and E as the correct answers. For these answers, it may help to remember that the modulus operator %, multiplication operator *, and division operator / have the same operator precedence.

3. Which of the following are valid JavaBean signatures? (Choose three.)

A public byte getNose(String nose)

✓ B public void setHead(int head)

✓ C public String getShoulders()

D public long isMouth()

E public void gimmeEars()

✓ F public boolean isToes()

i Option A is incorrect because a getter should not take a value. Option D is incorrect because the prefix is should only be with boolean values. Option E is incorrect because gimme is not a valid JavaBean prefix. Options B, C, and F are each proper JavaBean method signatures.

4.

```
20:  int[] crossword [] = new int[10][20];
21:  for (int i = 0; i < crossword.length; i++)
22:     for (int j = 0; j < crossword.length; j++)
23:        crossword[i][j] = 'x';
24:  System.out.println(crossword.size());
```

Which of the following are true? (Choose two.)

✓ A One line needs to be changed for this code to compile.

B Two lines need to be changed for this code to compile.

C Three lines need to be changed for this code to compile.

D If the code is fixed to compile, none of the cells in the 2D array have a value of 0.

✓ E If the code is fixed to compile, half of the cells in the 2D array have a value of 0.

F If the code is fixed to compile, all of the cells in the 2D array have a value of 0.

i Line 24 does not compile because arrays use length. It is ArrayList that uses size(). All of the other lines compile, making Option A correct. It is allowed to split up the braces in the 2D array declaration on line 20. The code is also allowed to use crossword.length as the loop condition on line 22, although this is not a good idea. The array starts out with all 200 of the cells initialized to the default value for an int of 0. Both loops iterate starting at 0 and stopping before 10, which causes only half of the array to be set to 'x'. The other half still has the initial default value of 0, making Option E correct.

5. Which of the following statements about java.lang.Error are most accurate? (Choose two.)

A An Error should be thrown if a file system resource becomes temporarily unavailable.

✓ B An application should never catch an Error.

C Error is a subclass of Exception, making it a checked exception.

✓ D It is possible to catch and handle an Error thrown in an application.

E An Error should be thrown if a user enters invalid input.

i Options A and E are incorrect because they indicate states that the application can possibly recover from. An Error generally indicates an unrecoverable problem. While it is possible to catch an Error, it is strongly recommended that an application never do so, making Options B and D correct. Finally, Option C is incorrect because Error extends from Throwable, not Exception, and is unchecked.

6.

```
import forest.Bird;
import jungle.tree.*;
import savana.*;
```

Given a class that uses the following import statements, which class would be automatically accessible without using its full package name? (Choose three.)

- ✓ **A** forest.Bird
- **B** savana.sand.Wave
- ✓ **C** jungle.tree.Huicungo
- ✓ **D** java.lang.Object
- **E** forest.Sloth
- **F** forest.ape.bonobo

i The first import statement allows only the class forest.Bird to be available, making Option A correct and Options E and F incorrect. Option B is incorrect since the third import statement only allows access to classes within the savana package, not any sub-packages. Option C is correct because the second import statement allows any class in the jungle.tree package to be accessible. Finally, Option D is correct because java.lang.* is implicitly included in all Java classes.

7.

```
ArrayList l = new ArrayList();
String s = new String();
StringBuilder sb = new StringBuilder();
LocalDateTime t = LocalDateTime.now();
```

How many of the following variables represent immutable objects?

- **A** None
- **B** One
- ✓ **C** Two
- **D** Three
- **E** Four
- **F** None of the above—this code doesn't compile.

i *Mutable* means the object can change state. *Immutable* means the object cannot change state. An ArrayList stores a collection of objects. It mutates as the elements change. A StringBuilder is also mutable as it improves performance by not creating a new object each time it changes. A String is immutable. Methods that look like they change the value simply return a different String object. The date/time objects added in Java 8, such as LocalDateTime, are also immutable. Therefore, Option C is correct with String and LocalDateTime as the immutable object types.

