CONFIDENTIAL

THE UNIVERSITY OF SYDNEY

SEAT NUMBER:	................………....................
STUDENT ID:	................………....................
SURNAME:	................………....................
OTHER NAMES:	................………....................
SIGNATURE:	................………....................

SCHOOL OF INFORMATION TECHNOLOGIES

INFO1103

INTRODUCTION TO PROGRAMMING

SEMESTER 2, 2014

TIME ALLOWED: TWO HOURS

This examination paper comprises 16 pages

INSTRUCTIONS TO CANDIDATES

Answer all questions using blue or black pen on this examination paper in the spaces provided.

The paper comprises 8 questions, each with multiple parts.

Questions are not worth equal marks. The mark awarded for each part is indicated. Marks total 100. To pass the exam you need at least 40% (= 40 marks).

The following material is allowed: One double-sided A4 page of your own notes. You do **not** need to leave this page with your examination paper.

THIS EXAMINATION PAPER IS NOT TO BE REMOVED FROM THE EXAMINATION ROOM

OFFICE USE ONLY

Q1 (22)	Q2 (10)	Q3 (8)	Q4 (8)	Q5 (6)	Q6 (12)	Q7 (12)	Q8 (12)	Q9 (10)	Total (100)

Question 1 [22 marks]

For each part of this question, indicate what the code will print.

a) [4 marks]

```
int i = 1;
while (i != 6)
{
   System.out.print(i + " ");
   i++;
   if (i == 6)
   {
      System.out.println("End");
   }
}
```

Answer:

```
1 2 3 4 5 End
```

b) [6 marks]

```
int n = 3;
for (int i = 0; i <= n; i++)
{
   for (int j = 0; j <= i; j++)
   {
      System.out.print(i);
   }
   System.out.println();
}
```

Answer:

```
0
11
222
3333
```

c) [4 marks]

```
int a = 1;
int b = 1;
do
{
   a = a + b;
   b++;
}
while (a < 10 * b);
System.out.println(a);
```

Answer:

```
211
```

d) [3 marks]

```
int s1 = 20;
if (s1 <= 20)
{
    System.out.print("1");
}
if (s1 <= 40)
{
    System.out.print("2");
}
if (s1 <= 30)
{
    System.out.print("3");
}
```

Answer:

```
123
```

f) [3 marks]

```
ArrayList<Integer> numbers;
numbers.add(4);
System.out.println(numbers.size());
```

Answer:

Error as the array list is not created and initialized with `new`, it is only declared.

g) [2 marks]

```
int[][] arr =
{
   { 1, 2, 3, 0 },
   { 4, 5, 6, 0 },
   { 0, 0, 0, 0 }
};
int[][] arr2 = arr;
System.out.println(arr2[2][1] + arr2[1][2]);
```

Answer:

6

Question 2 [10 marks]

Write a program in a class called called `Vowels.` It should read a word as string from the keyboard (using Scanner) and then print this word by replacing the vowels with underscores. There are 5 vowels in English: a, e, i, o and u. Assume that the word contains only lower-case characters.

Example output:

```
Enter a word with no spaces: marina
m_r_n_
```

Use good indentation and appropriate variable names. No comments are required.

Answer:

```
public class Vowels
{
     public static void main(String[] args)
      {
         Scanner in = new Scanner(System.in);
         System.out.print("Enter a word with no spaces: ");
         String word;
         word = in.next();
         for (int n = 0; n < word.length(); n++)
         {
              // Check each letter to see if a vowel, and increment counter
              char ch = word.charAt(n);
              if (ch == 'a'   || ch == 'e'
                          || ch == 'i' || ch == 'o' || ch == 'u' )
              {
                   System.out.print("_");
              }
              else
              {
                   System.out.print(word.charAt(n));
              }
         }
     }
}
```

Question 3 [8 marks]

Write a method called `inOrder` that takes as an argument a one-dimensional array of integers. It returns true if the array values are in increasing order and false otherwise. For example:

- if the array `ar` is: `23 3 8 6 15,` the call `inOrder(ar)` will return false;
- if the array `ar` is: `2 3 10 12 18,` the call `inOrder(ar)` will return true;
- if the array `ar` is: `2 3 10 10 18,` the call `inOrder(ar)` will return false.

