IVE Information Technology

Information & Communications Technology
&
Multimedia & Entertainment Technology

Programme Board

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

8 JANUARY, 2015
1:30 PM TO 3:30 PM (2 hours)

Instructions:

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

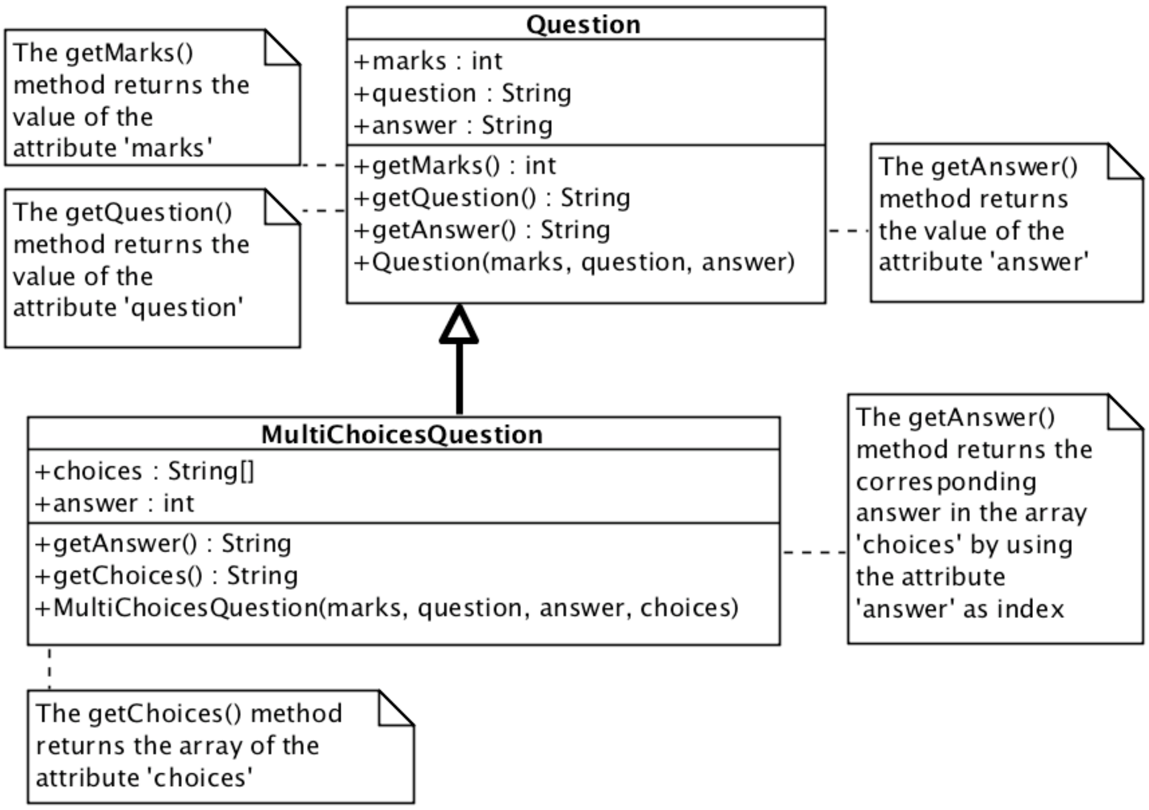
(b) This paper contains FOUR questions.

(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.

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

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


Question
+marks : int
+question : String
+answer : String
+getMarks() : int
+getQuestion() : String
+getAnswer() : String
+Question(marks, question, answer)
The getMarks() method returns the value of the attribute 'marks'
The getQuestion() method returns the value of the attribute 'question'
The getAnswer() method returns the value of the attribute 'answer'
MultiChoicesQuestion
+choices : String[]
+answer : int
+getAnswer() : String
+getChoices() : String
+MultiChoicesQuestion(marks, question, answer, choices)
The getAnswer() method returns the corresponding answer in the array 'choices' by using the attribute 'answer' as index
The getChoices() method returns the array of the attribute 'choices'

Test code for testing the ***MultiChoicesQuestion*** class is given below:

```
choices = ["24","32","44"];
mq = new MultiChoicesQuestion(20, "20 + 12 = ?", 1, choices);
console.log("marks:",mq.getMarks(), ", question:",
    mq.getQuestion(), ", answer:", mq.getAnswer());
```

The expected output of the test code:

```
marks: 20 , question: 20 + 12 = ? , answer: 32
```

Based on the class diagram given above, implement the classes ***Question*** and ***MultiChoicesQuestion*** in Javascript. [15 marks]