8.

```
StringBuilder builder = new StringBuilder("Leaves growing");
do {
   builder.delete(0, 5);
} while (builder.length() > 5);
System.out.println(builder);
```

What is the output of the following?

A Leaves growing

B ing

✓ C wing

D The code does not compile.

E The code compiles but throws an exception at runtime.

i On the first iteration through the loop, the first five characters are removed and builder becomes s growing. Since there are more than five characters left, the loop iterates again. This time, five more characters are removed and builder becomes wing. This matches Option C.

9.

```
package reality;
public class Equivalency {
   public static void main(String[] edges) {
      final String ceiling = "up";
      String floor = new String("up");
      final String wall = new String(floor);
      System.out.print((ceiling==wall)+" "+(floor==wall)+" "+ceiling.equals(wall));
   }
}
```

What is the output of the following application?

A false false false

B true true true

C false true true

✓ D false false true

E It does not compile.

i The code compiles without issue, so Option E is incorrect. The key here is that none of the variables are assigned the same object due to the use of the new keyword. Comparing any two variables with == will always result in an evaluation of false, making the first two values of the print statement be false and false. On the other hand, they all have an underlying String value equivalent to up, so calling equals() on any two variables will return true. Option D is the correct answer that matches what the application will print.

10.

```
1:  public class Giggles {
2:     public static void main(String[] args) {
3:        String lol = "lol";
4:        System.out.println(lol.toUpperCase() == lol);
5:        System.out.println(lol.toUpperCase() == lol.toUpperCase());
6:        System.out.println(lol.toUpperCase().equals(lol));
7:        System.out.println(lol.toUpperCase().equals(lol.toUpperCase()));
8:        System.out.println(lol.toUpperCase().equalsIgnoreCase(lol));
9:        System.out.println(lol.toUpperCase().equalsIgnoreCase(lol.toUpperCase()));
10:    }
11: }
```

How many times does the following code print true?

- A One
- B Two
- ✓ C Three
- D Four
- E Five
- F None—the code does not compile.

i Lines 4 and 5 both print false since a String should be compared with a method rather than ==, especially when not comparing two values from the string pool. Line 6 also prints false because one value is uppercase and the other is lowercase. Line 7 prints true because both values are uppercase. Lines 8 and 9 print true because they don't look at the case. This makes Option C the answer.

11.

```
14:  String race = "";
15:  outer:
16:  do {
17:  inner:
18:     do {
19:        race += "x";
20:     } while (race.length() <= 4);
21:  } while (race.length() < 4);
22: System.out.println(race);
```

Which lines can be removed together without stopping the code from compiling and while printing the same output? (Choose three.)

- ✓ A Lines 15 and 17
- ✓ B Lines 15, 16, and 21
- ✓ C Line 17
- D Lines 17, 18, and 20
- E Line 20
- F Line 21

i Let's look at each one in turn. Option A is correct because the labels are not referenced. Option B is correct because the outer while is broader than the inner while. Since there is no other code in the loop, it is not needed. Option C is also correct because a label is not used. Option D is incorrect because the inner loop is more specific than the outer loop. While the code still compiles, it prints one less chapter. Options E and F are incorrect because you cannot remove one half of a loop construct and have it compile.

12. `long bigNum = _____;`

Which of the following do not compile when filling in the blank? (Choose two.)

- A 1234
- ✓ B 1234.0
- ✓ C 1234.0L
- D 1234l
- E 1234L
- F 1_234

i A long cannot contain a number with decimal points, preventing Options B and C from compiling. Options D and E show you can force a number to be a long by ending it with an upper- or lowercase L. This does not work if the number has a decimal point. Option F shows how to use underscores to break up a number.

13.

```
import java.time.*;
public class OnePlusOne {
   public static void main(String... nums) {
      LocalTime time = LocalTime.of(1, 11);
      while (time.getHour() < 1) {
         time.plusHours(1);
         System.out.println("in loop");
      }
   }
}
```

How many lines does this program print?

