Strathmore UNIVERSITY

SCHOOL OF COMPUTING AND ENGINEERING SCIENCES

BACHELOR OF COMPUTER NETWORKING AND SECURITY

CNS 1207: OBJECT ORIENTED PROGRAMMING

END OF SEMESTER EXAM GUIDE

Date: 15th December 2023 **Time:** 15:30-17:30

Instructions:

1. This Examination consists of **FIVE** questions.
2. Answer **Question ONE (COMPULSORY)** and any other **TWO** questions.

QUESTION 1 (30 Marks)

a) Which of the following constructors are invalid? **(3 Marks)**

```
public int ClassA(int one) {
...
}
public ClassB(int one, int two) {
...
}
void ClassC( ) {
...
}
```

b) What is the main purpose of a constructor? **(1 Mark)**

c) Complete the following constructor. **(2 Marks)**

```
class Test {
private double score;
public Test(double val) {
//assign the value of parameter to
//the data member
}
}
```

d) If the data member speed is private, is the following statement valid in a client program? **(2 Marks)**

```
Robot aibo;
aibo = new Robot();
double currentSpeed = aibo.speed;
```

e) Declare two class constants named MIN_BALANCE and MAX_BALANCE whose data types are double. **(2 Marks)**

f) Suppose Truck and Motorcycle are subclasses of Vehicle. Which of the following declarations are invalid? **(3 Marks)**

```
Truck t = new Vehicle();
Vehicle v = new Truck();
Motorcycle m1 = new Vehicle();
Motorcycle m2 = new Truck();
```

g) What is the purpose of the instanceof operator? **(1 Mark)**

h) If X is a private member of the Super class, is X accessible from a subclass of Super? **(2 Marks)**

i) If X is a protected member of the Super class, is X of one instance accessible from another instance of Super? What about from the instances of a subclass of Super? **(2 Marks)**

j) How do you call the superclass's constructor from its subclass? **(2 Marks)**

k) Can you create an instance of an abstract class? **(1 Mark)**

l) Must an abstract class include an abstract method? **(1 Mark)**

m) What is wrong with the following declaration? **(2 Marks)**

```
class Vehicle {
abstract public getVIN();
...
}
```

n) What must be defined before you can create an object? **(1 Mark)**

o) What's wrong with the following code? **(2 Marks)**

```
JFrame myWindow();
myWindow.setVisible(true);
```

p) Write statements to display the following shopping list on a message dialog: **(3 Marks)**

```
Shopping List:
       Apple
       Banana
       Lowfat Milk
```

QUESTION TWO (15 MARKS)

a) What will be the value of mystery when the following code is executed? **(2 Marks)**

```
String text, mystery;
text = "mocha chai latte";
mystery = text.substring(1,5);
```

b) What will be displayed on the message dialog when the following code is executed? **(2 Marks)**

```
String text = "I, Claudius";
JOptionPane.showMessageDialog( null, text.indexOf("I") );
```

c) Write a code fragment to display today's date in the 07-04-2002 format. **(3 Marks)**

Using the Bicycle class definition provided.

```
class Bicycle {
// Data Members
private String ownerName;
//Constructor: Initialzes the data member
public Bicycle( ) {
ownerName = "Unknown";
tagNo = "Unassigned";
}
...
}
```

d) Extend the Bicycle class by adding the second data member tagNo of type String. Declare this data member as private. **(2 Marks)**

e) Add a new method to the Bicycle class that assigns a tag number. This method will be called as

```
Bicycle bike;
bike = new Bicycle( );
...
bike.setTagNo("2004-134R");
```

(3 Marks)

f) Add another method to the Bicycle class that returns the bicycle's tag number. This method will be called as

```
Bicycle bike;
bike = new Bicycle( );
...
String tag = bike.getTagNo( );
```

(3 Marks)

QUESTION THREE (15 MARKS)