Write also the javadoc comments for this method (general comment, param and return part). Use appropriate variable names and good indentation.

Answer:

```
/**
 * Checks if the array is in increasing order
 * @param myArray - the given array
 * @return true if the array is in order, and false otherwise
 */
public static boolean inOrder(int[] myArray)
{
   // Assume that the array is in order
   boolean ordered = true;

   // Traverse the array and check if the elements are in increasing order
   for (int i = 0; i < myArray.length - 1; i++)
   {
      if (myArray[i] > myArray[i + 1])
      {
         ordered = false;
      }
   }
   return ordered;
}
```

Question 4 [8 marks]

Write a program in a class called `TwoArrayLists`. It should:

- Read *N* integer numbers from the keyboard (using Scanner) and store them in an array list.
- Store the odd of these numbers also in a second array list and print the content of this arry list.
- Declare *N* as a named constant and initialize it to a chosen value, e.g. 6.

Use appropriate variable names and good indentation. There is no need to write comemnts.

Answer:

```
public class TwoArrayLists
{

       public static final int N = 6;
       public static void main(String[] args)
       {
```

```
                ArrayList<Integer> numbers = new ArrayList<Integer>();
                ArrayList<Integer> oddNumbers = new ArrayList<Integer>();
                int input;
                Scanner in = new Scanner(System.in);
                for (int i = 0; i < N; i++)
                {
                   System.out.println("Enter an integer: ");
                   input = in.nextInt();
                   numbers.add(input);
                   if (numbers.get(i) % 2 != 0)
                   {
                      oddNumbers.add(numbers.get(i));
                   }
             }

                //Print the elements of the second array list
                System.out.print("The second array list conrains:");
                for (int i = 0; i < oddNumbers.size(); i++)
                {
                       System.out.print(oddNumbers.get(i)+" ");
                }
        }

}
```

Question 5 [6 marks]

Circle the correct answer. Each part is worth 1 mark.

1) What must a subclass do to modify a private superclass instance variable?

a) The subclass must simply use the name of the superclass instance variable.
b) The subclass must declare its own instance variable with the same name as the superclass instance variable.
c) The subclass must use a public method of the superclass (if it exists) to update the superclass's private instance variable.
d) The subclass must have its own public method to update the superclass's private instance variable.

Answer: c

2) Consider the following code:

```
public void deposit(double amount)
{
   transactionCount ++;
   super.deposit(amount);
}
```

Which of the following statements is true?

a) This method will call itself.
b) This method calls a public method in its subclass.
c) This method calls a private method in its superclass.
d) This method calls a public method in its superclass.

Answer: d

3) Consider the following code:

```
public class Employee
{
   . . .
   public void setDepartment(String deptName)
   {
      . . .
   }
}
public class Programmer extends Employee
{
   . . .
   public void setProjectName(String projName)
   {
      . . .
   }
   public void setDepartment(String deptName)
   {
      . . .
   }
}
```

Which of the following statements is NOT true?

a) The `Programmer` class can call the `setDepartment` method of the `Employee` class.
b) The `Programmer` class overloads the `setDepartment` method.
c) The `Programmer` class overrides the `setDepartment` method.
d) The `Programmer` class can call the `setDepartment` method of the `Programmer` class.

