Software and Programming I

Lab 7:
Construction of a
Simulated Cash Register
and a Student Class

Tobi Brodie (Tobi@dcs.bbk.ac.uk)

Coursework Plagiarism

Plagiarism is claiming the work of others as your own.

Plagiarism is a serious offence and can result in your exclusion from all colleges of the University of London.

The College policy on Plagiarism can be viewed here:
http://www.bbk.ac.uk/mybirkbeck/services/rules/Assessment_Offences.pdf

Copying work, either from each other or the internet, to complete the lab exercises amounts to plagiarism. When your lab exercises are assessed, you will be asked to explain exactly how your programs work and your coding choices before it is accepted.

Lab 7 Objectives

Implement classes with instance variables, mutator methods, accessor methods and a constructor.

Implement test classes in which objects can be created and manipulated.

Make changes to classes, without changing the public interface, to reinforce the concept of encapsulation.

A quick reminder

- An object holds **instance variables** that are accessed by its methods
- All **instances variables** should be private
- Most **methods** should be public
- A **mutator** method changes the object on which it operates
- An **accessor** method just queries the object for some information without changing it
- A **constructor** initialises the instance variables of an object
- The name of the constructor is the name of the class (the constructor has no return type, not even `void`)

Exercise 1: CashRegister

Implement the class **CashRegister** that simulates a cash register.

The cash register should have the following functionality:

1) Add an item’s price
2) Clear the cash register for a new sale
3) Get the (current) number of items in the sale
4) Get the (current) total price of the sale

CashRegister Class
(from SP1 lecture notes, lecture 6, 20/02/2020)

```
/**
A simulated cash register
*/
public class CashRegister
{
    /* private data
       you will need to declare instance variables to store the
       values for the cash register's item count and total price */

    /* methods (public interface) */
    public void addItem(double price)
    {
       /* This mutator (setter) method will be called each time
       there is a new item added, so you will need to increment
       the item count by 1, and add the current price to the total
       price (current price is the value stored in this method's only
       parameter,  price) */
    }
```

CashRegister class (2)

```
public void clear()
{
        /* This mutator method should  reset the local variables that
           store the values for the item count & total price */
}
public double getTotal()
{
        /* This is a standard accessor (getter) method. It should
           return the current value stored  in the variable you
           declared for the total price */
}
public int getCount()
{
        /* This accessor method should return the value stored for
           the item count */
}
```

CashRegister class (3)

```
  /* constructor */

    public CashRegister()
    {
      /* Within the constructor you need to initialise the
      instance variables for the total price and item count.
      The variables were declared earlier so should not
      include the data type */
    }
}  // end of class CashRegister
```

Remember:

- The constructor shares the same name as the class
- The constructor does not have a return type

Exercise 1 (cont): testing CashRegister class

The `CashRegister` on its own will not create objects, therefore, implement a test class to create instances of CashRegister.

The `CashRegisterTest` class from last week's lecture notes (shown on the next slide) can be used to test the `CashRegister` class as is, but **must be modified** to include the following in the lab today (see slide 11):

- The creation of 3 instances of `CashRegister`, each of which adds at least 2 item prices

- The item prices for each instance should be read by a `Scanner` object from the keyboard

Testing CashRegister Class

(from SP1 lecture notes, lecture 6, 20/2/2020, slide 21)

```
public class CashRegisterTest
{
   public static void main(String[] args)
   {
      // create an instance (register 1) of the CashRegister class
      CashRegister r1 = new CashRegister();
      // add items to register 1
      r1.addItem(2.95);
      r1.addItem(1.99);
      // use the CashRegister instance method getCount()
      System.out.println(r1.getCount());
      System.out.println((r1.getCount() == 2) ? "OK" : "FAIL");
      // use the CashRegister instance method getTotal()
      System.out.printf("%.2f\n", r1.getTotal());
      System.out.println((r1.getTotal()== 4.94)? "OK” : "FAIL");
   }
}
```

Exercise 1: CashRegisterTest class

```
import java.util.Scanner;

public class CashRegisterTest
{
   public static void main(String[] args)
   {
     /* write code to create Scanner object input */

     CashRegister r1 = new CashRegister(); // new instance

     /* write code: use input.nextDouble() to get an item's value and store in a
          new variable price of type double */

     r1.addItem(price);  // CashRegister instance method

     /* repeat code to get second item price and add item to object */

     System.out.println("Total cost is " + r1.getTotal());

     System.out.println("The number of items bought is "

                                                  + r1.getCount());
```