- ✓ A None
- B One
- C Two
- D This is an infinite loop.
- E The code does not compile.

i A while loop checks the condition before executing. Since the hour is not less than one, the loop never enters, and Option A is correct. This is good, because we'd have an infinite loop if the loop was entered since the result of plusHours is ignored.

14.

```
1:   package fun;
2:   public class Sudoku {
3:      static int[][] game;
4:
5:      public static void main(String args[]) {
6:         game[3][3] = 6;
7:         Object[] obj = game;
8:         obj[3] = 'X';
9:         System.out.println(game[3][3]);
10:     }
11:  }
```

What is the result of running the following program?

A 6

B X

C The code does not compile.

✓ D The code compiles but throws a NullPointerException at runtime.

E The code compiles but throws a different exception at runtime.

F The output is not guaranteed.

i This question appears to ask you about involved array logic. Instead, it is checking to see if you remember that instance and class variables are initialized to null. Line 6 throws a NullPointerException. If the array was declared, the answer would be E because the code would throw an ArrayStoreException on line 8.

15. Which of the following use generics and compile without warnings? (Choose two.)

A List<String> a = new ArrayList();

B List<> b = new ArrayList();

✓ C List<String> c = new ArrayList<>();

D List<> d = new ArrayList<>();

✓ E List<String> e = new ArrayList<String>();

F List<> f = new ArrayList<String>();

i The diamond operator is only allowed to be used when instantiating rather than declaring. In other words, it can't go on the left side of the equal sign. Therefore, Options B, D, and F are incorrect. The remaining three options compile. However, Option A produces a warning because generics are not used on the right side of the assignment operator. Therefore, Options C and E are correct. Option C is better than Option E since it uses the diamond operator rather than specifying a redundant type.

16.

```
public static void main(String[] args) {
   String shoe1 = new String("sandal");
   String shoe2 = new String("flip flop");
   String shoe3 = new String("croc");

   shoe1 = shoe2;
   shoe2 = shoe3;
   shoe3 = shoe1;
}
```

Which of the following are true right before the main() method ends? (Choose two.)

- A No objects are eligible for garbage collection.
- ✓ B One object is eligible for garbage collection.
- C Two objects are eligible for garbage collection.
- ✓ D No objects are guaranteed to be garbage collected.
- E One object is guaranteed to be garbage collected.
- F Two objects are guaranteed to be garbage collected.

i At the end of the method, shoe1 and shoe3 both point to "flip flop". shoe2 points to "croc". Since there are no references to "sandal", it is eligible for garbage collection, making Option B correct. However, garage collection is not guaranteed to run, so Option D is also correct.

17. How many lines of the following application do not compile?

- A None. The code compiles and prints swim!.
- B None. The code compiles and prints a stack trace.
- ✓ C One
- D Two
- E Three

```
package ocean;
class BubbleException extends Exception {}
class Fish {
   Fish getFish() throws BubbleException {
      throw new RuntimeException("fish!");
   }
}
public final class Clownfish extends Fish {
   public final Clownfish getFish() {
      throw new RuntimeException("clown!");
   }
   public static void main(String[] bubbles) {
      final Fish f = new Clownfish();
      f.getFish();
      System.out.println("swim!");
   }
}
```

i The code does not compile, so Options A and B are incorrect. The getFish() method is declared properly in the Fish class and successfully overridden in the Clownfish class. An overridden method must not declare any new or broader checked exceptions, but it is allowed to declare narrower exceptions or drop checked exceptions. The overridden method also uses a covariant return type. The use of final on the method and class declarations has no meaningful impact, since the methods and classes are not extended in this application. So where does the compilation error occur? In the main() method! Even though the Clownfish version of getFish() does not declare a checked exception, the call f.getFish() uses a Fish reference variable. Since the Fish reference variable is used and that version of the method declares a checked Exception, the compiler enforces that the checked exception must be handled by the main() method. Since this checked exception is not handled with a try-catch block nor by the main() method declaration, the code does not compile, and Option C is the correct answer.

