Java Software Solutions, 9e (Lewis/Loftus)
Chapter 3 Using Classes and Objects

TRUE/FALSE

1. Only difficult programming problems require a pseudocode solution before the programmer creates the program itself.

   ANS: F
   Some form of design should be created for all problems before the program is written, with the possible exception of only the most trivial of problems. The form of the design may vary. For instance, programmers in the 1960s and 1970s often used flowcharts instead of pseudocode.

2. You may apply the prefix and postfix increment and decrement operators to instances of the `Integer` class.

   ANS: F
   These operators may only be applied to numeric data. An instance of a class is an object and and not a piece of primitive data.

3. In Java, the symbol `"="` and the symbol `"=="` are used interchangeably.

   ANS: F
   `"="` is used for assignment statements while `"=="` is used to test equality in conditional statements.

4. When comparing any primitive type of variable, the == operator should always be used to test if two values are equal.

   ANS: F
   This is true of `int`, `short`, `byte`, `long`, `char` and `boolean`, but not of `double` or `float` variables. If two `double` variables, `x` and `y`, are being tested, (`x == y`) is `true` only if they are exactly equal to the last decimal point. It is common instead to compare these two values but allow for a small difference in value. For instance, if `THETA = 0.000001`, we might test `x` and `y` by (`x - y <= THETA`) instead of `(x == y)` to get a better idea of whether they are close enough to be considered equal.

5. These two ways to set up a `String` will yield identical results:

   ```
   String my_string = "12345";
   String my_string = 12345;
   ```

   ANS: F
   In fact, the second statement won't even compile! The second statement will receive a syntax error about incompatible types. There is no automatic conversion from a number to a `String`.

6. These two ways to set up a `String` will yield identical results:

   ```
   String my_string = new String("a bunch of words");
   ```

```
String my_string = "a bunch of words";
```

ANS: T
Java has a very strong notion about what `Strings` are and how they function. The second declaration is merely shorthand for the first. The string reference will be initialized identically by both declarations.

7. These two ways to set up a `String` will yield identical results:

```
String my_string = new String("123.45");
String my_string = "" + 123.45;
```

ANS: T
Java understands the + operator when used to combine `Strings` with numbers means that the number should be converted into a numeric `String`, and then concatenation should occur.

8. These two ways to set up a `String` will yield identical results:

```
String my_string = new String(12345);
String my_string = new String("12345");
```

ANS: F
In fact, the first statement won't even compile! There is no overloaded version of the `String` constructor that accepts a numeric argument.

9. The following statement will display the value `127`.

```
System.out.println("123" + 4);
```

ANS: F
`"123"` is a `String`. So, the + will cause `String` concatenation to occur. The `int 4` will be converted into the `String "4"`, and the `String "1234"` will be created and displayed.

10. You may use the `String.replace()` method to remove characters from a `String`.

ANS: F
The `replace()` method only replaces single characters with other single characters. The `replace()` method will not add or delete characters to a `String`; the `String` length will remain the same.

11. If you need to import not only the top-level of a package, but all its secondary levels as well, you should use the following:

```
import package.*.*;
```

ANS: F
The `import` statement can only be used with a single `*` (wild card). If you need to import all the secondary levels of a package as well, you must write them out explicitly:

```
import package.A.*;
import package.B.*;
```

12. All the methods in the `Math` class are declared to be `static`.

ANS: T
The `Math` class methods are designed to be generally useful in arithmetic expressions, so no instances of anything are required to make use of them. This is achieved by ensuring that all the `Math` methods are `static`.

13. The `printf` method within `System.out` is designed to ease the conversion of legacy C code into Java.

ANS: T
C programs use the C `printf` function for output. Java's `printf` method is modeled closely after the C `printf` function, so C output statements can be translated into Java extremely easily.

14. The names of the wrapper classes are just the names of the primitive data types, but with an initial uppercase letter.

ANS: F
This is true for most of the wrapper classes, but it is false for `int` (Integer) and `char` (Character).

MULTIPLE CHOICE

1. In Java a variable may contain

   a. a value or a reference
   b. a package
   c. a method
   d. a class
   e. Any of these

   ANS: A
   Java variables contain values or references to instances of classes (which contain values and/or additional references).

2. If two variables contain aliases of the same object then

   a. the object may be modified using either alias
   b. the object cannot be modified unless there's a single reference to it
   c. a third alias is created if/when the object is modified
   d. the object will become an "orphan" if both variables are set to `null`
   e. answers A and D are both correct

   ANS: E
   By definition, an alias provides a means by which an object can be modified (an alias is like a pointer). If both variables are set to `null`, then the object is not referenced by any variable (via any alias) and it does, indeed, become an "orphan" and it will be garbage collected at some point in the future.