Exercise 1: CashRegisterTest class (2)

```
        /* write additional code to:
           (i) create the remaining two instances of CashRegister
           (ii) add a number of item prices in each of them
        */
      }
} // end of class CashRegisterTest
```

Exercise 2: Student Class

1. Implement a class `Student`.

The `Student` constructor should have two parameters of type `String` that will receive the values for the student's first and last names. The following instance methods will be required:

- `getName()` – this accessor method should return a `String` obtained by concatenating the first and last names (separated by a single space)
- `addQuizScore(`**`int`**` score)` – this mutator method should update the instance variable with the value of the `score` parameter
- `getTotalScore()` – accessor – returns the total score
- `getAverageScore()` – accessor – returns the average of all scores as accumulated by the `addQuizScore` method

2. Implement a suitable test class for `Student`.

Structure of Student Class

```
/**    Student class    */
public class Student
{
   // declare private data

   /* methods (public interface) */
   public void addQuizScore(int score)
    {
          // write code to add score to totalScore
          // increment scoreCount
   }

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

Structure of Student Class (2)

```
public int getTotalScore()
{
      // return totalScore
}
public double getAverageScore()
{
      // calculate average score and return the result
      // careful with count == 0, see SP1 lecture 4, 6/2/2020, slide 11
}
/* constructor */
public Student(String firstName, String lastName)
{
      name = firstName + " " + lastName;
            /* next initialise the instance variables:
               total quiz score and score count  */
}
}   // end of class Student
```

Test the Student Class

```
public class StudentTest
{

   public static void main(String[] args)
   {

      Student student1 = new Student("Sam", "Wise");

      /*
         write more code to test the methods:
         getName(), addQuizScore(int score),
         getTotalScore() and getAverageScore()
      */
    }
}  // end of class StudentTest
```

Marked Exercise 5: CashRegisterP816

Re-implement the class `CashRegister` so that it keeps track of the price of each added item. Re-implement the `clear`, `addItem`, `getTotal` and `getCount` methods. Add instance methods (see slide 19 for hints)

- `cancelLast` that removes the last added item and
- `displayAll` that prints out the prices of all items in the current sale.

The re-implementation, `CashRegisterP816`, along with the `CashRegisterP816Test` class, are the fifth marked exercise and you are required to complete them and show the working program by the 12th March. Failure to do so by this date will mean you cannot sit the second in-class test.

Marked Exercise 5: CashRegisterP816 (2)

Additional functionality required for `CashRegisterP816` (compared with `CashRegister`):

- One constructor of the class should receive a parameter that limits the number of items that can be added to the instance of the class.
- The second constructor of the class, without arguments, should use the default capacity (42).
- The method `addItem` should check that the item price is positive (and ignore any items with negative or zero price). The method should never exceed the capacity and ignore any items if the cash register is full.
- The new instance method `cancelLast()` removes the last item entered into the cash register (however, it should never attempt to remove an item from an empty cash register).

Your program should follow the pattern from

http://www.dcs.bbk.ac.uk/~roman/sp1/java/CashRegisterP816.java and

http://www.dcs.bbk.ac.uk/~roman/sp1/java/CashRegisterP816Test.java

Marked Exercise 5: CashRegisterP816 (3)

Hints:

To implement the additional functionality required, it is recommended that you use an array to store each item’s price in the instance of the `CashRegisterP816` class.

You should use the `count` variable when you add each new item, calculate the total price, print the cash register contents or cancel the last entered item. Remember that array indexes begin at 0!

The `getTotal()` method will need to calculate the total of all items stored in the array. A loop construct is ideal for this.

When implementing the `cancelLast()` method, the value stored in the variable `count` should be considered.

Home Work
Java for Everyone by C. Horstmann

Read Chapter 8, which is available online from

http://vufind.lib.bbk.ac.uk/vufind/Record/566484

and complete the following exercises:

- Exercise P8.2
- Exercise P8.4
- Exercise P8.12
- Exercise P8.14