Answer: b

4) Consider the following class hierarchy:

```
public class Vehicle
{
   private String type;
   public Vehicle(String type)
   {
      this.type = type;
   }
   public String getType()
   {
      return type;
   }
}

public class LandVehicle extends Vehicle
{
   public LandVehicle(String type)
   {
      . . .
   }
}

public class Auto extends LandVehicle
{
   public Auto(String type)
   {
      . . .
   }
}
```

Which of the following code fragments is NOT valid in Java?

a) `Vehicle myAuto = new Auto("sedan");`
b) `LandVehicle myAuto = new Auto("sedan");`
c) `Auto myAuto = new Auto("sedan");`
d) `LandVehicle myAuto = new Vehicle("sedan");`

Answer: d

5) Consider the following code:

```
Employee anEmployee = new Programmer();
anEmployee.increaseSalary(2500);
```

If the `Programmer` class inherits from the `Employee` class, and both classes have an implementation of the `increaseSalary` method with the same set of parameters and the same return type, which statement is correct?

a) The `increaseSalary` method of the `Programmer` class will be executed.
b) The `increaseSalary` method of the `Employee` class will be executed.
c) You must specify in the code which class's `increaseSalary` method is to be used.
d) It is not possible to determine which class's method is called.

Answer: a

6) Consider the following code:

```
Vehicle aVehicle = new Auto();
aVehicle.moveForward(200);
```

Assume that the `Auto` class inherits from the `Vehicle` class, and both classes have an implementation of the `moveForward` method with the same set of parameters and the same return type. Which class's `moveForward` method is to be executed is determined by ____.

a) the actual object type.
b) the variable's type.
c) the hierarchy of the classes.
d) it is not possible to determine which method is executed.

Answer: a

Question 6 [14 marks]

a) Write a class called `Clock` that represents a clock and a class called `AlarmClock` that represents an alarm closck and inherits from Clock. Include the followings:

- In the class `Clock:`
    - o 2 instance variables of type integer: `hour` and `minute` to represent the hour and minutes of the clock, respectively;
    - o 1 constructor that takes two parameters corresponding to the hour and minutes; using the methods `setHour` and `setMinute` described below.
    - o 2 accessor methods: `getHour` and `getMinute;`

- 2 mutator methods: `setHour` and `setMinute.` The method setHour should take 1 parameter corresponding to the new hour and check if it is valid, i.e. is between 0 and 12. If so, it should set `hour` to the new value, otherwise it should print an error message. The method `setMinute` is similar to `setHour`; it sets `minute` to the new minute value taken as an argument if it is valid, i.e. is between 0 and 59, and prints an error message otherwise.
- a method `toString` that returns the current time of the clock as a string.

- In the class `AlarmClock`
  - 3 additional instance variables – `alarmOn`, that shows if the alarm is on or off, `hrSet` and `minSet` that show the hour and minutes of the alarm, respectively.
  - 1 constructor that takes 2 parameters corresponding to the hour and minutes of the clock. It should call the constructor of the superclass. It should also set `alarmOn`, `hrSet` and `minSet` to appropriate default values.
  - A method setAlarm that takes two parameters, corresponding to the hour and minutes of the alarm, and sets the instance variables accordingly. There is no need to check if the parameter values are valid.
  - A method called `changeAlarm` that sets the alarm on, if it was off, and sets the alarm off if it was on. It doesn't take any parameters.
  - a method `toString` that returns the information about the alarm clock – current time, if the alarm is switched on or off, and the alarm time.

b) Write a class called `ClockTester` that tests the functionality of the classes `Clock` and `AlarmClock`. It should:

- Create an object of type AlarmClock and print its details.
- Change its current hour, set its alarm and print its details.

Write your code below. There is no need to write comments.