3. Which properties are true of `String` objects?

   a. Their lengths never change.
   b. The shortest `String` has zero length.
   c. Individual characters within a `String` may be changed using the `replace` method.
   d. The index of the first character in a `String` is one.
   e. Only answers A and B are true.

   ANS: E
   `Strings` are immutable. That means that once a `String` object is created it cannot be changed. Therefore the length of a `String` never changes once it has been created. The shortest length `String` is "". Since there are no characters between the quotes, the length is zero. The `replace` method allows you to create a new `String` from an original one, replacing some of the characters. The index of the first location in a `String` is zero, not one. Also, the last byte of every `String` contains the end-of-string character which is a `byte` of low-values, or binary zeros.

4. What happens if you attempt to use a variable before it has been initialized?

   a. A syntax error may be generated by the compiler.
   b. A run-time error may occur during execution.
   c. A "garbage" or "uninitialized" value will be used in the computation.
   d. A value of zero is used.
   e. Only answers A and B are correct.

   ANS: E
   Many times the compiler is able to detect the attempted use of an uninitialized variable and it will generate a syntax error in this case. If such a use escapes detection by the compiler then a run-time error occurs upon usage. Java is a very "safe" language, so it does not allow "garbage" or zero to be used if an uninitialized variable is used in a computation.

5. What is the function of the dot operator?

   a. It serves to separate the integer portion from the fractional portion of a floating-point number.
   b. It allows one to access the data within an object when given a reference to the object.
   c. It allows one to invoke a method within an object when given a reference to the object.
   d. It is used to terminate commands, similar to the way a period terminates a sentence in English.
   e. Both B and C are correct.

   ANS: E
   The dot operator is appended directly after the object reference, followed by the data or method to which access is desired. In the case of data, the access may be for reading or writing. In the case of a method, the access is to allow one to invoke the method. The dot within a floating-point number is a decimal point and not a dot operator.

6. In Java, instantiation means

   a. noticing the first time something is used
   b. creating a new object of the class
   c. creating a new alias for an existing object
   d. launching a method
   e. none of these

ANS: B
Instantiation is the process of creating a new instance of an object. This usually is accomplished by using the `new` operator. In the case of `Strings`, new instances (instantiation) may be created just by using quotes in an expression. For example:

```
String s;
s = "A new string (instance)";
```

7. Assume you write a program that uses the `Random` class but you fail to include an `import` statement for `java.util.Random` or `java.util.*`. What will happen when you attempt to compile and run your program?

    a. The program won't run but it will compile with a warning about missing the class.
    b. The program won't compile and you'll receive a syntax error about the missing class.
    c. The program will compile but you'll receive a warning about the missing class.
    d. The program will encounter a run-time error when it attempts to access any member of the `Random` class.
    e. none of these

    ANS: B
    The missing class means that there will be undefined variables and/or methods. The compiler will detect these and will issue error messages. Your program won't be executable.

8. In the `StringMutation` program shown in Listing 3.1, if `phrase` is initialized to `"Einstein"`, what will `Mutation # 3` yield if `mutation1 = phrase.concat(".")`?

    a. `Einstein.`
    b. `EINSTEIN.`
    c. `XINSTXIN.`
    d. `einstein.`
    e. `xinstxin.`

    ANS: C
    `Mutation #2` changes every letter to upper case. Then `Mutation #3` replaces `E`s with `X`s.

9. Which of the following will yield a pseudorandom number in the range `[-5, +5)`, given:

    ```
    Random gen = new Random();
    ```

    a. `gen.nextFloat()*5`
    b. `gen.nextFloat()*10-5`
    c. `gen.nextFloat()*5-10`
    d. `gen.nextInt()*10-5`
    e. `gen.nextInt(10)-5`

    ANS: B
    `nextFloat` yields a pseudorandom number in the range `[0, 1)`; multiplying by `10` yields numbers in the range `[0, 10)`; subtracting `5` yields numbers in the range `[-5, 5)`.

