Designing the Public Interface of a Class

1. In this lab, you will implement a vending machine. The vending machine holds cans of soda. To buy a can of soda, the customer needs to insert a token into the machine. When the token is inserted, a can drops from the can reservoir into the product delivery slot. The vending machine can be filled with more cans. The goal is to determine how many cans and tokens are in the machine at any given time. What methods would you supply for a VendingMachine class? Describe them informally.

2. Now translate those informal descriptions into Java method signatures, such as

```java
public void fillUp(int cans)
```

Give the names, parameters, and return types of the methods. Do not implement them yet.

3. What instance variables would you supply? Hint: You need to track the number of cans and tokens.

Implementing Methods

4. Consider what happens when a user inserts a token into the vending machine. The number of tokens is increased, and the number of cans is decreased. Implement a method:

```java
public void insertToken()
{
    // instructions for updating the token and can counts
}
```

You need to use the instance fields that you defined in the previous problem. Do not worry about the case where there are no more cans in the vending machine. You will learn how to program a decision "if can count is > 0" later in this course. For now, assume that the insertToken method will not be called if the vending machine is empty. What is the code of your method?

5. Now supply a method fillUp(int cans) to add more cans to the machine. Simply add the number of new cans to the can count. What is the code of your method?
6. Next, supply two methods `getCanCount` and `getTokenCount` that return the current values of the can and token counts. (You may want to look at the `getBalance` method of the `BankAccount` class for guidance.) What is the code of your methods?

Putting It All Together

7. You have implemented all methods of the `VendingMachine` class. Put them together into a class, like this:

```java
class VendingMachine {
    public your first method
    public your second method
    ...
    private your first instance field
    private your second instance field
}
```

What is the code for your complete class?

Testing a Class

8. Now test your class with the following test program.

```java
public class VendingMachineTester {
    public static void main(String[] args) {
        VendingMachine machine = new VendingMachine();
        machine.fillUp(10); // fill up with ten cans
        machine.insertToken();
        machine.insertToken();
        System.out.print("Token count = ");
        System.out.println(machine.getTokenCount());
        System.out.print("Can count = ");
        System.out.println(machine.getCanCount());
    }
}
```

What is the output of the test program?
Implementing Constructors

9. The `VendingMachine` class in the preceding example does not have any constructors. Instances of a class with no constructor are always constructed with all instance variables set to zero (or null if they are object references). It is always a good idea to provide an explicit constructor. In this lab, you should provide two constructors for the `VendingMachine` class:

- a default constructor that initializes the vending machine with 10 soda cans
- a constructor `VendingMachine(int cans)` that initializes the vending machine with the given number of cans

Both constructors should initialize the token count to 0. What is the code for your constructors?

Discovering Classes

10. Consider the following task: You are on vacation and want to send postcards to your friends. A typical postcard might look like this:

Dear Sue: I am having a great time on
the island of Java. The weather
is great. Wish you were here!
Love,
Janice

You decide to write a computer program that sends postcards to various friends, each of them with the same message, except that the first name is substituted to match each recipient. Concepts are discovered through the process of abstraction, taking away inessential features, until only the essence of the concept remains.

Using the abstraction process described in the book, what black box (class that will used to build objects of its type) can you identify?
11. We want to be able to write a program that will use our Postcard class to send postcards with the same message to different recipients.

The following class implements a Postcard.

```java
public class Postcard {
    public Postcard(String aSender, String aMessage) {
        message = aMessage;
        sender = aSender;
        recipient = "";
    }
    private String message;
    private String sender;
    private String recipient;
}
```

Notice that we do not set the recipient in the constructor because we want to be able to change the recipient, and keep the same message and sender. What method would you add to support this functionality? Implement the method.

12. Add a `print()` method to the `Postcard` class, that displays the contents of the postcard on the screen. What is the code of your method?

13. Try out your class with the following test code:

```java
public class PostcardTester {
    public static void main(String[] args) {
        String text = "I am having a great time on the island of Java. Weather is great. Wish you were here!";

        Postcard postcard = new Postcard("Janice", text);
        postcard.setRecipient("Sue");
        postcard.print();
        postcard.setRecipient("Tim");
        postcard.print();
    }
}
```

What is the output of your program?