king Saud University
College of Computer &Information Science
CSC111 – Lab
OOP – III -
All Sections

Lab Exercise 1

Part 1

Design a class named Student to represent a student information. The class contains:

- A string data field named **studName** that specifies the student name.
- An integer data field named **studAge** that specifies the student age.
- A double data field named **studGPA** that specifies the student GPA.

Write a test class **TestStudent** that creates a student object then set his age.
Use the given partial implementation:

```
public class Student
{
   // data members
   // define instance variables studName, studAge, studGPA
   /* modifier† datatype  variable name*/
   /* modifier† datatype  variable name*/
   /* modifier† datatype  variable name*/
}
public class TestStudent{

   public static void main(String[] args){
         //create an object
         Student s1 = new Student();

         //set name, age and GPA for the object
         s1.studName = "Ahmed";

        //set age for s1
        s1.studGPA = -3.4;
        //print the student information
        System.out.println("Student Name:\t" + s1.studName
        +"\nStudent Age:\t" + s1.studAge +
        "\nStudent GPA:\t" + s1.studGPA);
   }
}
```

Sample Run

```
Student Name:   Ahmed
Student Age:    21
Student GPA:    -3.4
```

Part 2

In the previous part the user has set GPA to -3.4 which is illegible value for a GPA. We have to avoid this problem by checking the new GPA value to make sure that it is an eligible value, i.e. we should use private attributes (information hiding), setters and getters method.
Now make all your variables **private** instead of **public** and add the following setters and getters:

- A method named `setName(String studentName)` to store the student name.
- A method named `setAge(int studentAge)` to store the student age.
- A method named `setGPA(double studentGPA)` to store the student GPA.
- A method named `getName()` that returns the name of student.
- A method named `getAge()` that returns the student age
- A method named `getGPA()` that returns the student GPA

Use the class you in the previous part and the modify the test class to use the methods you implemented in part2 to set new values instead of using the variables directly.

```
public class Student
{
// data members
/*      */
// Set Student Name
public void setName(String studentName)
{
studName = studentName;
}
```

```
// Set the student age
/* setAge method */

// Set the student GPA
/*setGPA method */


//return student age
public int getAge()
{
    return studAge;
}

//return student name
/*getName()*/

//return student GPA
/*getGPA()*/

public class TestStudent{

     public static void main(String[] args){
        //create an object
         Student s1 = new Student();
        //set name, age and GPA for the object
        s1.setName("Ahmed");
        //set age for Ahmed
        s1.setGPA(-3.4);
        //print the student information
        System.out.println("Student Name:\t" + s1.getName()
        +"\nStudent Age:\t" + s1.getAge() +
        "\nStudent GPA:\t" + s1.getGPA());

     }
}
```

Sample Run

```
Student Name:   Ahmed
Student Age:    22
Student GPA:    -3.4
```

Part 3

As you may notice in the previous part we were able to force the user of class `Student` to use setters to set new values to the variables and we getters in case he needs to read the value of the variable. In this part, we

are going to check the new value inside the setters before we assign the the new value to the instance variable. For example, when the new value of the GPA is -3.4 we check first if the new value of the GPA is > 0. If so, we assign it to the instance variable otherwise we print an error message. In this case, -3.4 is not > 0 so we will not continue with the assignment.
Add an if statement inside the **setAge()** and **setGPA()** methods to check if the new value is an acceptable value or not. If yes, assign it to the instance variable, if not, print an error message and terminate the program.

```
public class Student
{
  // data members
  // method to Set the student name
  /*setName*/

  public void setGPA(double studentGPA)
  {
    if(studentGPA >= 0)
       studGPA = studentGPA;
    else
       System.out.println("The new value of the student
       GPA: " + studentGPA +
       " is not an acceptable value\nPlease try
       again...");
     }
   }

  // method to Set the student age (you must verify age)
  /*setAge*/

   //getage()
   //return student age

     public int getAge()
     {
          return studAge;
     }

     /*getName()*/
     //return student name


    /*getGPA()*/
    //return student GPA
```

```
}

class TestStudent{

      public static void main(String[] args){
        //create objects
        Student s1 = new Student();
        /* creation of a new student object s2*/

        //set name, age and GPA for the object s1
        s1.setName("Ahmed");
        //set age for s1
        s1.setGPA(3.4);
        //print s1 information

        System.out.println("Student Name:\t" + s1.getName()
        +"\nStudent Age:\t" + s1.getAge() +
        "\nStudent GPA:\t" + s1.getGPA());

        //set name, age and GPA for the object s2

        //set name for s2
        s2.setAge(23);
        s1.setGPA(-4.3);
       //print s2 information

       System.out.println(/*s2 information*/);
     }
}
```

Lab Exercise 2

Design a class named **Stock** for a company stock system.

The class contains:

- Data fields **`symbol`**, **`name`**, **`currentPrice`** and **`previousClosingPrice`**.
- A method named **`getChangePercent()`** that returns by how much percent the change of the price of the item has been lowered.
- Methods **`setSymbol()`** ,**`setCurrentPrice()`**, **`setPreviousClosingPrice`** and **`setName()`** that set the new values to the variables.
- Methods **`getPreviousClosingPrice()`** and **`getCurrentPrice()`** that return the previous closing price and the current price variables.

Draw the UML diagram for the class and then implement the class. Write a test program that prompts the user to enter the variables **`symbol`**, **`name`**, **`currentPrice`** and **`previousClosingPrice`** and displays the current price and the percentage in which the price has been cahnged by.

Sample Run 1

```
Enter symbol of stock: s
Enter company name: IAN
Enter previous closing price: 150
Enter current price: 120
Previous Closing Price: 150.0
Current Price: 120.0
Price Change: -20.0%
```

Solution

UML

Unlike previous program, in this program we are going to solve everything at once, i.e., write the whole class at once. First phase is to design your program as an OOP program. Draw UML diagrams for the two classes, `Stock` and `TestStock`.

Stock
- symbol: String - name: String - previousClosingPrice: double - currentPrice: double
+ getChangePercent():double + getCurrentPrice(): double + setSymbol(): void + setName():void

TestStock
+ main(): void

Now write your program. Construct two classes `Stock` and `TestStock`.

```java
import java.util.Scanner;
public class TestStock {
   public static void main(String[] args) {
      Scanner input = new Scanner(System.in);
      /* create object Stock */
      System.out.print("Enter symbol of stock:");
      String newSynbol = input.next();
      stock.setSymbol(newSymbol);
      System.out.print("Enter company name:");
      /* set the company name to name*/
      System.out.print("Enter previous closing price:");
      double prevPrice = input.nextDouble();
      /* set previous closing price to prevPrice*/
      System.out.print("Enter curret price:");
      /* Set the new current price to currentPrice*/
      /* Display stock info */

}

public class Stock {

     //data members
     /*         */

     public double getChangePercent() {
      /* return …………*/
     }

     // mwthod getPreviousClosingPrice
     /*                 */

     // method getCurrentPrice
     /*                 */

     public void setName(String newName){
     /*set name to newName */
     }

     /* setter for symbol */

     /* setter for current price */

     /* setter for previous closing price */

}
```