18.

```
import java.util.*;
import java.util.function.*;

public class PrintNegative {

  public static void main(String[] args) {
     List<Integer> list= new ArrayList<>();
     list.add(-5);
     list.add(0);
     list.add(5);
     print(list, e -> e < 0);
  }

   public static void print(List<Integer> list, Predicate<Integer> p) {
      for (Integer num : list)
         if (p.test(num))
            System.out.println(num);
  }
}
```

How many lines does this code output?

✓ **A** One

B Two

C Three

D None. It doesn't compile.

E None. It throws an exception at runtime.

i This is a correct example of using lambdas. The code creates an ArrayList with three elements. The print() method loops through and checks for negative numbers. Option A is correct.

19. Which keywords are required with a try statement?
I. finalize
II. catch
III. throws
IV. finally

A I only

B II only

C III only

D IV only

E I or II, or both

✓ **F** None of the above

i A try statement requires a catch or a finally block. It can also have both a catch and a finally block. Since no option matches these rules, Option F is the correct answer. Note that finalize is not a keyword but a method inherited from java.lang.Object. Lastly, the throws keyword can be applied to method declarations and is not used as part of a try statement.

20.

```
12:  int result = 8;
13:  loop: while (result > 7) {
14:     result++;
15:     do {
16:        result--;
17:     } while (result > 5);
18:     break loop;
19:  }
20:  System.out.println(result);
```

What is the output of the following?

✓ **A** 5

B 7

C 8

D The code does not compile.

E The code compiles but throws an exception at runtime.

i On line 12, result is first set to 8. On line 13, the boolean condition is true because 8 > 7. On line 13, result is incremented to 9. Then the inner loop runs, decrementing result until it is no longer greater than 5. On line 18, loop execution is completed because result is equal to 5. The break statement says to skip to after the labeled loop, which is line 20. Then result is printed as 5, making Option A correct.

21. What is the result of compiling and executing the following application?

```
package reptile;
public class Alligator {
   static int teeth;
   double scaleToughness;
   public Alligator() {
      teeth++;
   }
   public void snap(int teeth) {
      System.out.print(teeth+" ");
      teeth--;
   }
   public static void main(String[] unused) {
      new Alligator().snap(teeth);
      new Alligator().snap(teeth);
   }
}
```

A 0 1

B 1 1

✓ **C** 1 2

D 2 2

E The code does not compile.

F The code compiles but produces an exception at runtime.

i The code compiles and runs without exception, making Options E and F incorrect. The question is testing your knowledge of variable scope. The teeth variable is static in the Alligator class, meaning the same value is accessible from all instances of the class, including the static main() method. The static variable teeth is incremented each time the constructor is called. Since teeth is a local variable within the snap() method, the argument value is used, but changes to the local variable do not affect the static variable teeth. Since the local variable teeth is not used after it is decremented, the decrement operation has no meaningful effect on the program flow or the static variable teeth. Since the constructor is called twice, with snap() executed after each constructor call, the output printed is 1 2, making Option C the correct answer.

22.

```
public class Costume {
   public static void main(String[] black) {
      String witch = "b";
      String tail = "lack";
      witch.concat(tail);
      System.out.println(witch);
   }
}
```

What is the output of the following?

✓ **A** b

B black

C lack

D The code does not compile.

E The code compiles but throws an exception at runtime.

i A String is immutable. Since the result of the concat() method call is ignored, the value of witch never changes. It stays as a single letter, and Option A is correct.

23. Which modifiers can be independently applied to an interface method? (Choose three.)

✓ **A** default

B protected

✓ **C** static

D private

E final

✓ **F** abstract

i An interface method is exactly one of three types: default, static, or abstract. For this reason, Options A, C, and F are correct. An interface method cannot be protected nor private because the access modifier is always public, even when not specified, making Options B and D incorrect. Option E is also incorrect because final cannot be applied to static methods, since they cannot be overridden. It can also not be applied to default and abstract methods because they are always able to be overridden.

24.

```
public class Shoelaces {
   public static void main(String[] args) {
      String tie = null;
      while (tie = null)
         tie = "shoelace";
         System.out.print(tie);
   }
}
```

What is the output of the following?

