IVE Information Technology

Information & Communications Technology

Programme Board

Instructions:

(a) This paper has a total of NINE pages including the covering page.

(b) This paper contains FOUR questions.

(c) Answer ALL questions. Each question is worth 25 marks.

Note: The result of this assessment will not be counted if you do not meet the minimum attendance requirement (if any) governed by the general academic regulations of your programme/course unless approval of the campus principal has been granted.

HIGHER DIPLOMA IN

SOFTWARE ENGINEERING
(IT114105)
COMPUTER SYSTEMS ADMINISTRATION
(IT124106)

MODULE TITLE:

Contemporary Topics in Software Engineering/ Computer Systems Administration

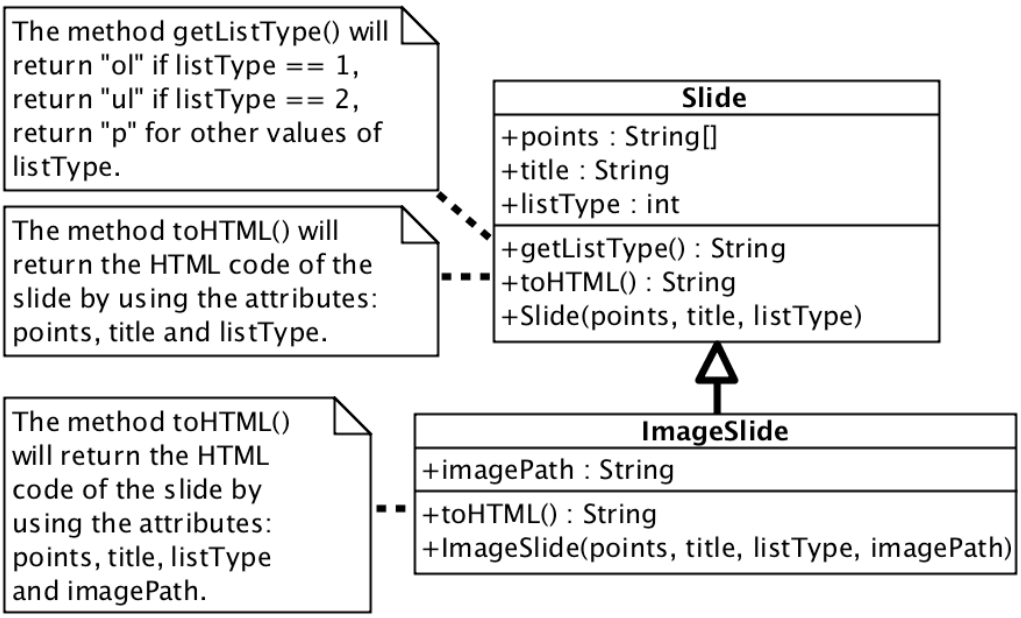
MODULE CODE: **ITP4507/ITP4605**

SEMESTER 4
MAIN EXAMINATION

5 JANUARY, 2016
1:30 PM TO 3:30 PM (2 hours)

This paper contains FOUR questions.
Answer ALL questions.
Each question is worth TWENTY-FIVE marks.

Q1 (a) Given the following class diagram and the test program in JavaScript:



Test code for testing the ***Slide*** and ***ImageSlide*** classes are given below:

```
points = ["Content Overview","Introduction to OO",
   "Final Conclusion"];
title = "OO Programming";
listType = 1;
imagePath = "images/oop1.jpg";
s = new Slide(points, title, listType);
document.write(s.toHTML());
listType = 2;
s2 = new ImageSlide(points, title, listType, imagePath);
document.write(s2.toHTML());
```

The expected output of **`document.write(s.toHTML())`** in HTML:

```
<h1>OO Programming</h1>
<ol>
    <li>Content Overview</li>
    <li>Introduction to OO</li>
    <li>Final Conclusion</li>
</ol>
```

QUESTION Q1 CONTINUES ON THE NEXT PAGE

QUESTION Q1 CONTINUES FROM THE PREVIOUS PAGE

The expected output of **`document.write(s2.toHTML())`** in HTML:

```
<h1>OO Programming</h1>
<ul>
    <li>Content Overview</li>
    <li>Introduction to OO</li>
    <li>Final Conclusion</li>
</ul>
<img src="images/oop1.jpg" />
```

Based on the class diagram given above, implement the classes ***Slide*** and ***ImageSlide*** in Javascript. [20 marks]