10. What will be displayed by the following command?

    ```
    System.out.println(Math.pow(3, 3-1));
    ```

    a. `9`
    b. `8`
    c. `6`

d. `4`
e. `27`

ANS: A
This evaluates to `Math.pow(3, 2)` which is `3^2` which is `9`

11. Given the following two lines of code, what can be said about `s1` and `s2`?

```
String s1 = "testing" + "123";
String s2 = new String("testing 123");
```

a. `s1` and `s2` are both references to the same `String` object.
b. the line declaring `s1` is legal in Java; the line declaring `s2` will produce a syntax error.
c. `s1` and `s2` are references to different `String` objects.
d. `s1` and `s2` will compare "equal."
e. None of the above

ANS: C
Both declarations are legal Java. `s1` is a `String` reference, and it is initialized to the `String` `"testing123"`. s2 is a `String` reference, and it is initialized to the `String` `"testing 123"`. Note the space between the `"testing"` and the `"123"`. So the two `Strings` are not equal.

12. An alias is when

a. two different reference variables refer to the same physical object
b. two different numeric variables refer to the same physical object
c. two different numeric variables contain identical values
d. two variables have the same name
e. None of these

ANS: A
An "alias" occurs when there are two or more references to the same physical object so that by following either reference, one can read/write/modify the object.

13. An API is

a. an Abstract Programming Interface
b. an Application Programmer's Interface
c. an Application Programming Interface
d. an Abstract Programmer's Interface
e. an Absolute Programming Interface

ANS: C
An API is an Application Programming Interface. This is a collection of related classes that have been built for specific purposes. APIs usually reside in a class library.

14. `Java.text`'s `NumberFormat` class includes methods that

a. allow you to format currency
b. allow you to format percentages
c. round their display during the formatting process
d. truncate their display during the formatting process
e. A, B, C, but not D

ANS: E

`NumberFormat` always rounds; it never truncates. And it does provide methods for both currency and percentages.

15. The advantages of the `DecimalFormat` class compared to the `NumberFormat` class include

    a. precise control over the number of digits to be displayed
    b. control over the presence of a leading zero
    c. the ability to truncate values rather than round them
    d. the ability to display a percent sign (%) automatically at the beginning of the display
    e. only A and B

    ANS: E
    While `DecimalFormat` does provide far more control than `NumberFormat`, truncation remains in the hands of the programmer via one or more of the `Math` methods. The % symbol would appear at the end of the display, not the beginning.

16. The advantage(s) of the `Random` class's pseudorandom number generators, compared to the `Math.random` method, is(are) that

    a. you may create several random number generators
    b. the generators in `Random` are more efficient than the one in `Math.random`
    c. you can generate random `ints`, `floats`, and `ints` within a range
    d. you can initialize and reinitialize `Random` generators
    e. all except B are true

    ANS: E
    The efficiency of all the random number generators are the same. The advantages of `Random` generators over `Math.random` include all the other properties.

17. The `String` class's `compareTo` method

    a. compares two strings in a case-independent manner
    b. yields `true` or `false`
    c. yields `0` if the two strings are identical
    d. rerturns `1` if the first string comes lexically before the second string
    e. None of these

    ANS: C
    `compareTo` yields `0` if the two strings are identical, `-1` if the first string comes lexically before the second, `+1` if the first string comes lexically after the second.

18. Which statement is true, given the following code fragment?

```
String strA = "aBcDeFg";
String strB = strA.toLowerCase();
strB = strB.toUpperCase();
string strC = strA.toUpperCase();
```

    a. `strB.equals(strC)` would be `true`
    b. `strB.compareTo(strC)` would yield `0`
    c. `strA.equals(strC)` would be `true`
    d. `strA.compareTo(strC)` would yield `0`
    e. None of these

ANS: B
`strB` contains the uppercase forms of all the letters in `strA`; so does `strC`. So `compareTo` would yield `0`, indicating equality.

19. In addition to providing a mechanism to convert primitive data into objects, what else do the wrapper classes provide?

a. enumerations
b. `static` constants
c. arrays to contain the data
d. exceptions
e. None of these

ANS: B
The wrapper classes also provide `static` constants, like `MIN_VALUE` and `MAX_VALUE` (the smallest and largest `ints`).

20. Which is now the preferred approach for developing Java programs that use graphics and GUIs?

a. Swing
b. AWT
c. JavaFX
d. API
e. All of these are preferred approaches

ANS: C
The JavaFX API has now replaced the AWT and Swing for developing graphical programs.