- A null
- B shoelace
- C shoelaceshoelace
- ✓ D The code does not compile.
- E This is an infinite loop.
- F The code compiles but throws an exception at runtime.

i Look at the loop condition carefully. It tries to assign null to a String variable. This is not an expression that returns a boolean. Therefore, the code does not compile, and Option D is correct. If this was fixed by making the loop condition tie == null, then Option B would be correct.

25. What statements are true about compiling a Java class file? (Choose two.)

- A If the file does not contain a package statement, then the compiler considers the class part of the java.lang package.
- ✓ B The compiler assumes every class implicitly imports the java.lang.* package.
- C The compiler assumes every class implicitly imports the java.util.* package.
- D Java requires every file to declare a package statement.
- E Java requires every file to declare at least one import statement.
- ✓ F If the class declaration does not extend another class, then it implicitly extends the java.lang.Object class.

i A class may be defined without a package statement, making the class part of the default package. For this reason, Options A and D are incorrect. Every Java class implicitly imports exactly one package, java.lang.*, making Option B correct and Option C incorrect. Option E is incorrect because an import statement is not required. Finally, Option F is correct; any class that does not extend another class implicitly extends java.lang.Object.

26.

```
package woods;
interface Plant {
   default String grow() { return "Grow!"; }
}
interface Living {
   public default String grow() { return "Growing!"; }
}
public class Tree implements Plant, Living {  // m1
   public String grow(int height) { return "Super Growing!"; }
   public static void main(String[] leaves) {
      Plant p = new Tree();  // m2
      System.out.print(((Living)p).grow());  // m3
   }
}
```

What is the output of the following application?

A Grow!

B Growing!

C Super Growing!

✓ D It does not compile because of line m1.

E It does not compile because of line m2.

F It does not compile because of line m3.

i A class cannot inherit two interfaces that declare the same default method, unless the class overrides them. In this case, the version of grow() in the Tree class is an overloaded method, not an overridden one. Therefore, the code does not compile due to the declaration of Tree on line m1, and Option D is the correct answer.

27.

```
public static void main(String... args) {
   String name = "Desiree";
   int _number = 694;
   boolean profit$$$;
   System.out.println(name + " won. "
       + _number + " profit? " + profit$$$);
}
```

What is the result of the following?

A The declaration of name does not compile.

B The declaration of _number does not compile.

C The declaration of profit$$$ does not compile.

✓ D The println statement does not compile.

E The code compiles and runs successfully.

F The code compiles and throws an exception at runtime.

i Variables are allowed to start with an underscore and are allowed to contain a $. Therefore, all the variable declarations compile, making Options A, B, and C incorrect. However, the println() refers to the uninitialized local boolean. Since local variables are not automatically initialized, the code does not compile, and Option D is correct.

28. Given a variable x, _____ decreases the value of x by 1 and returns the original value, while _____ increases the value of x by 1 and returns the new value.

- ✓ **A** x--, ++x
- **B** x--, x++
- **C** --x, x++
- **D** --x, ++x

i Prefix operators, such as --x and ++x, modify the variable and evaluate to the new value, while postfix operators, such as x-- and x++, modify the variable but return the original value. Therefore, Option A is the correct answer.

29. Given the following two classes in the same package, which constructors contain compiler errors? (Choose three.)

```
public class Big {
   public Big(boolean stillIn) {
      super();
   }
}

public class Trouble extends Big {
   public Trouble()  {}
   public Trouble(int deep) {
      super(false);
      this();
   }
   public Trouble(String now, int... deep) {
      this(3);
   }
   public Trouble(long deep) {
      this("check",deep);
   }
   public Trouble(double test) {
      super(test>5 ? true : false);
   }
}
```

- **A** public Big(boolean stillIn)
- ✓ **B** public Trouble()
- ✓ **C** public Trouble(int deep)
- **D** public Trouble(String now, int... deep)
- ✓ **E** public Trouble(long deep)
- **F** public Trouble(double test)

i The constructors declared by Options A, D, and F compile without issue. Option B does not compile. Since there is no call to a parent constructor or constructor in the same class, the compiler inserts a no-argument super() call as the first line of the constructor. Because Big does not have a no-argument constructor, the no-argument constructor Trouble() does not compile. Option C also does not compile because super() and this() cannot be called in the same constructor. Note that if the super() statement was removed, it would still not compile since this would be a recursive constructor call. Finally, Option E does not compile. There is no matching constructor that can take a String followed by a long value. If the input argument deep was an int in this constructor, then it would match the constructor used in Option D and compile without issue.

