Herbert Schildt's *Java: A Beginner's Guide*, 6th ed.

Chapter 4 Self Test Answers

1. **What is the difference between a class and an object?**

   According to Schildt on page 104, "A class is a template that defines the form of an object. It specifies both the data and the code that will operate on the object.... Objects are *instances* of a class." The class declaration is like a blueprint of a building, many buildings can be created based on the blueprint, but each one is simply an instance of the class.

2. **How is a class defined?**

   A class is defined by declaring its exact form and nature. A (slightly modified) sample from Schildt page 105:

   ```
   class className {
     //declare instance variables
     int var1;
     double var2;
     String var3;

     //declare methods
     void method1(){
       //body of method
     }

     int method2(){
       //body of method
       return int;
     }
   } //end className
   ```

3. **What does each object have its own copy of?**

   Each object, or instance of a class, will have its own copy of *instance variables*. This means if there are three (3) objects created from a class, each has its own copy of instance variables. Variables are the data that is manipulated by the computer, and need to be separate for each particular object.

4. **Using two separate statements, show how to declare an object called *counter* of a class called *MyCounter*.**

   Normally, the object is usually declared and referenced to a new object using the following syntax:

   ```
   Animal dog = new Animal();
   ```

   However, it is possible to split the declaration and the referencing into two lines:

   ```
   Animal dog; // declare a reference to an object
   dog = new Animal(); //allocate an Animal object
   ```

5. **Show how a method called *myMeth()* is declared if it has a return type of *double* and has two *int* parameters called *a* and *b*.**

   This method would be declared as follows:

   ```
   double myMeth(int a, int b){
     // body of method
     return value;  // must be a double
   }
   ```

6. **How must a method return if it returns a value?**

   If a method returns a value of any data type, it must have a *return* statement which returns a value of the stated return to type to the object that called it. For example:

   ```
   int multiply(int a, int b){
     int x = a * b;
     return x;
   }
   ```

   Could be called by an object such as calc1 of they Calculator data type (class).

7. **What name does a constructor have?**

   A constructor always bears the exact same name as the class in which it is created. Java always creates a default constructor with no parameters, but if the programmer creates a constructor, the default constructor is no longer available. The programmer must designate the appropriate parameters whenever creating an object (declaring an instance of the class).

8. **What does *new* do?**

   The keyword *new* creates a new object of the class type being requested by allocating space in the memory of the computer to that object (it utilizes virtual memory addresses which the operating system translates into physical memory addresses). If there is insufficient memory available, this will result in a run-time exception. However, with modern computers having such vast amounts of RAM, it is unlikely that a run-time exception will occur unless working on an exceptionally large program.

9. **What is garbage collection, and how does it work? What is *finalize()*?**

   Garbage collection is completely automated in Java (unlike C++). There is no way be certain exactly when garbage collection will occur. When it does run, it looks for objects which no longer have any references pointing towards them. This means that all of the variables which point towards an object are out of scope, making the object eligible for garbage collection. Once garbage collection runs, it frees up space in memory to be allocated to new objects.

   finalize() is a specialized method which is rarely used in Java but may be useful in some applications. It is a method that runs immediately before an object is garbage collected. Possible uses for *finalize()* include such things as instructions on how to properly close an open file or perhaps to send a console message to the user.

10. **What is *this*?**

    this is a reference to the instance variable of the object (as opposed to the parameter passed to a method). It is often left out, although it is implied. However, in certain situations it is very important that *this* be included, such as when parameters share the same names as instance variables. For example:

    ```
    int multiply(int a, int b){
      this.a = a;
      this.b = b;
      return a * b;
    }
    ```

    This technique can often be avoided if parameters are named differently than instance variables.

11. **Can a constructor have one or more parameters?**

    Constructors can have any number of parameters, including none. Constructors are used to set initial values of instance variables within an object, and the number of parameters is limited only by the number of instance variables. Programmers can even designate multiple versions of constructors that react differently based on the parameters supplied by the object upon its creation.

12. **If a method returns no value, what must its return type be?**

    If a method returns no value, then its return type must be *void*. This is common throughout Java, with the most common instance being the *main* method used as a starting place for all programs.