**Lab 9**

**Exercise 1 (Array Instance Variables, Arrays)**

1. Create a class called DoortoDoor to represent a door to door sales operation in an estate. The class should contain two (single dimensional arrays) instance variables:

houses // array of house numbers, int

sales // array of sales per house, int

1. Write a constructor that is passed in the number of houses in the estate and initialises both instance variables, i.e. creates the two arrays to be of this size

1. Write a method, fillHouseswhich assigns the following house number values to the houses array e.g. 1,2,3,4,5…….n (where n = number of houses in the estate)

1. Write a methodfillSales(int houseNum, int salequantity)whichassigns the sales quantity to the appropriate house in the sales array, the house numbers start at 1

1. Write a method displayAll to display each house number and the corresponding sales
2. Write a method calcAverageSales which returns the average sales value of all sales
3. Write a method minSales which prints the minimum sales value and the corresponding house number
4. Write a **test class** that will ask the user for the number of houses in the estate and create a DoorToDoor object for that estate. Fill the house array with house numbers.

Write a for loop that asks the user for the sales value for each house and populates the sales array for that house number with the sales value entered by the user by calling the fillSales method

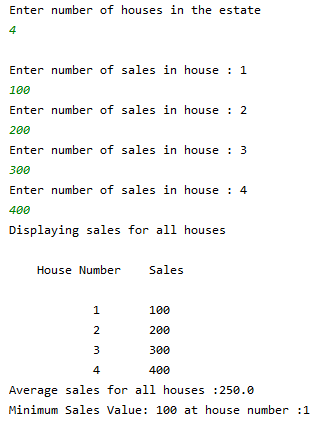
Display the average sales value from all the sales made

Display the minimum sales value and the corresponding house number

Display all the house numbers and their corresponding sales

See sample output on the next page

**Sample Output**



**Exercise 2 (Getters & Setters to be used to access/update data)**

**Part 1**

(a) Create a class called Student which contains the following private instance variables,

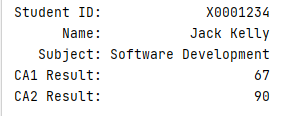
    String studentID;    String name;     String subject;     int ca1result;     int ca2 result;

Code setter methods for all instance variables. Code getter methods for each of the CA results.

Include in your Student class a print method so that the information on the student is printed to the screen.

(b) Create a test class, called TestStudent which creates one object called student1 of type Student. Initialise the above instance variables using the setter methods with appropriate values.

Call the print method from the TestStudent class to print the details of the student formatted as shown below:

.

**Part 2**

(a) Create another class called **Printcard** which contains the following private instance variables and a print method which prints the details of the Printcard, write setter methods for all instance variables.

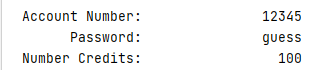
String accountNumber, String password, int numberCredits

 (b) Create a test class, called **TestPrintCard**, and create one Printcard object.

Initialise the instance variables with  appropriate information, using the setters.

Print the details of the printcard object to the screen formatted as shown below:

.

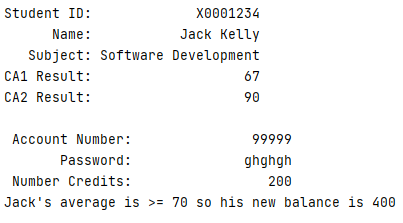


**Part 3**

Add the following details to your program **TestStudent** to use two different classes (the program already uses Student class so you have to also use PrintCard) by including the following:

* 1. Create an instance of PrintCard called mycard ,initialise the instance variables of the Printcard with appropriate values.
  2. Call the print method to display the details of the card
  3. In the **TestStudent**, calculate the average of the 2 CA results from the Student object,student1.
  4. If the average CA mark >=70, the print card should be updated to 400 credits, i.e. call the appropriate setter method. Modify your Test Student program to show the new card balance if the student achieves the required average, you will need to add appropriate getter method for this in the PrintCard class.

**Sample Output**



**Exercise 3**

Create a class called Product that can be used to represent a product in a

shop. The class should have the following private instance variables:

* product code (integer)
* product name (String)
* sellingPrice (double)
* costPrice (double)
* onHand (int) // represents the stock on hand

You will also need to write the following methods:

* a constructor to initialise the instance variable using a parameter list
* get methods for each of the data fields onHand, sellingPrice, and costPrice.
* set methods for productCode , productName ,sellingPrice and costPrice. An input argument to the setsellingPrice() method is a double value representing the new selling price. An input argument to the setcostPrice() method is a double value representing the new cost price. Input argument for setProductCode() is an integer representing the new product code and the input argument for setProductName() is a String representing the new product name.
* a method stockIn() used for increasing stock An input argument is the amount of goods (integer value). The stock on hand should be increased by this amount.
* a print method to print all details of the product

Write a program to test your class by performing the following i.e.: In the TestProduct

class:

* Create one object prod1 of type Product using the constructor you created
* Print the details of the object
* Call the methods, setCostPrice() and setSellingPrice() to change the cost and

selling price to new values

* Call stockIn() and increase the stock by 20 units
* Print the details of the object

Add the following functionality to the Product class

* a method sellProd() used when a product is being sold. An input argument to this method is an integer value representing the quantity sold. If the stock on hand is less than the required quantity no action is taken, otherwise the onHand is reduced by the quantity sold. The method returns a boolean value to indicate whether or not the stock on hand was reduced.

Add the following functionality to the TestProduct class

* a user wishes to purchase 10 units of the product. Call the sellProd() method to determine if there is sufficient stock to complete the sale. If not, display a message to the user and use the stockIn () method to acquire 100 more units of stock. If the sale is successful, print the profit made on the sale and also the amount of stock remaining.

**Sample Output**

