Sri Lanka Institute of Information Technology

B. Sc. Special Honours Degree in
Information Technology

Final Examination
Year 4, Semester I (2018)

IT421 – Modern Topics in IT

Duration: 2 Hours

Instruction to Candidates:

- This paper is preceded by **10 minutes reading period**. The supervisor will indicate when answering may commence
- There are 04 questions available answer for all 04 questions
- There are 08 pages including cover page
- This is an open book examination.
- Paper type – online
- **Question 03** is based on the **MTIT_2018_Refactor** project
- Use the Word document to write answers for **Question 04**

Question 1 (30 marks)

This question is related to the **Lambda Expressions** in Java section.

a) Transform following conventional Java program in to Java Lambda Expressions format.

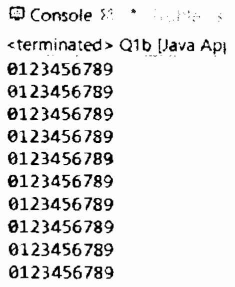
```
interface IGradeService{

      public String checkGrade(List<Integer> listOfMarks);
}

public class StudentsGrade implements IGradeService{

      @Override
      public String checkGrade(List<Integer> listOfMarks) {

            int total = 0;
            for (Integer mark : listOfMarks) {
                  total = total + mark;
            }
            double average = total/listOfMarks.size();
            if(average >= 80.0){
                  return "Best";
            }else if((average < 80.0) && (average >= 60.0)){
                  return "Merit";
            }else if((average < 60.0) && (average >= 45.0)){
                  return "Pass";
            }else{
                  return "Fail";
            }
      }

      public static void main(String[] args) {

            StudentsGrade grade = new StudentsGrade();
            ArrayList<Integer> listOfMarks = new ArrayList<Integer>();
            listOfMarks.add(85);
            listOfMarks.add(75);
            listOfMarks.add(60);
            listOfMarks.add(80);
            listOfMarks.add(100);

            String result = grade.checkGrade(listOfMarks);
            System.out.println("Result is = " + result);
       }
      }
```

(10 marks)

b) Convert the following Lambda code to conventional java program (Your program should display the same output as below) **(Hint:- override the run() method in Runnable interface)**

```
public class Q1b {

   public static void main(String[] args) {
     Runnable r2 = () -> {
        IntStream.iterate(0, x -> x + 1).limit(10).forEach(x -> {
                IntStream.iterate(0, y -> y + 1).limit(10).forEach(y -> {
                     System.out.print(y);
                 });
                     System.out.println();
               });
           };
               new Thread(r2).start();
       }
   }
```

Output



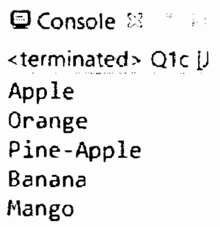
```
Console
<terminated> Q1b [Java Ap
0123456789
0123456789
0123456789
0123456789
0123456789
0123456789
0123456789
0123456789
0123456789
0123456789
```

(10 marks)

c) Convert below ***Lambda Expression*** example into ***Method reference*** format.

```
interface IReference {
      void displayFruits();
}

public class Q1cAns {

      public void displayFruits() {
            List<String> fruites = new ArrayList<String>();
            fruites.add("Apple");
            fruites.add("Orange");
            fruites.add("Pine-Apple");
            fruites.add("Banana");
            fruites.add("Mango");
            fruites.forEach(System.out::println);
      }

      public static void main(String[] args) {
            Q1cAns first = new Q1cAns();
            IReference iReference = first::displayFruits;
            iReference.displayFruits();
      }
}
```

Your output should be same as follows.



```
Console
<terminated> Q1c [J
Apple
Orange
Pine-Apple
Banana
Mango
```

(10 marks)

Question 2 **(25 marks)**

a) Using Java reflection implement a program to modify values of **private final fields** in the **Engineer** class given below. Use the Java Reflection technique to modify the Employee ID and Salary.

```
class Engineer{

      private final String employeeID;

      private final double salary;

      public Engineer() {
            this.employeeID = "EMP3005";
            this.salary = 150000.00;
      }

      @Override
      public String toString() {
            return "Employee = " + this.employeeID + ", Salary = " +
            this.salary;
      }
}
```

Output should be displayed as below.

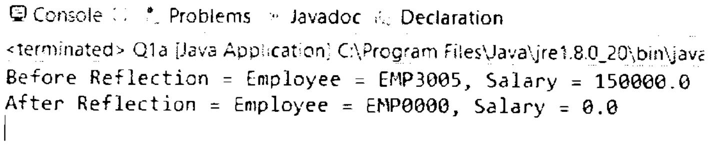
Output:

```
Employee Details  : Employee ID = EMP3005, Salary = 150000.00
Modified Employee Details : Employee ID = EMP0000, Salary = 0.0
```

