Ahmadu Bello University
Department of Mathematics
First Semester Examinations – June 2014
COSC211: Introduction to Object Oriented Programming I

Attempt **Four** questions Time: 120 mins

1. Examine the following code for the class `Horse` and answer the questions that follow.

```
1. //Horse.java
2.
3. public class Horse{
4.    private static final double MAX_WT = 500.0;
5.    private static final double MIN_WT = 25.0;
6.    private static final double MAX_HT = 2.0;
7.    private static final double MIN_HT = 0.5;
8.    private static final String DEF_NAME = "Horse";
9.    private static final double DEF_WT = 25.0;
10.   private static final double DEF_HT = 0.5;
11.
12.   private String name;
13.   private double weight;  //kilograms
14.   private double height;  //metres
15.
16.   //getters
17.   public String getName(){
18.      return name;
19.   }
20.
21.   public double getWeight(){
22.      return weight;
23.   }
24.
25.   public double getHeight(){
26.      return height;
27.   }
28.
```

```
29.    //setters
30.    public void setName(String name){
31.      this.name = name;
32.    }
33.
34.    public void setWeight(double weight){
35.      if (weight > MAX_WT) weight = MAX_WT;
36.      if (weight < MIN_WT) weight = MIN_WT;
37.      this.weight = weight;
38.    }
39.
40.    public void setHeight(double height){
41.      if (height > MAX_HT) height = MAX_HT;
42.      if (height < MIN_HT) height = MIN_HT;
43.      this.height = height;
44.    }
45.
46.    //constructors
47.    public Horse(String name, double weight, double
           height){
48.      setName(name);
49.      setWeight(weight);
50.      setHeight(height);
51.    }
52.
53.    public Horse(String name){
54.      this(name, DEF_WT, DEF_HT);
55.    }
56.
57.    public Horse(){
58.      this(DEF_NAME);
59.    }
60.
61.    public String toString(){
62.      return String.format("Name: %s\nWeight: " +
                              "%8.2f\nHeight: %8.2f",
                              name, weight, height);
63.    }
64. }//end of class horse
```


(*b*) Define a `toString()` method for the `Order` class that can be used to display the order details.

(*c*) Write an `OrderTest` class that will instantiate three objects from this class and display the order details.

6. (*a*) Write an application that will prompt the user to enter the first term, common difference, and the number of terms of an arithmetic progression (AP). It should compute the *n*th term of the series and the sum of the first *n* terms. Your code should ensure that the number of terms, *n*, is positive.

[Hint: use $T_n = a + (n-1)d$, and $S_n = An(a + T_n)$, where *a* is the first term, *n* is the number of terms, *d* is the common difference, T_n is the *n*th term of the series, and S_n is the sum of the first *n* terms.]

(*b*) A quadratic equation of the form $ax^2 + bx + c = 0$ has roots $\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$.

Write a fragment of code that efficiently determines the values of the roots (`root1` and `root2`) of the given equation. Assume that all the variables have been declared as type `double`. Note that the roots are real and distinct, real and equal, or complex according to whether the *discriminant* ($b^2 - 4ac$) is positive, zero or negative, respectively.

Format the output in the following ways (where `root1` and `root2` are the calculated roots of the equation).

(*i*) The roots are real and distinct : `root1`, `root2`.
(*ii*) The roots are real and equal: `root1`.
(*iii*) The roots are complex.

Note: the roots should not be displayed if they are complex.

5. Examine the following code and answer the questions that follow.

```
 1. //Order.java
 2. /*An order is placed for a number of items, if
 3.  *the number is big enough a discount is allowed
 4.
 5. public class Order{
 6.    private static final double UNIT_PRICE = 1000;
 7.    private static final double DISC_RATE = 0.05;
 8.    private static final int MIN_DISC_QUANT = 100;
 9.
10.    private int quantity;
11.
12.    public Order(int quantity){
13.       setQuantity(quantity)
14.    }//end of constructor
15.
16.    public getQuantity(){
17.       return quantity;
18.    }//end of method getQuantity()
19.
20.    public void setQuantity(quantity){
21.       if (quantity < 0);
22.          quantity = 0;
23.       this.quantity = quantity;
24.    }//end of method setQuantity()
25.
26.    public double calculateCost(){
27.       double totalCost = UNIT_PRICE * quantity;
28.
29.       if (quantity >= MIN_DISC_QUANT){
30.          discount = totalCost * DISC_RATE;
31.          totalCost -= discount;
32.       }//end of selection structure
33.
34.    }//end of method calculateCost()
35. }//end of class Order
```

(*a*) There are eight lines containing errors that would be detected by the Java compiler. Identify them and rewrite the lines with the errors corrected.

