Student ID Number:_____

SAMPLE FINAL EXAMINATION

This Sample Exam only lets you know how the exam looks like.
All the answers can be found in the lecture notes and lab solutions.

PAPER DESCRIPTION: Program Design & Construction

PAPER CODE: COMP603/ENSE600

TIME ALLOWED: 2 Hours plus 5 Minutes Reading Time

TOTAL MARKS: 100

INSTRUCTIONS:

1. Candidates must **NOT** write during the reading time
2. Dictionaries and thesaurus are allowed
3. Use blue or black pen only
4. Answer ALL the questions
5. **Please write your Student ID number on each page of the exam booklet and on each sheet of paper used.**

EXAM SUMMARY:

Question	Marks
Question 1	20
Question 2	25
Question 3	15
Question 4	15
Question 5	10
Question 6	15
TOTAL:	**100**

Question 1: Object-Oriented Programming Language [20 marks]

a) Briefly explain: **(6 marks)**

i. **Method Overloading**
ii. **Encapsulation**

NB: "briefly explain" means no more than 3 sentences.

b) Briefly explain **final class** and **final method**. **(6 marks)**

c) Use suitable examples to demonstrate and explain: **aggregation** and **composition.** **(8 marks)**

Question 2: Read Programs and Write Outputs [25 marks]

NB: assume all the relevant packages are imported in all the following programs.

a) What is the output from the following Java program? **(8 marks)**

```
public class StringTest {
   public static void main(String[] args){
      String s1= "PDC";
      String s2="PDC";
      String s3=new String("PDC");
      String s4=new String("PDC");
      System.out.println("1. "+(s1==s2));
      System.out.println("2. "+(s1==s3));
      System.out.println("3. "+(s3==s4));
      s3=s4;
      s4="PDC";
      System.out.println("4. "+(s3==s4));
   }
}
```

Your Answer:

b) Read the Java Program below and **write the output** by running **Test3** **(12 marks)**

```
public class Test3{

   public static void main(String[] args)    {
      ResultString strInMain = new ResultString();
      strInMain.str="Hello";
```

```
        System.out.println("Before change (in main): " + strInMain.str);
        changeData(strInMain);
        System.out.println("After change (in main): " + strInMain.str);
    }

    public static void changeData(ResultString result) {
        result.str=result.str+", it's a nice day!";
        ResultString tempStrResult = new ResultString();
        tempStrResult.str="Hi";

        result = tempStrResult;
        result.str=result.str+", it's a wet day!";
        System.out.println("tempStr (in method): "+tempStrResult.str);
    }
}
```

Your Answer:

c) Read the Java Program related to **Static variables** and write the output by running **StaticTest** **(5 marks)**

```
public class TheNumbers {
      static  int  aNumber=0;
      int  anotherNumber=0;
}

public class StaticTest {

   public static void main(String[] args)  {
      TheNumbers numObj1=new TheNumbers();
      TheNumbers numObj2=new TheNumbers();
     for(int i=0;i<5;i++)  {
         numObj1.aNumber++;
         numObj1.anotherNumber++;
         System.out.println(numObj1.aNumber+" "+numObj1.anotherNumber);
      }

     for(int i=0;i<5;i++)  {
         numObj2.aNumber++;
         numObj2.anotherNumber++;
         System.out.println(numObj2.aNumber+"   "+numObj2.anotherNumber);
      }
   }
}
```

Your Answer:

Question 3: Version Control and Multi-Threading [15 marks]

a) Briefly explain what **version control** is.

(5 marks)

b) Briefly explain the differences among the three methods for controlling a thread: **interrupt**, **isInterrupted** and **interrupted** **(5 marks)**

c) **Multiple Answer Question**: Which **TWO** output(s) from options **A to C** are possibly printed out from the following code fragment? **(5 marks)**

```
public void run(){
        for(int i=1;i<=2;i++)
                System.out.println( i+": "+word+" isInterrupted? "+ isInterrupted() );
}

public static void main(String[ ] args){
        Thread ping = new PingPongInterrupt("ping");
        Thread pong = new PingPongInterrupt("PONG");
        ping.start();
        pong.start();
        pong.interrupt();
    }
}
```

A:

1: ping isInterrupted? True
1: PONG interrupted? False
2: ping interrupted? True
2: PONG interrupted? True

B:

1: ping interrupted? False
1: PONG interrupted? True
2: ping interrupted? False
2: PONG interrupted? True

C:

1: ping interrupted? False
2: ping interrupted? False
1: PONG interrupted? True
2: PONG interrupted? True

Your Answer:

Question 4: Java Database Connectivity (JDBC) **[15 marks]**

The CAR table below is a table in the database CarDB. It contains 4 attributes and 4 records. The data types of the 4 attributes are: ID: int, MAKE: varchar, MODEL: varchar, PRICE: int, respectively.