Answer:

```
public class Clock
{
	private int hour;
	private int minute;

	/*
	 * Create a clock with given hour and minutes
	 * @param hr - the given hour
	 * @param min - the given minutes
	 */
	public Clock(int hr, int min)
	{
		setHour(hr);
		setMinute(min);
	}

	/**
```

```
         * Check if the given hour h is valid, i.e. in the range of 0 - 24
         * If yes, set hour to h, otherwise print an error message
         * @param h - the given hour
         */
        public void setHour(int h)
        {
                if((h >= 0) && (h <= 24))
                        hour = h;
                else
                        System.out.println("Error: invalid hour");
        }

        /**
         * Check if the given minutes m are valid, i.e. in the range of 0 - 59
         * If yes, set minute to m, otherwise print an error message
         * @param m - the given minutes
         */
        public void setMinute(int m)
        {
                if((m >= 0) && (m <= 59))
                        minute = m;
                else
                        System.out.println("Error: invalid minute");
        }

        //Accessor methods
        public int getHour()
        {
                return hour;
        }

        public int getMinute()
        {
                return minute;
        }


        public String toString()
        {
                return "The current time is: "
                  + hour + ":" + minute;
        }
}

public class AlarmClock extends Clock
{

        private boolean alarmOn;
        private int hrSet, minSet;

        /*
         * Create an alarm clock with given hour and minutes
         * @param hr - the given hour
         * @param min - the given minutes
         */
        public AlarmClock(int hr, int min)
        {
                super(hr,min);
                alarmOn = false;
```

```
            hrSet = 0;
            minSet = 0;
        }

        /*
         * Set the alarm hour and minutes to the given values and set the alarm on
         * @param hr - the given hour
         * @param min - the given minutes
         */
        public void setAlarm(int hr, int min)
        {
            hrSet = hr;
            minSet = min;
            alarmOn = true;
        }

        public void changeAlarm()
        {
            if (alarmOn = false)
                alarmOn = true;
            else alarmOn = false;
        }

        public String toString()
        {
            return super.toString() +", the alarm is " + alarmOn + ", set for "
                        + hrSet + ":"+ minSet;
        }

}


public class ClockTester
{
        public static void main(String args[])
        {
            AlarmClock myClock = new AlarmClock(21,30);
            System.out.println(myClock.toString());
            myClock.setHour(23);
            myClock.setAlarm(7,30);
            System.out.println(myClock.toString());
        }
}
```

Question 7 [12 marks]

1) Write an interface type called `Computable.` It is an interface for methods that return the perimeter and area of a geometric figure. It includes two methods: `getPerimeter` and `getArea` that doesn't take any parameters and return a value of type `double`.
2) Write a class called `Circle`. It should:

- implement the interface `Comparable`;
- have 1 instance variable: the radius of the circle, which is of type double;
- have 1 constructor that takes as a parameter 1 variable corresponding to the radius of the circle.

3) Write a class called `Rectangle`. It should:

- implement the interface `Comparable`;
- have 2 instance variables: the width and height of the rectangle;
- have 1 constructor that takes as a parameter 2 variables corresponding to the width and height of the rectangle.

4) Write a class called `ComputableTester` to test the functionality of `Circle`, `Rectangle` and `Computable`. It should create 1 object of type `Circle` and 2 objects of type `Rectangle`, then place them in an array from type `Computable`, traverse this array using a loop and print the perimeter and area of the 3 figures.

Answer:

```
/**
 * An interface for methods that return the
 * perimeter and area of a geometric figure
 */
public interface Computable
{

       /**@return the perimeter*/
       double getPerimeter();

       /**@return the area*/
       double getArea();
}
public class Circle implements Computable
{
       private double radius;

       public Circle (double aRadius)
       {
              radius = aRadius;
       }

       public double getPerimeter()
       {
              return 2 * Math.PI * radius;
       }

       public double getArea()
       {
              return Math.PI * radius * radius;
       }
}

public class Rectangle implements Computable
{
       private double width;
       private double height;

       public Rectangle (double aWidth, double aHeight)
       {
           width = aWidth;
           height = aHeight;
       }
```

```
        public double getPerimeter()
        {
                return 2 * (width + height);
        }

        public double getArea()
        {
                return width *height;
        }
}

public class ComputableTester
{
        public static void main(String[] args)
        {
                Computable[] figures = new Computable[3];
                figures[0] = new Circle(10);
                figures[1] = new Rectangle(2,3);
                figures[2] = new Rectangle(4,6);

                for (int i = 0; i < figures.length; i++)
                {
                        System.out.println("figure " + i + ":");
                        System.out.println("perimeter = " + figures[i].getPerimeter());
                        System.out.println("area = " + figures[i].getArea());
                }
   }
```

Question 8 [10 marks]

1) [8 marks] Write a program that reads a file containing two columns of floating-point numbers. Prompt the user to enter the file name. Print the average of each column. Throw and catch an exception if the file is not found and print a corersponding message.