(b) The JavaScript code for implementing the classes ***Slide*** and ***ImageSlide*** in **Q1 part(a)** is stored in the file ***Slide.js*** and is applied in the following web page to generate a slide. You are asked to use the ***json string*** which is store in the variable ***jsonString*** to create a slide on the web page. [5 marks]

```
<html>
<body>
    <script src="Slide.js" type="text/javascript"></script>
    <script type="text/javascript">

    jsonString = '{"points":["Content Overview",'+
                '"Introduction to OO", "Final Conclusion"],'+
                '"title":"OO Programming","listType":1}';
    /*
    code to use jsonString and the class Slide
    to create a HTML slide content
    */

    </script>
</body>
</html>
```

Q2 You are asked to apply the Memento Design Pattern to solve the following problem. A simplified Stock Exchange system allows system administrators to create, update and view stocks. The ***Stock*** class for the system is implemented as follows:

```
public class Stock{
    private int stockCode;
    private int lotSize;

    public Stock(int stockCode, int lotSize){
         this.stockCode = stockCode;
         this.lotSize = lotSize;
    }

    public void setStockCode(int stockCode){
         this.stockCode = stockCode;
    }

    public void setLotSize(int lotSize){
         this.lotSize = lotSize;
    }

    public int getStockCode(){return stockCode;}

    public int getLotSize(){return lotSize;}

    public void showDetails(){
         System.out.println( "Stock Code : " + stockCode
                                + ", Lot Size : " + lotSize);
   }
}
```

(a) State and briefly describe the roles in the **Memento Design Pattern**. [5 marks]

(b) Implement a Java class ***StockMemento*** with a constructor for storing the original state of a ***Stock*** object and the ***restore()*** method which is used to recover the original state of the ***Stock*** object. [5 marks]

(c) Write a ***CareTaker*** class with the ***StockMemento*** class implemented in ***Q2 part (b)***. The ***CareTaker*** class has a stack for storing ***StockMemento*** objects. The ***CareTaker*** must have 3 methods: a constructor, ***saveStock(Stock stock)***, and ***undo()***. The ***saveStock(Stock stock)*** method is used to save the original state of a ***Stock*** object by creating a ***StockMemento*** object and then push the ***StockMemento*** object in the stack. The ***undo()*** method is to pop a ***StockMemento*** object from the stack and restore the ***Stock*** object associated with the ***StockMemento*** object to the original state saved by the ***StockMemento*** object. [5 marks]

QUESTION Q2 CONTINUES ON THE NEXT PAGE

QUESTION Q2 CONTINUES FROM THE PREVIOUS PAGE

(d) Draw a suitable class diagram to illustrate how you apply the Memento Design Pattern in this question. You must show clearly the associations among the classes. However, you **DO NOT** need to show the attributes and methods of the classes in your class diagram. [5 marks]

(e) Write a test program ***TestMemento.java*** to test the classes implemented in **Q2 parts (b) and (c)**.

(i) Your test program are required to do the following steps in sequence:

1. Create a ***CareTaker*** object
2. Create a ***Stock*** object with stock code = 13 and lot size = 1000
3. The ***CareTaker*** object saves the state of the ***Stock*** object
4. Set the ***Stock*** object's stock code to 1113
5. The ***CareTaker*** object saves the state of the ***Stock*** object
6. Set the ***Stock*** object's lot size to 500
7. Undo the last change to the ***Stock*** object
8. Display the details of the ***Stock*** object

[4 marks]

(ii) Assume that your test program is implemented successfully, what is the expected output when the test program runs?

[1 mark]

Q3 Giant Fitness Club fully subscribe the America Fitness. In order to use the existing Giant Fitness Club System (GFCS) to view the members' information in America Fitness Club System (AFCS), the IT manager asks his subordinate to review the API in both systems.

There are THREE essential Java classes ***AClubMember***, ***GClubMember***, and ***DisplayMember***. Class ***AClubMember*** stores the name, hkid, date of join in US format. Class ***GClubMember*** stores the name, hkid, date of join in UK format. Class ***DisplayMember*** shows the ***GClubMember***. The source codes of the THREE classes are as follow.

```
class AClubMember {
    String lastName; // Family Name
    String initials; // name
    int[] joinDate = new int[3];// MM, DD, YYYY
    public AClubMember( ) { this("UNKNOW","UNKNOW",0,0,0);}
    public AClubMember (String lastName, String initials, int mm,
       int dd, int yyyy) {
       this.lastName = lastName;
       this.initials = initials;
       joinDate[0] = mm; joinDate[1] = dd; joinDate[2] = yyyy;
    }
    public String getLastName( ) { return lastName; }
    public String getInitials( ) { return initials; }
    public int[] getJoinDate ( ) { return joinDate; }
}

class GClubMember {
    String surname; // Family Name
    String firstName; // name
    int[] annualDate = new int[3];// DD, MM, YYYY
    public GClubMember( ) { this("UNKNOW","UNKNOW",0,0,0);}
    public GClubMember (String surname, String firstName,
       int dd, int mm, int yyyy) {
       this.surname = surname;
       this.firstName = firstName;
       annualDate[0] = dd; annualDate[1] = mm;
       annualDate[2] = yyyy;
    }
    public String getSurname( ) { return surname; }
    public String getFirstName( ) { return firstName; }
    public int[] getAnnualDate( ) { return annualDate; }
}
```