30.

```
public class Stats {
   // INSERT CODE
   public static void main(String[] math) {
     System.out.println(max - min);
   }
}
```

Which of the following can replace the comment so this code outputs 100? (Choose two.)

- A final int min, max = 100;
- B final int min = 0, max = 100;
- C int min, max = 100;
- D int min = 0, max = 100;
- ✓ E static int min, max = 100;
- ✓ F static int min = 0, max = 100;

i A static method is not allowed to access instance variables without an instance of the class, making Options E and F correct. Notice that only max is initialized to 100 in Option E. Since min doesn't have a value specified, it gets the default value, which is 0.

31. Which of the following statements are true about Java operators and statements? (Choose two.)

- A Both right-hand sides of the ternary expression will be evaluated at runtime.
- ✓ B A switch statement may contain at most one default statement.
- C A single if-then statement can have multiple else statements.
- D The | and || operator are interchangeable, always producing the same results at runtime.
- ✓ E The ! operator may not be applied to numeric expressions.

i The ternary ? : operator only evaluates one of the two right-hand expressions at runtime, so Option A is incorrect. A switch statement may contain at most one optional default statement, making Option B correct. A single if-then statement can have at most one else statement, so Option C is incorrect. Note that you can join if-then-else statements together, but each else requires an additional if-then statement. The disjunctive | operator will always evaluate both operands, while the disjunctive short-circuit || operator will only evaluate the right-hand side of the expression if the left-hand side evaluates to false. Therefore, they are not interchangeable, especially if the right-hand side of the expression modifies a variable. For this reason, Option D is incorrect. Finally, Option E is correct. The logical complement ! operator may only be applied to boolean expressions, not numeric ones.

32.

```
1:   public class Legos {
2:      public static void main(String[] args) {
3:         StringBuilder sb = new StringBuilder();
4:         sb.append("red");
5:         sb.deleteCharAt(0);
6:         sb.delete(1, 1);
7:         System.out.println(sb);
8:      }
9:   }
```

What is the output of the following?

A r

B e

✓ C ed

D red

E The code does not compile.

F The code compiles but throws an exception at runtime.

i Line 3 creates an empty StringBuilder. Line 4 adds three characters to it. Line 5 removes the first character resulting in ed. Line 6 deletes the characters starting at position 1 and ending right before position 1. Since there are no indexes that meet that description, the line has no effect. Therefore, Option C is correct.

33. Which of the following is a valid method name in Java? (Choose two.)

✓ A ____()

B %run()

C check-Activity()

✓ D $Hum2()

E sing\\3()

F po#ut ()

i Java methods must start with a letter, the dollar $ symbol, or the underscore _ character. For this reason, Option B is incorrect, and Options A and D are correct. Despite how Option A looks, it is a valid method signature in Java. Options C, E, and F do not compile because the symbols -, \, and # are not allowed in method names, respectively.

34. Which of the following statements about inheritance are true? (Choose two.)

A Inheritance is better than using static methods for accessing data in other classes.

✓ B Inheritance allows a method to be overridden in a subclass, possibly changing the expected behavior of other methods in a superclass.

✓ C Inheritance allows objects to inherit commonly used attributes and methods.

D It is possible to create a Java class that does not inherit from any other.

E Inheritance tends to make applications more complicated.

i First off, Option A is incorrect, since whether or not static or inherited methods are chosen is a matter of design and individual preference. Options B and C are true statements about inheritance and two of the

most important reasons Java supports inheritance. Option D is incorrect because all Java classes extend java.lang.Object. Option E is incorrect. Whether or not inheritance simplifies or complicates a design is based on the skills of the developer creating the application.