You will use the Demo class (in the file Demo.java) to test the MetricConverter class (in the file MetricConverter.java).

```
import javax.swing.*;

/**
 * This class is used to demonstrate the functionality of the MetricConverter
 * class.
 */

class Demo {

    public static void main (String[] args) {


        MetricConverter converter = new MetricConverter( );

        double inputInches;
        double centimeters, inches;
        String inputInchesAsString;

        //Get input
        inputInchesAsString      = JOptionPane.showInputDialog(null,
                                              "Enter inches: ");
        inputInches =  Double.parseDouble(inputInchesAsString);

        //Perform various conversion routines
        centimeters = converter.inchesToCentimeters( inputInches );
        inches = converter.centimetersToInches( centimeters );
```

```
        //Display the result
        JOptionPane.showMessageDialog(null,
                                      "Input: " + inputInches +
                                       " inches is equivalent to " +
                                       centimeters   + " centimeters");
        JOptionPane.showMessageDialog(null,
                                       "Converting back to inches: " + inches);

    }

}
```

```
/**
 * This class provides various routines to
 * convert metric measurements to U.S. units and
 * vice versa.
 */

class MetricConverter {

//---------------------------------
//    Data Members
//---------------------------------

  /**
   * A factor to convert inches to centimeters
   */
  public static final double INCHES_TO_CENTIMETERS = 2.54;

  /**
   * A factor to convert centimeters to inches
   */
  public static final double CENTIMETERS_TO_INCHES = 1 / 2.54;

  /**
   * A factor to convert feet to inches
   */
  public static final double FEET_TO_INCHES        = 12.0;

//---------------------------------
//    Constructors
//---------------------------------

   /**
    * Default constructor
    */
```

```
   public MetricConverter( ) {

   }

//-----------------------------------------------
//  Public Methods:
//
//      double  inchesToCentimeters        ( double         )
//      double  centimetersToInches        ( double         )
//      double  feetAndInchesToCentimeters ( double, double )
//-----------------------------------------------

   /**
    * Converts a given length in inches to
    * equivalent centimeters.
    *
    * @param inches  the length expressed in inches
    * @return length expressed in centimeters
    */
   public double inchesToCentimeters( double inches ) {
      return inches * INCHES_TO_CENTIMETERS;
   }

   /**
    * Converts a given length in feet and inches to
    * equivalent centimeters.
    *
    * @param feet   the feet portion of the length
    * @param inches the inch portion of the length
    * @return length expressed in centimeters
    */
   public double feetAndInchesToCentimeters(double feet, double inches) {
       // this is a stub
       return 0;
   }
}
```

a) As you have seen from the earlier checkpoint question, the centimetersToInches() method is not yet implemented. Using the inchesToCentimeters() method as a model, implement the centimetersToInches() method. **(3 Marks)**

b) Finish the implementation of the feetAndInchesToCentimeters() method. **(6 Marks)**
As you write it, follow these guidelines:

Use the class constants defined in this class.

Write the implementation using this strategy: first, calculate the total number of inches from the parameters feet and inches. (inches).

Then, call another method in this class that is already defined to convert the total inches to centimeters.

Make changes to your main() method in the Demo class to test your implementation. Keep these facts in mind:

feetAndInchesToCentimeters() takes two doubles as arguments. Use inputInches (declared as a double) and declare a new int local variable in your main() method called inputFeet to hold the input.

Don't forget to ask the user for the number of feet to convert, and place that value in your new variable before you call feetAndInchesToCentimeters().

Make sure your output looks just as readable as those already shown in the main() method.

Use the OutPoint class provided.

```
/**
 * This class provides various routines to
 * manipulate points in the coordinate plane (x, y).
 */

public class OurPoint {

//---------------------------------
//    Data Members
//---------------------------------

  /**
   * The x coordinate of a point
   */
    private double xCoord;

  /**
   * The y coordinate of a point
   */
    private double yCoord;

//---------------------------------
//    Constructors
//---------------------------------

   /**
    * Default constructor
    */
   public OurPoint( ) {
       xCoord = 0;
       yCoord = 0;
   }
```

```java
   public OurPoint(double xInit, double yInit ) {
       double xCoord, yCoord;
       xCoord = xInit;
       yCoord = yInit;
   }


//-----------------------------------------------
//      Public Methods:
//
//          void setX  ( double );
//          void setY  ( double );
//          double getX ( double );
//          double getY ( double );
//-----------------------------------------------

   /**
    * Changes the value of the x-coordinate of a point
    *
    *
    * @param x the new value for the x-coordinate of this point
    */
   public void setX (double x) {
       xCoord = x;
   }

   /**
    * Changes the value of the y-coordinate of a point
    *
    * @param y the new value for the x-coordinate of this point
    */
   public void setY (double y) {
       // this is a stub
   }

   /**
    * Returns the x-coordinate of a point
    *
    * @return the x-coordinate of this point
    */
   public double getX () {
       return xCoord;
   }

   /**
    * Returns the y-coordinate of a point
    *
    * @return the y-coordinate of this point
```

```
    */
   public double getY () {
       // this is a stub
       return 0.0;
   }


    // main method used to test the OurPoint class
    //-----------------------------------------------------


   public static void main (String [] args)  {
      OurPoint q  = new OurPoint (-7, 4); // Coordinates of q are (-5, 12)
      System.out.println("The x-coordinate of q is " + q.getX());
      System.out.println("The y-coordinate of q is " + q.getY());

   }


}
```