(*a*) On lines 4 to 10 fields are declared to be both `static` and `final`. Explain the meanings of these two keywords in this context.
(*b*) On lines 17 to 27 three *getters* are defined. What is their purpose? Why are they required?
(*c*) Two of the *setter* methods perform *data validation*. What is the purpose of performing data validation? Explain how the data validation in one of these setter methods works.
(*d*) Explain the usage of the `this` keyword on line 31.
(*e*) Explain what is meant by *constructor overloading*. Explain how this is carried out in the `Horse` class.
(*f*) There is a `toString()` method defined on lines 61 to 63. Explain what it does and how it is used.

2. (*a*) Create a class named `Grade` that includes the following.

(*i*) Declarations for two *fields* named `name` and `score` of type `string` and `double` respectively.
(*ii*) Definitions of two *constructors* where the first one should has no parameter while the second contains parameters that initialize the fields declared in (*i*) above.
(*iii*) *Getters* and *setters* of the fields declared in (*i*) above.
(*iv*) A method called `isScoreValid()` that will returns `true` if `score` is neither greater than 100 nor less than 0, `false` otherwise.
(*v*) A method called `computeGrade()` that will returns the `grade` (`char`) as follows.

```
grade = 'A' if score>=70
grade = 'B' if 60 <= score < 70
grade = 'C' if 50 <= score < 60
grade = 'D' if 45 <= score < 50
grade = 'E' if 40 <= score < 45
grade = 'F' if score < 40
```

(*b*) Create a test class called `GradeTest` that will enable the user to enter a student's name and score and see the students grade. The output should look like

```
Fati's score is 62 and grade is B
```

or (if the score is more than 100 or less than zero)

```
The score is invalid
```

3. Examine the following code and answer the questions that follow.

```
 1. //ProcessMarks
 2.
 3. import java.util.Scanner;
 4.
 5. public class ProcessMarks{
 6.    public static void main(String[] args){
 7.       Scanner input = new Scanner(System.in);
 8.       int sum = 0, num = 0;
 9.       System.out.print("Enter a student mark " +
                  "(-1 to exit): ");
10.       int mark = input.nextInt();
11.
12.       while(mark >= 0){
13.          if (mark >= 40){
14.             sum = sum + mark;
15.             num++;
16.          }
17.          System.out.print("Enter a " +
                     "student mark (-1 to exit): ");
18.          mark = input.nextInt();
19.       }
20.
21.       if (num > 0){
22.          double average = (double)sum / num;
23.          System.out.printf("The required " +
                 "average is %.1f\n", average);
24.          }
25.       else
26.          System.out.println("The are no " +
                 "marks to average");
27.    }//end of main()
28. }//end of class ProcessMarks
```

(*a*) What is the purpose of the `import` statement on line 3?
(*b*) What is the most important difference between a *pre-condition* loop and a *post-condition* loop? What type of loop starts on line 12?
(*c*) What is the `sum` that is evaluated on line 14?
(*d*) Explain the purpose of `(double)` on line 22 where `average` is evaluated.

(e) When running this program a user enters 45, 56, 34, 68 and then –1 in response to the prompts. Show the resulting output.
(*f*) When running this program a user enters 20, 34, 39, 15 and then –1 in response to the prompts. Show the resulting output.

4. *Ahmadu Bello University Press* wants to know the maximum, minimum, total, and average profit gained from years 2005 to 2014. Besides that, they are interested in knowing the difference between the maximum and minimum profit and those profits that are above average. The profits are as follows.

Year	**Profit(x)**
2005	5,000,000.34
2006	2,005.000.00
2007	3,020,000.97
2008	5,057,800.20
2009	4,500,000.67
2010	5,000,000.00
2011	3,048,900.56
2012	4,800,000.50
2013	2,980,000.71
2014	4,909,000.80

(*a*) Create a class called `Profit` that has the following members.

(*i*) Two array fields to store the profit and year named `profit` and `year` using appropriate data types.
(*ii*) A method called `getMaxProfit()` that will return the maximum profit.
(*iii*) A method called `getMinProfit()` that will return the minimum profit.
(*iv*) A method called `getSumOfProfit()` that will return the total profit.
(*v*) A method called `getAverageProfit()` that will return the average profit.
(*vi*) A method called `getRange()` that will return the difference between the maximum and minimum profits.
(*vii*) A method called `showProfitBelowAverage()` that will display all profits that are below average.

(*b*) Create a test class that will instantiate the `Profit` class and display the required results with suitable headings and labels.