Starting Out with Java – Early Objects
Answers to Review Questions and Exercises

Chapter 3

Multiple Choice and True/False

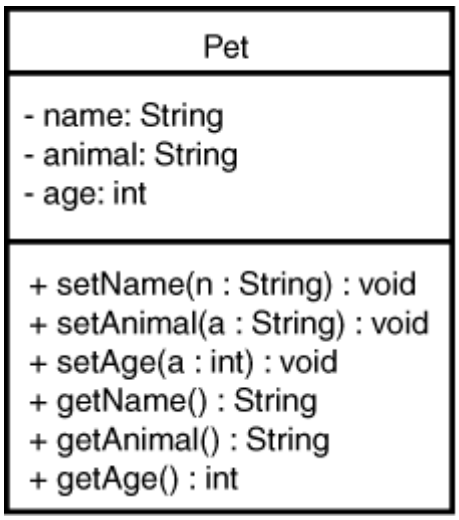
1. a
2. a
3. d
4. c
5. b
6. b
7. a
8. d
9. b
10. b
11. a
12. c
13. d
14. True
15. True
16. False
17. True
18. False
19. False

Find the Error

1. The constructor cannot have a return type, not even `void`.
2. The method should have `int` as its return type.
3. The parentheses are missing. The statement should read:
   ```
   Rectangle box = new Rectangle();
   ```

Algorithm Workbench

1.
a) UML diagram:



b) Class code:

```
public class Pet
{
   private String name;    // The pet's name
   private String animal;  // The type of animal
   private int age;        // The pet's age

   /**
    * setName method
    */

   public void setName(String n)
   {
      name = n;
   }

   /**
    * setAnimal method
    */

   public void setAnimal(String a)
   {
      animal = a;
   }

   /**
    * setAge method
    */

   public void setAge(int a)
   {
      age = a;
   }

   /**
    * getName method
    */
```

```
   public String getName()
   {
      return name;
   }

   /**
    * getAnimal method
    */

   public String getAnimal()
   {
      return animal;
   }

   /**
    * getAge method
    */

   public int getAge()
   {
      return age;
   }
}
```

2.

a) Constructor:

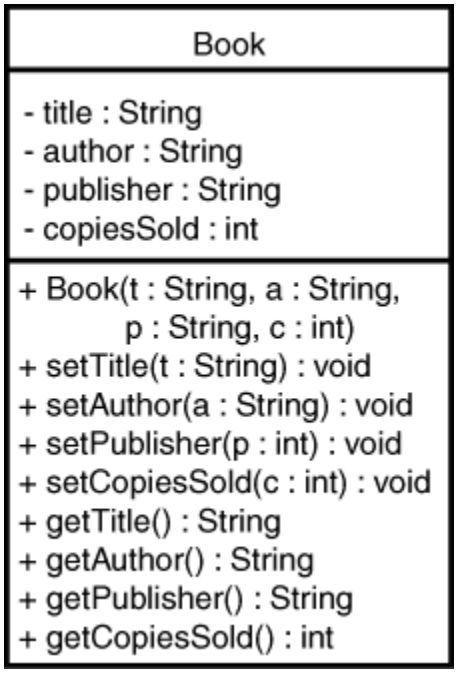
```
public Book(String t, String a, String p, int c)
{
   title = t;
   author = a;
   publisher = p;
   copiesSold = c;
}
```

b) Accessor and mutator methods

```
public void setTitle(String t)
{
   title = t;
}

public void setAuthor(String a)
{
   author = a;
}

public void setPublisher(String p)
{
   publisher = p;
```

```
}

public void setCopiesSold(int c)
{
   copiesSold = c;
}

public String getTitle()
{
   return title;
}

public String getAuthor()
{
   return author;
}

public String getPublisher()
{
   return publisher;
}

public int getCopiesSolde()
{
   return copiesSold;
}
```

c) UML diagram:



3. a) After eliminating duplicates, objects, and primitive values, the potential classes are: *bank*, *account*, and *customer*
   b) The only class needed for this particular problem is *account*.
   c) The account class knows its balance and interest rate.
   The account can calculate interest earned.

Short Answer

1. A class is a collection of programming statements that specify the attributes and methods that a particular type of object may have. You should think of a class as a "blueprint" that describes an object. An instance of a class is an actual object that exists in memory.
2. Classes are analogous to the blueprint.
3. An accessor method is a method that gets a value from a class's field but does not change it. A mutator method is a method that stores a value in a field or in some other way changes the value of a field.
4. When an object's fields are hidden from outside code, the fields are protected from accidental corruption. It is good idea to make all of a class's fields private and to provide access to those fields through methods.
5. Methods that are members of the class.
6. It creates an object (an instance of a class) in memory.
7. It looks in the current folder or directory for the file `Customer.class`. If that file does not exist, the compiler searches for the file `Customer.java` and compiles it. This creates the file `Customer.class`, which makes the `Customer` class available. The same procedure is followed when the compiler searches for the `Account` class.
8. This means that the parameter variable holds a copy of the value passed to it. Changes made to the parameter variable do not affect the argument.
9. Because they execute when an object is created.
10. The term "attribute" is a generic OOP term that refers to an item of data held by an object. The term "field" is a Java-specific term that refers to a member of a class that holds data. In Java, you use fields as attributes.
11. An argument is a value that is passed into a method. A parameter variable is a variable in the method that holds a copy of the argument.
12. If you do not write a constructor for a class, Java automatically provides one.
13. The default constructor.