Use good indentation and variable names. No comments are required.

Answer:

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;


public class ReadingFileWithNumbers
{
   public static void main(String[] args)
   {
      Scanner in = new Scanner(System.in);
      System.out.print("Enter the file name: ");
      String filename = in.next();
      try
      {
         Scanner inFile = new Scanner(new File(filename));
         double sum1 = 0.0;
         double sum2 = 0.0;
         int count = 0;
         while (inFile.hasNextDouble())
         {
            sum1 = sum1 + inFile.nextDouble();
            sum2 = sum2 + inFile.nextDouble();
            count++;
         }
         System.out.println("Average of column 1: " + sum1 / count);
         System.out.println("Average of column 2: " + sum2 / count);
      }
      catch (FileNotFoundException e)
      {
         System.out.println("File not found!");
      }
   }
}
```

2) [1 marks] Write one line of code that will open a file called `dataOut.txt` for writing.

Answer:

```
PrintWriter out = new PrintWriter("dataOut.txt");
```

2) [1 mark] If a program `Prog` is started with the command

```
java  Prog  -Dname=data  -I\xyz  -v  hat.txt  b.txt  z.txt
```

what are the values of `args[0]`, `args[1]` and so on?

Answer:

```
args[0] = -Dname=data
args[1] = -I\xyz
args[2] = -v
args[3] = hat.txt
args[4] = b.txt
args[5] = z.txt
```

3) [1 mark] Circle the correct answer.
When you start a Java program from a command line and supply argument values, the values are stored as_____.

a) `int` values
b) `float` values
c) `String` values
d) the type of value indicated by the argument

Answer: c

4) [1 mark] Circle the correct answer.

a) Statements that may cause an exception should be placed within a `catch` block.
b) The main method of a Java program will handle any error encountered in the program.
c) Statements that may cause an exception should be placed within a `throws` block.
d) Statements that may cause an exception should be placed within a `try` block.

Answer: d

Question 9 [10 marks]

a) [8 marks] There are *n* people in a room, where *n* is an integer greater than or equal to 1. Each person shakes hands once with every other person. What is the total number of handshakes in a room? Write a program in a class called `Handshake` to solve this problem. It should:

a) Include the following **recursive** method to solve the problem:

```
/**
  * Computes the number of handshakes between n people using recursion
  * @param n - the number of people
  * @return - the number of handshakes
 */
public static int computeHandshakes(int n)
```

b) In the main method, write a loop that iterates for a number of people between 1 and 10 inclusive, computes the number of handshakes by calling the method `computeHandshakes` and prints them.

Use good indentation and variable names and write appropriate comments.

Answer:

```
public class Handshake
{
      /**
       * Computes the number of handshakes between n people using recursion
       * @param n - the number of people
       * @return - the number of handshakes
       */
      public static int computeHandshakes(int n)
      {
            // The base cases: 1) number of people is 1, handshakes = 0
            //                 2) number of people is 2, handshakes = 1
            if (n == 1)
                  return 0;
            else if (n == 2)
                  return 1;

            // Recursive case:  the nth person shakes hands with n-1 people;
            // then compute the number of handshakes for the remaining n-1 people
            return (n-1) + computeHandshakes(n-1);
      }

      // Main method to test the handshake method
      public static void main(String[] args)
      {
            for (int i = 1; i <= 10; i++)
            {
                  System.out.println("If there are " + i + " people " +
                        "in the room then there will be " +
                        computeHandshakes(i) + " handshakes.");
            }
      }
}
```

b) [1 marks] What is the difference between recursion and iteration?

Answer:
Recursion solves a problem by using the solution to the same problem with simpler values. Iteration uses some form of a loop to solve a problem.

c) [1 marks] What does *infinite recursion* mean? When does it occur?

Answer:
Infinite recursion means that the recursion never stops. It occurs when a method calls itself over and over again and never reaches an end.

END OF EXAM