Hint :- Your answer should start as follows.

```
public class Q1a {

 public static void main(String[] args) {

   Class<?> clazz;
   try {
      System.out.println("Before Reflection = " + new Engineer());

        ≪ Implement your code from here ≫

      System.out.println("After Reflection = " + engineer);

    } catch (ClassNotFoundException | NoSuchFieldException |
SecurityException | IllegalArgumentException| IllegalAccessException |
InstantiationException  e) {
        e.printStackTrace();
      }
   }
 }
```

Output


Console   Problems   Javadoc   Declaration

```
<terminated> Q1a [Java Application] C:\Program Files\Java\jre1.8.0_20\bin\java
Before Reflection = Employee = EMP3005, Salary = 150000.0
After Reflection = Employee = EMP0000, Salary = 0.0
```

(10 marks)

b) Refer the below implemented **Calculator** class. You should read all **private static final Fields** in **Calculator** class using Java Reflection technique. Your reflection class should give output as below

Display Output:

```
*
/
100.0
4.0
```

```
public class Caculator {

        private static final String MUL = "*";
        private static final String DIV = "/";
        private static final double NO1 = 100.00;
        private static final double NO2 = 4.0;

        public void calculate(String sign){

                if(MUL.equals(sign)){
                        System.out.println("Value = " + mul(NO1, NO2));
                }
                else if(DIV.equals(sign)){
                        System.out.println("Value = " + div(NO1, NO2));
                }
                else{
                        System.out.println("Please enter correct sign * or /");
                }
        }
        public double mul(double no1, double no2){
                return (no1 * no2);
        }

        public double div(double no1, double no2){
                return (no1 / no2);
        }
        }
```

(05 marks)

c) This is related to the above implemented Calculator class. Write a sample **reflection code** to execute the implemented method **calculate(sign)** and you have to display method output. (You can pass the multiply sign (*) as the parameter for this method)

```
Value = 400.0
```

(**Important:** - Marks are given only if you use reflection way of method execution. Marks will not be given if you just create objects and execute methods)

(10 marks)

Question 3 **(25 marks)**

You should refactor this code according to the **java coding conventions**. This code is not according to proper conventions and it is not readable. But the output in console should not be changed under any circumstances. Refactoring steps were given below.

Hint: - The sample code is about builder design pattern implementation. Please check the written main method of actual code that form the Sql query using builder design pattern.

In the provided code query is not as per the proper format: -
QueryBuilder().select().from().where().build() You should modify the given sample code to arrange it in proper order.

```
public static void main(String[] args) {

    // A valid query will be constructed
    Query query1 = new QueryBuilder().select("name").from("student").build();
    System.out.println(query1.toString());

    // A valid query will be constructed
    Query query2 = new QueryBuilder().select("name").from("student").where("name = 'Name1'").build();
    System.out.println(query2.toString());

    // will throw an exception
    Query query3 = new QueryBuilder().select("name").where("name = 'Name1'").build();
    System.out.println(query3.toString());
}
```

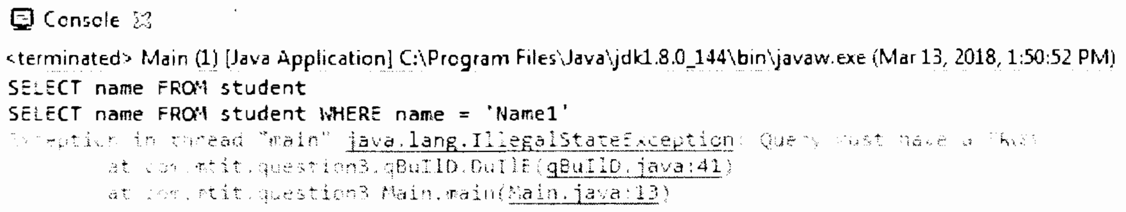
Refactoring Steps

a) Refactor the class **q** as **Query** and refactor all methods and variables in it
(05 marks)

b) Refactor the class **qBuIlD** as **QueryBuilder** and refactor all methods and variables in it
(10 marks)

c) Refactor the class **Main** by arranging all queries in proper format **QueryBuilder().select().from().where().build()**
(10 marks)

Output in console



```
Console
<terminated> Main (1) [Java Application] C:\Program Files\Java\jdk1.8.0_144\bin\javaw.exe (Mar 13, 2018, 1:50:52 PM)
SELECT name FROM student
SELECT name FROM student WHERE name = 'Name1'
Exception in thread "main" java.lang.IllegalStateException: Query must have a FROM
	at com.mtit.question3.qBuIlD.BuIlE(qBuIlD.java:41)
	at com.mtit.question3.Main.main(Main.java:13)
```

Question 4 (20 marks)

This question is related to the Enterprise Application Integration (EAI) and Unit Testing. Please use MS word document and write answers for below questions.

a) What are the approaches in Integration Testing.

(04 marks)

b) Explain one of the above approach with an example.

(05 marks)

c) Write 7 root patterns in Enterprise Application Integration.

(04 marks)

d) Explain one of the above root pattern with an example.

(03 marks)

e) Explain the issue in Service Oriented Architecture (SOA) and how it has been resolved by using mediator pattern? Explain it with a diagram.

(04 marks)

END OF THE EXAMINATION PAPER