1st PROGRAMMING EXERCISE

Writing a Java Class to Calculate the Mean and Variance of a Data Set

This is an introductory programming exercise you should do to refresh last year's knowledge of programming and start getting familiar with Java over the Christmas break before the beginning of the module in the next term. If you have trouble with it, you can get help in the first lab session in the next term.

The program to be developed in this exercise should calculate the mean and variance of a series of floating point numbers, i.e. "the data set". This is something that could be easily done in C by writing a small program that would have the following structure: two C functions *mean* and *variance* and a *main* program that initialises an array of double numbers, invokes the two functions by passing to them the array and prints the results. The "signature" of the two functions would be:

```
double mean (int numOfData, double data[])
double variance (int numOfData, double data[])
```

Note that in C we need to pass the size of the array as a separate parameter as the array is effectively a pointer to its first element.

We could do almost exactly the same in Java by introducing a MeanAndVariance class with two static methods *mean* and *variance* and a *main* method that would do exactly the same as the C program above. In this case, the "signature" of *mean* and *variance* would be simpler given that Java arrays are in fact objects which contain an instance variable *length* that denotes the array size, so there is no need to pass an additional parameter for the array size – note also the slightly different syntax of Java arrays:

```
static double mean (double[] data)
static double variance (double[] data)
```

The other difference in comparison to the C program is that we would have to call the mean and variance static methods by preceding them with the class name: static Java methods, i.e. those that do not use instance variables, are called with the convention <ClassName>.<methodName>(<parms>) while non-static methods, i.e. those that do use instance variables, need an object instance to be created and are called with the convention <objectReference>>.<methodName>(<parms>). So in our C-like Java program, we would call them using MeanAndVariance.mean(data) and MeanAndVariance.variance(data).

But the whole idea behind Java and other object-oriented languages such as C++, Objective-C, etc, is NOT to use them in a similar fashion to C but create objects and call their methods that operate on instance variables in addition to their parameters. So we should do this introductory exercise in a true object-oriented Java fashion, as opposed to C-style Java. We should then introduce a MeanAndVariance class that contains an array of double numbers through a *private* instance variable on which the *mean* and *variable* methods operate. This *data* instance variable should be initialised by the constructor. And because the data instance variable is private, we should also include public *getData* and *setData* methods to access it. In Java, private instance variables and methods are accessible only from within the class while public methods can be accessed from outside the class i.e. by other classes. Finally, all Java classes that contain instance variables should include a *toString* method, which returns the contents of an object instance, i.e. the instance variables, in the form of a string.

We are now ready to implement our first simple object-oriented class / program in Java that calculates the mean and variance of a small data set. We show the layout of this program on the next page, omitting the complete implementation which you should do.

```
import java.util.*;  // needed for the Scanner class in the enhanced version


public class MeanAndVariance {
   private double[] data;

   public double mean () {
      // method implementation here
   }

   public double variance () {
      // method implementation here
   }

   public double[] getData () {
      return data;
   }

   public void setData (double[] newData) {
      data = newData;
   }

   @Override // overrides Object toString - all Java classes inherit (ultimately) from Object
   public String toString () {
      // method implementation here
   }

   public MeanAndVariance (double[] myData) {  // constructor, called indirectly via new
      data = myData;
   }

   public MeanAndVariance () {  // constructor with no data, called indirectly via new
      data = null;
   }

   public static void main (String[] args) {
      double[] data1 = new double[5];
      // initialise data1 values here
      MeanAndVariance mv = new MeanAndVariance(data1); // create new object instance

      System.out.printf("The mean and variance of the following numbers are:\n");
      System.out.printf("Numbers: %s\n", mv.toString());
      System.out.printf("Mean: %f Variance: %f\n", mv.mean(), mv.variance());

      double[] data2 = new double[5];
      // initialise data2 values here
      mv.setData(data2); // reset data set

      System.out.printf("\nThe mean and variance of the following numbers are:\n");
      System.out.printf("Numbers: %s\n", mv.toString());
      System.out.printf("Mean: %f Variance: %f\n", mv.mean(), mv.variance());
   }
} // end class
```

We have now implemented our first object-oriented program in Java that finds the mean and variance of two data sets of 5 elements each. But the limitation is that the size of the data set is fixed to 5 and the values of each data set are "hard-coded" in the program. It would be nice to extend it to be able to get the data set size and values from the keyboard, i.e. the standard input. In order to do this, we can use the Scanner class which has methods nextInt and nextDouble to get the next integer or double number from the standard input.

So we can first introduce a static getNumOfData method which will read the number of data elements from the standard input. It also needs to check that the number of data specified by the user is at least 2 (also not a negative number), as otherwise there is no point in running the program. This method should be private, given that it will be only called by another method getDataSet which we will introduce next.

```
   private static int getNumOfData (Scanner input) {
      System.out.printf("Enter the number of arguments:\n");
      int ndata;

      while (true) {  // loop until we get it correctly
         ndata = input.nextInt();
         if (ndata < 2) {
            System.out.printf("the number of data should be >=2 !\n");
            continue;  // continue looping
         }
         break;  // we got it correctly, so break out of the loop
      }
      return ndata
   }
```

We should now introduce the getDataSet method which uses the one above.

```
   public void getDataSet (Scanner input) {
      int ndata = getNumOfData(input);
      data = new double[ndata];  // we create the array instance variable

      // here you should write code that gets exactly ndata numbers
      // from the standard input and initialises the array elements
      // you will need a while (true) loop as above
      // and you should use the Scanner nextDouble method
   }
```

Having now these two methods, we can rewrite our main program method to use these and be flexible instead of fixed size arrays with hard-coded values (see overleaf).

```
   public static void main (String[] args) {
      // create first new Scanner object instance that reads the standard input
      Scanner input = new Scanner(System.in);
      // create new MeanAndVariance object instance
      MeanAndVariance mv = new MeanAndVariance();

      mv.getDataSet(input);  // get data set 1
      System.out.printf("The mean and variance of the following numbers are:\n");
      System.out.printf("Numbers: %s\n", mv.toString());
      System.out.printf("Mean: %f Variance: %f\n", mv.mean(), mv.variance());
      System.out.printf("\n");

      mv.getDataSet(input);  // get data set 2
      System.out.printf("\nThe mean and variance of the following numbers are:\n");
      System.out.printf("Numbers: %s\n", mv.toString());
      System.out.printf("Mean: %f Variance: %f\n", mv.mean(), mv.variance());
   }
```

We have now implemented our first flexible object-oriented program that includes a reusable class MeanAndVariance which we can use elsewhere for calculating the mean and variance of a data set which can be given either from the keyboard through the *getDataSet* method or passed as array parameter in the *setData* method. A key aspect in object-oriented languages is to write highly reusable classes and this exercise demonstrates how to do this. The only thing this class does not do is to deal with incorrect keyboard input through exceptions, e.g. if the user does not type integer or floating point numbers but strings.

You should all implement this program to get familiar with Java and object-oriented programming, in fact the detailed description of this assignment is meant to be an introductory tutorial to object-oriented programming. Another aspect of this assignment is that it introduces “incremental development / prototyping”: you can first implement the program that works with a fixed size / hard-coded value data set and then extend it to work with any size data set the user provides from the keyboard.

Note that you should demonstrate this program as part of the first assignment and it will count towards your first assignment mark.