QUESTION Q3 CONTINUES ON THE NEXT PAGE

QUESTION Q3 CONTINUES FROM THE PREVIOUS PAGE

```
class DisplayMember {
     public static void showGClubMember(GClubMember gm) {
         System.out.println("Member: " + gm.getFirstName( ) +
            ", "+ gm.getSurname( ) + "\t [Annual date: " +
            gm.getAnnualDate( )[0] + "-" +
            gm.getAnnualDate( )[1] + "-" +
            gm.getAnnualDate( )[2] +"]");
     }
    }
```

(a) Write a few sentences to describe the design of **Object Adapter Pattern**. [4 marks]

(b) Draw a suitable class diagram to illustrate how you apply the **Object Adapter Design Pattern** in this question. Show all the required classes, methods and the associations between the classes. For adapter class, you are also required to show attribute and constructor. [9 marks]

(c) You are required to use Java to implement your solution in **Q3 part (b)**. [6 marks]

(d) Comment why we cannot use **Class Adapter Pattern** to solve the above problem. [1 mark]

(e) Implement a simple test program to test the implementation of the adapter class in your answer to **Q3 part (c)**. [5 marks]

Q4 Brian is a junior programmer and is working for the Human Resource Department of a company. He was asked to write an application for printing salary statements for employees of the company. The company has two kinds of employee, **full-time employee** and **part-time employee**. The salary structure of the two kinds of employee are different. For full time employees, the salary is a fixed monthly salary. For part-time employees, the salary is equal to the total number of working hours of the month multiplied by the hourly wage. Finally, Brain has implemented the following classes for the application.

```
public class Employee {
   public static final int FullTime = 0;
   public static final int PartTime = 1;
   private int employeeType;
   private String name;
   private int monthlySalary;
   // monthly salary for part-time employee is set to $0
   private int hourlyWage;
   // hourly wage for full-time employee is set to $0
   private int hoursOfWork;
   // hours of work for full-time employee is set to 0 hr

   public Employee (String name, int employeeType, int monthlySalary,
          int hourlyWage, int hoursOfWork) {
      this.name = name;
      this.employeeType = employeeType;
      this.monthlySalary = monthlySalary;
      this.hourlyWage = hourlyWage;
      this.hoursOfWork = hoursOfWork;
   }
   public int getEmployeeType() { return employeeType; }

   public void ftSalaryStatement() {
      System.out.println("Full Time Staff - Name : " + name +
           ", Monthly Salary : $" + monthlySalary);
   }

   public void ptSalaryStatement() {
      System.out.println("Part Time Staff - Name : " + name +
            ", Hours of Work : " + hoursOfWork +
            ", Hourly Wage : $" + hourlyWage +
            ", Salary : $" + hourlyWage*hoursOfWork);
   }
}
```

QUESTION Q4 CONTINUES ON THE NEXT PAGE

QUESTION Q4 CONTINUES FROM THE PREVIOUS PAGE

```
public class SalarySummary{
   private Employee [] employees;
   private String year;
   private String month;

   public SalarySummary(Employee [] employees, String year, String month)
   {
      this.employees = employees;
      this.year = year;
      this.month = month;
   }

   public void printSalarySummary() {
      System.out.println("Salary Summary for " + month + "," + year
                                                          + " :- ");
      for (int i = 0 ; i < employees.length ; i++)
          if (employees[i].getEmployeeType() == Employee.FullTime)
             employees[i].ftSalaryStatement();
          else if (employees[i].getEmployeeType() == Employee.PartTime)
             employees[i].ptSalaryStatement();
   }
}
```

Brian showed his classes to his supervisor. His supervisor told him that his program did not conform to the Open-Closed Principle and recommended him to modify the design of his program so that the programs can be easily extended to handle new types of Employee (such as contract staff) which have different salary packages.

(a) Briefly describe the **Open-Closed Principle**. [2 marks]

(b) Explain why the program does not conform to the Open-Closed Principle when the program is extended to handle new employee types . [2 marks]

(c) Draw a class diagram to show your new design which conforms to the Open-Closed Principle. Show all the required classes, attributes, methods, constructors and the associations between the classes. [10 marks]

(d) Write the Java codes for implementing the classes which have been changed or added in the new design given in your class design to the answer to **Q4 part (c)**. [11 marks]

Note : For the ***SalarySummary*** class, you only need to show your implementation of the revised ***printSalarySummary*** method because there is no need to change the other codes in this class.

******* END OF PAPER *******