c) Implement a second constructor in the OurPoint class that allows the user to initialize a point with an x- and y- coordinate when it is created. **(2 Marks)**

d) Complete the implementation of the setY() stub. **(2 Marks)**

e) Complete the implementation of the getY() stub. **(2 Marks)**

QUESTION FOUR (15 Marks)

a) Consider the following class declaration. **(6 Marks)**

```
class QuestionOne {
      public final int   A = 345;
      public       int   b;
      private      float c;

      private void methodOne( int a ) {
            b = a;
      }

      public float methodTwo() {
            return 23;
      }
}
```

Identify invalid statements in the following main class. For each invalid statement, state why it is invalid.

```
class Q1Main {
      public static void main( String[] args ) {
            QuestionOne q1;
            q1 = new QuestionOne();

1           q1.A = 12;
            q1.b = 12;
2           q1.c = 12;

3           q1.methodOne( 12 );
4           q1.methodOne();
5           System.out.println( q1.methodTwo( 12 ) );
6           q1.c = q1.methodTwo();
      }
}
```

b) Is there any problem with the following class? Is the passing of an argument to the private methods appropriate? Are the data members appropriate? Explain. **(4 Marks)**

```
/*
      Problem Question4
*/
class MyText {

      private String word;
      private String temp;
      private int    idx;

      public String firstLetter( ) {
            idx = 0;
```

```
            return getLetter(word);
        }

        public String lastLetter( ) {
            idx = word.length() – 1;
            return getLetter(word);
        }

        private String getLetter(String str) {
            temp = str.substring(idx, idx+1);
            return temp;
        }
    }
```

c) What will be the output from the following code? **(3 Marks)**

```
    class Q2Main {
        public static void main( String[] args ) {
            QuestionTwo q2;
            q2 = new QuestionTwo( );
            q2.init():

            q2.increment();
            q2.increment();

            System.out.println( q2.getCount() );
        }
    }

    class QuestionTwo {
        private int count;

        public void init( ) {
            count = 1;
        }
```

```
        public void increment( ) {
                count = count + 1;
        }

        public int getCount() {
                return count;
        }
}
```

d) What will be the output from the following code? **(3 Marks)**

```
class Q3Main {
        public static void main( String[] args ) {
                QuestionThree q3;
                q3 = new QuestionThree( );
                q3init():

                q3.count = q3.increment() + q3.increment();

                System.out.println( q2.increment() );
        }
}

class QuestionThree {
        public int count;

        public void init( ) {
                count = 1;
        }

        public int increment() {
                count = count + 1;
                return count;
        }
}
```

QUESTION FIVE (15 Marks)

a) Consider the following class definitions. Identify invalid statements. **(6 Marks)**

```
class Car {
    public      String      make;
    protected   int         weight;
    private     String      color;

    …
}

class ElectricCar extends Car {
    private     int         rechargeHour;

    public ElectricCar() {
        …
    }

    // Copy Constructor
    public ElectricCar( ElectricCar car ) {
        this.make   = car.make;
        this.weight = car.weight;
        this.color  = new String( car.color );

        this.rechargeHour = car.rechargeHour;
    }

    …
}

class TestMain {
    public static void main( String[] args ) {
        Car         myCar;
        ElectricCar myElecCar;
```

```
            myCar = new Car();
            myCar.make = “Chevy”;
            myCar.weight = 1000;

            myCar.color = “Red”;



            myElecCar = new Car();
            myCar.make = “Chevy”;
            myCar.weight = 500;

            myCar.color = “Silver”;

        }
    }
```

b) Write code for the following: **(9 Marks)**

a. Create a class called Dog.
b. Include the following instance fields in the class Dog.
   String name
   String breed
   String barkNoise initialized to “Woof”
   double weight.
c. Include a constructor that creates a Dog object using the parameters for the name, breed and weight fields.
d. Create accessor and mutator methods for the fields name, breed and weight fields.
e. Add a functional method called bark that will display the value of the barkNoise field to the console.
f. Add an overloaded method called bark that will accept a String parameter for the noise the dog will make when it barks.
g. Create a main method and within it create a dog object.
h. Create method calls to test the methods defined in Dog.