21. In a development environment that fully supports JavaFX, which of the following is true?

a. The `launch` method of the `scene` class is called automatically.
b. The `launch` method of the `Application` class is called automatically.
c. Since the `launch` method is called automatically, you do not need to write the `main` method.
d. Since the `launch` method is called automatically, you do not need to call it in the `main` method.
e. Both B and C are true

ANS: E

22. Which of the following packages includes classes that represent shapes in JavaFX?

a. `javafx.scene.shape`
b. `javafx.shape.scene`
c. `javafx.shapes`
d. `javafx.stage.shape`
e. `javafx.stage.scene`

ANS: A

23. Write a declaration for a `Rectangle` named `squareShape` that is 400 pixels wide, 400 pixels high, and its upper-left corner position is at point (50, 50).

a. `Rectangle squareShape = new squareShape(50, 50, 400, 400);`

b. `squareShape = new Rectangle(50, 50, 400, 400);`
c. `Rectangle squareShape = new Rectangle(50, 400, 50, 400);`
d. `Rectangle squareShape = new Rectangle(50, 50, 400, 400);`
e. `Rectangle = new squareShape(400, 400, 50, 50);`

ANS: D

24. What does the following code fragment do?

```
Circle circle = new Circle(100, 100, 50);
circle.setFill(Color.GREEN);
Rectangle rectangle = new Rectangle(70, 70, 225, 75);
rectangle.setStroke(Color.BLUE);
rectangle.setStrokeWidth(2);
rectangle.setFill(null);
```

a. It creates a green circle, centered at coordinates (100, 100).
b. It creates a rectangle which overlaps the circle since its upper-left corner is at coordinates (70, 70) and its dimensions are 225 X 75.
c. Since the rectangle's fill color is set to `null`, the circle will be visible through the rectangle.
d. The rectangle will have a blue border.
e. All of these are true.

ANS: E
Shapes are drawn in the order they are added to a container such as a group or pane. So, to make one shape appear in front of another, it should be added after it. If the fill color of the rectangle is set to `null`, anything behind it will show through.

PROBLEM

1. Rewrite the following five assignment statements into a single statement using prefix and postfix increment and decrement operators as necessary. Assume all variables are `int` variables.

```
x = y + 1;
y = y + 1;
z = z - 1;
x = x - z;
x = x + 1;
```

ANS:
```
x = (y++ + 1) - (--z) + 1;
```

2. Consider the condition `(x == y)`. How is this handled if `x` and `y` are primitive types? How is this handled if `x` and `y` are objects?

ANS:
If `x` and `y` are primitive types, then the values stored in them are compared and `true` is returned if the two values are equal. If they are objects, then the object that each references are compared. If they are the same object, it returns `true`, otherwise it returns `false`.

3. Given two `String` variables, `s1` and `s2`, is it possible for `(s1 != s2)` to be `true` while `(s1.equals(s2))` is also `true`? Why or why not?

   ANS:
   The condition `(s1 != s2)` means that `s1` and `s2` are references to two distinct objects. It says nothing about the contents of the referenced objects. So, the contents of `s1` may well be identical or different from the contents of `s2`. Therefore it is quite possible that `(s1.equals(s2))` is `true` while `(s1 != s2)` is also `true`.

4. What is a wrapper class? Why are wrapper classes useful?

   ANS:
   A wrapper class is a class that allows you to embed one piece of primitive data within an object. There are methods for extracting the data from the object. Wrapper classes are useful where you have methods (or constructors) that only accept objects and not primitive data. By wrapping the primitive data within an object the functionality of the method (and its class) may be accessed.

Problem Ch 03-1

Assume an interactive Java program which asks the user for his/her first name and last name, and outputs the user's initials.

5. **Refer to Problem Ch 03-1.** For the program to get a name interactively a `Scanner` object must be instantiated. Write the Java statement to do this.

   ANS:
   ```
   Scanner scan = new Scanner(System.in);
   ```

6. **Refer to Problem Ch 03-1.** Write a statement using a `Scanner` method to get the first name interactively.

   ANS:
   ```
   firstName = scan.nextLine();
   ```

7. **Refer to Problem Ch 03-1.** Write a method to extract the initial from the first name.

   ANS:
   ```
   firstInitial = firstName.substring(0,1);
   ```

8. **Refer to Problem Ch 03-1.** Write a statement to guarantee that the initial will be a capital letter.

   ANS:
   ```
   firstInitial = firstInitial.toUpperCase();
   ```

9. What is autoboxing?

   ANS:
   Autoboxing provides automatic conversions between primitive values and corresponding wrapper objects. It makes your code shorter by relieving you of the need to provide explicit wrapping of primitive values and explicit extraction of primitive values.

10. Write a statement to create a `Color` object equivalent to `Color.ORANGE` using the `rgb` method.

    ANS:
    ```
    Color orange = Color.rgb(255, 165, 0);
    ```

11. Why is it often a good idea to group particular elements together in a `scene`?

    ANS:
    This makes it much easier to apply transformations such as rotations and position shifts to an entire group at once.