Aviel Resnick

21.4 Problem Set

411:

1. What is an abstract class? Give an example.
   1. An abstract class is a class which is never instantiated, and lacks implementation.
   2. For example, a graphing class could have an abstract draw method, which can be implemented and used by a variety of objects (different shapes).
2. Why do we declare abstract methods?
   1. If a method can be reused, with a varied implementation, then declaring the method abstract, will allow the above to occur.
3. What is a final method? Give an example.
   1. A final method is a method whose implementation should not be changed by subclasses.
   2. For example, if a method squares an input, then the implementation should be consistent.
4. What is a protected variable? Give an example.
   1. A protected variable is a private variable, except it is visible to subclasses.
   2. One example could be a private variable, price, which is only relevant to its class, and any subclasses.

414-415:

1. Describe the process by which the Java Virtual Machine (JVM) locates the right method to execute at run time in the following example.
   1. The JVM searches for SomeMethod in the methods defined in someObject. Once finding someObject's definition of someMethod, it sees the call to the superclass's version of someMethod, and then executes the code from the superclass.

|  |
| --- |
| // In the server's code   private int myData;   public String someMethod(int x){  return myData + super.someMethod(x);  }   // In the client's code   System.out.println(someObject.someMethod(10)); |

2. Give one example, other than those discussed in this chapter, of the relationships of   
dependency, aggregation, and inheritance among classes.

1. An employee class aggregates data of a position and a salary.
2. In terms of shapes, a square inherits the behavior of a rectangle.
3. For dependency, A man class can send a message to an human class. For example, a man class could call the method getAge from the human class.

Polymorphism:

Polymorphism literally means many shapes, and in Java, refers to reusing chunks of code, by grouping relevant code under the same name. Literally, methods in different classes with the same name are considered polymorphic, and generally refer to similar code. For example (imaginary project), a math class, which might contain a method for computing the volume of a given shape, might be reused for different shapes, all under the name (volume).