35. Which of the following statements about Java are true?
I. The java command uses . to separate packages.
II. Java supports functional programming.
III. Java is object oriented.
IV. Java supports polymorphism.

- A I only
- B II only
- C II and III.
- D I, III, and IV
- ✓ E I, II, III, and IV
- F None are true.

i All of the statements are true statements about Java, making Option E the correct answer. Java was built with object-oriented programming and polymorphism in mind. Also, Java supports functional programming using lambda expressions.

36.
```
String[][] listing = new String[][] { { "Book", "34.99" },
   { "Game", "29.99" }, { "Pen", ".99" } };
System.out.println(listing.length + " " + listing[0].length);
```

What is the output of the following?

- A 2 2
- B 2 3
- ✓ C 3 2
- D 3 3
- E The code does not compile.
- F The code compiles but throws an exception at runtime.

i This array has three elements, making listing.length output 3. It so happens that each element references an array of the same size. But the code checks the first element and sees it is an array of size two, making the answer Option C.

37. Which of the following variable types is permitted in a switch statement? (Choose three.)

- ✓ **A** Character
- ✓ **B** Byte
- **C** Double
- **D** long
- ✓ **E** String
- **F** Object

i A switch statement supports the primitive types byte, short, char, and int and their associated wrapper classes Character, Byte, Short, and Integer. It also supports the String class and enumerated types. Floating-point types like float and double are not supported, nor is the Object class. For these reasons, Options A, B, and E are correct.

38.

```
public class Shoot {
   interface Target {
      boolean needToAim(double angle);
   }
   static void prepare(double angle, Target t) {
      boolean ready = t.needToAim(angle);  // k1
      System.out.println(ready);
   }
   public static void main(String[] args) {
      prepare(45, d => d > 5 || d < -5);   // k2
   }
}
```

What does the following do?

- **A** It prints true.
- **B** It prints false.
- **C** It doesn't compile due to line k1.
- ✓ **D** It doesn't compile due to line k2.
- **E** It doesn't compile due to another line.

i The lambda syntax is incorrect. It should be →, not =>. Therefore, Option D is correct. If this was fixed, Option A would be correct.

39. Which of the following is a valid code comment in Java? (Choose three.)

- **A** /** Insert */ in next method **/
- ✓ **B** /****** Find the kitty cat */
- ✓ **C** // Is this a bug?
- **D** / Begin method - performStart() /
- ✓ **E** /*** TODO: Call grandma ***/
- **F** # Updated code by Patti

i The /* */ syntax can have additional (and uneven) * characters in Java, making Options B and E correct. Option C is the standard way to comment a single line with two slashes //. Option A contains a */ in the middle of the expected comment, making the part after the comment Insert **/ invalid. Option D is

incorrect because a single slash / is not valid comment in Java. Finally, the # is not a comment character in Java, so Option F is incorrect.

40.

```
package food;
public class Grass {
   public static int seeds = 10;
   public static Grass getGrass() {return new Grass();}
}

package woods;
// INSERT CODE HERE
public class Deer {
   public void eat() {
      getGrass();
      System.out.print(seeds);
   }
}
```

Given the following two classes, each in a different package, which lines allow the second class to compile when inserted independently? (Choose two.)

✓ **A** import static food.Grass.getGrass;
import static food.Grass.seeds;

B import static food.*;

C static import food.Grass.*;

D import food.Grass.*;

E static import food.Grass.getGrass;
static import food.Grass.seeds;

✓ **F** import static food.Grass.*;

i A static import is used to import static members of another class. Option A is correct because the method getGrass and variable seeds are imported. Option F is also correct because a wildcard on the Grass class for all visible static members is allowed. Option B is incorrect because the wildcard must be on a class, not a package. Options C and E are incorrect since the keywords import and static are reversed. Option D is incorrect because the static keyword is missing.