Employees

ID	MAKE	MODEL	PRICE
1	TOYOTA	CAMRY	12000
2	TOYOTA	COROLLA	15000
3	NISSAN	PULSAR	8000

a) Briefly explain **how to use JDBC**. **(5 marks)**

b) **Add codes** to the following code fragment to establish a connection with Database CarDB **(5 marks)**

```
public static Connection conn;
public static String url="url="jdbc:mysql://localhost/CarDB";
public static String username="root";
public static String password="010101";

public void establishMySQLConnection()
{
   //your code goes here:

}
```

c) **Add codes** get the Model and Price of all Toyota cars in Car and print the query results **(5 marks)**

```
public ResultSet getQuery()
   {
      ResultSet rs=null;
      try {
         Statement statement=mysqlConn.createStatement();
         String sqlQuery="select model, price from car where brand='Toyota' “;
         //your code goes here:

      } catch (SQLException ex) {
         System.out.println(ex.getMessage());
      }
      return rs;
   }
```

Question 5: Design Patterns **[10 marks]**

a) Briefly explain **MVC**, including both the **problem** and **solution**.
(6 marks)

b) Use codes to demonstrate the use of **varargs idiom** **(4 marks)**

Question 6: Testing and Code Smells **[15 marks]**

a) Briefly explain **EACH** of the following **four terms**:

i. System function testing
ii. Usability testing
iii. Performance testing
iv. Stress testing

. **(5 marks)**

b) This question is regarding codes smells and refactor.

A junior developer Leo is asked to develop a program. The requirement is as follows: There are two types of questions: Math Quiz Question (MQQ) - The user needs to key in the results of the calculation. Multiple Choice Question (MCQ) - Each of the questions contains question text, four options, the correct answer and the explanations. The users need to input the selected option.

He has also completed the requirements:

i. MQQ weights 5 marks, while MCQ weights 10 marks.
ii. When a question is generated, the computer name and the timestamp should be captured.
iii. The 10 questions contain both MQQ and MCQ, two types of questions, and are randomly generated.
iv. MCQ questions should be managed externally, such as in a text file, rather than hardcoded into the program.
v. For each question, the info can be displayed. For MQQ, attributes like the question, correct answer, score and timestamp are required to be printed out. While, two more fields, i.e., the explanation and options, should be printed out for MCQ.

His codes are as follows:

NB: Assume that Question class and User class are properly created. Below Question Class only shows a fraction of the entire program.

```
public class Question {

   public Question() {
      this.questionType = generateNumber(2);

      if (questionType == 0) {
         this.num1 = this.generateNumber(100);
         this.num2 = this.generateNumber(100);
         this.mathOp = this.generateOperator();
         while (mathOp.equals("/") && num2 == 0) {
            this.num2 = this.generateNumber(100);
         }
         this.questionText = num1 + mathOp + num2 + "= ?";
         this.questionAnswer = getResult() + "";
         this.questionScore = 5;
         this.createdBy = getHostName();
         this.createdOn = new Date();

      } else if (questionType == 1) {
         List<String> questionList = new ArrayList<>();
         try {
            BufferedReader br = new BufferedReader(new
FileReader("MillionaireQuestions.txt"));
            String line = "";
            while ((line = br.readLine()) != null) {
               questionList.add(line.trim());
            }
            br.close();
         } catch (IOException ex) {
            System.err.println("IOException Error: " + ex.getMessage());
         }

         String[] questionArray =
questionList.get(generateNumber(questionList.size())).split("\\|");
         this.questionText = questionArray[0];
         this.questionOptions = questionArray[1];
         this.questionAnswer = questionArray[2];
         this.explanation = questionArray[3];
         this.questionScore = 10;
         this.createdBy = getHostName();
         this.createdOn = new Date();
      }
   }
```

```
    public void printQuestionInfo() {
        if (questionType == 0) {
            System.out.println("Quiz Question: " + this.questionText);
            System.out.println("The correct answer is: " + this.questionAnswer);
            System.out.println("The score of this question weights: " +
this.questionScore);
            System.out.println("Created by " + this.createdBy + ", Created on " +
this.createdOn);

        } else if (questionType == 1) {
            System.out.println("Multiple Choice Question: " + this.questionText);
            System.out.println(this.questionOptions);
            System.out.println("The correct answer is: " + this.questionAnswer);
            System.out.println("Explanation: " + this.explanation);
            System.out.println("The score of this question weights: " +
this.questionScore);
            System.out.println("Created by " + this.createdBy + ", Created on " +
this.createdOn);
        } else {

        }
    }

    @Override
    public boolean equals(Object obj) {
        return obj != null && obj instanceof Question
                && ((Question) obj).questionText.equals(this.questionText);
    }

    @Override
    public int hashCode() {
        String s = this.questionText;
        //System.out.println(s);
        return s.hashCode();
    }
}
```

```
public static void main(String[] args) {
    User user = new User();

    HashSet<Question> questions = new HashSet();
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter your name: ");
    user.userName = scanner.nextLine();

    System.out.println("Enter your answer as a number. Press 'X' to quit.");
    String answer;

    while (questions.size() < 10) {
        questions.add(new Question());
```

```
        }

        for (Question question : questions) {
            question.printQuestion();
            do {
                answer = scanner.nextLine();
            } while (!isValidAnswer(answer));

            if (answer.trim().equalsIgnoreCase("x")) {
                break;
            }

            if (question.checkAnswer(answer)) {
                user.score += question.questionScore;
                System.out.println("Correct! You get " + question.questionScore + " Points!");
            } else {
                user.score -= question.questionScore;
                System.out.println("Wrong! You lose " + question.questionScore + " Points!");
            }

        }
        System.out.println(user.userName + ", your score is: " + user.score);
}
```

Leo is very satisfied with his work since all of the proposed requirements are fulfilled. However, his leader told him that some other types of questions would be added to the program in the near future. Please give suggestions to Leo regarding how to refactor the codes. **(10 marks)**

Scratch