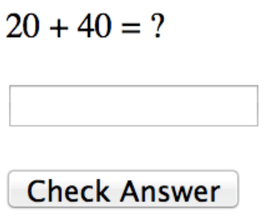
QUESTION Q1 CONTINUES ON THE NEXT PAGE

QUESTION Q1 CONTINUES FROM THE PREVIOUS PAGE

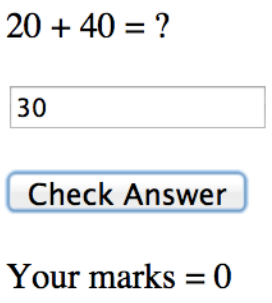
(b) The JavaScript code for implementing the classes ***Question*** and ***MultiChoicesQuestion*** in **Q1 part(a)** is stored in the file ***Question.js*** and is applied in the following web page to generate the question. You are asked to implement the ***checkAnswer()*** function for completing the implementation of the following web page to check the answer entered by the user.

```
<html>
<body>
<script src="Question.js" type="text/javascript"></script>
<script type="text/javascript">
   q = new Question(20, "20 + 40 = ?", "60");
   document.write(q.getQuestion()+"<p />");
</script>
<form>
   <input id="answer" /><p />
   <input type="button" value="Check Answer"
       onclick="checkAnswer()" />
</form>
<div id="result"></div>
<script>
// code for function checkAnswer() to be completed ...
</script>
</body>
</html>
```

The following is the screen dump of the web page when it is loaded:


20 + 40 = ?

The following is the screen dump of the web page which will give a 0 mark if the user enters incorrect answer '30' in the input box and presses the button:


20 + 40 = ?

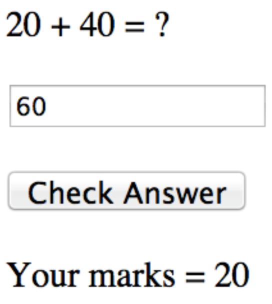
30

Your marks = 0

QUESTION Q1 CONTINUES ON THE NEXT PAGE

QUESTION Q1 CONTINUES FROM THE PREVIOUS PAGE

The following is the screen dump of the web page which will give a corresponding marks according to the question object ***q*** if the user enters correct answer '60' in the input box and presses the button:


20 + 40 = ?

60

Check Answer

Your marks = 20

To check the answer of a question, a user enters an answer in the input field and presses the button. The ***checkAnswer()*** function gets the answer from the input field, gets the marks according to the answer, changes the content in div (with id="result") to show the marks.

Write the ***checkAnswer()*** function in JavaScript. [10 marks]

Q2. (a) James is a junior programmer who is going to develop a Text Editor. He has developed a test program to control the words inside a paragraph. The test program let users append or delete last appended word in a paragraph. The ***ParagraphEditor*** and ***Paragraph*** classes are implemented as follows:

```
import java.util.Scanner;
import java.util.Stack;
public class ParagraphEditor {
  public static void main(String[] args) {
    Paragraph p = new Paragraph();
    Stack commandStack = new Stack();

    Scanner sc = new Scanner(System.in);
    int option, currentIndex = 0;
    String word, command, undoWord;

    while (true) {
      System.out.println("\n0 = exit, 1 = undo, 2 = append word, "
              + "\n3 = delete last appended word");
      System.out.print("Enter option: ");
      option = sc.nextInt();
      switch (option) {
        case 0:
          System.out.println("-- End --");
          System.exit(0);
        case 1:
          if (commandStack.size() == 0) {
            System.out.println("No command to undo");
            break;
          }
          System.out.println("\nUndo Last Command");
          command = (String) commandStack.pop();
          if (command.equals("Append")) {
            currentIndex = (int) commandStack.pop();
            p.removeWordAt(p.getWordSize() - 1);
          }
          if (command.equals("Delete")) {
            undoWord = (String) commandStack.pop();
            if(undoWord.length()>0) p.appendWord(undoWord);
          }
          break;
```

QUESTION Q2 CONTINUES ON THE NEXT PAGE

QUESTION Q2 CONTINUES FROM THE PREVIOUS PAGE

```
          case 2:
            System.out.print("\nAppend word: ");
            word = sc.next();
            p.appendWord(word);
            commandStack.push(p.getWordSize() - 1);
            commandStack.push("Append");
            break;

          case 3:
            System.out.println("\nDelete last appended word");
            word = p.removeWordAt(p.getWordSize() - 1);
            commandStack.push(word);
            commandStack.push("Delete");
            break;
          default:
            System.out.println("Invalid Option");
        }
        System.out.println("Paragraph Content: " + p);
      }
  }//main
}//ParagraphEditor

import java.util.Enumeration;
import java.util.Vector;

public class Paragraph {
  private Vector _content;

  public Paragraph() {
    _content = new Vector();
  }

  public String toString() {
    String showContent = "";
    Enumeration content = _content.elements();
    while (content.hasMoreElements()) {
      showContent += content.nextElement().toString()+" ";
    }
    return showContent;
  }

  public void appendWord(String word) {
    _content.add(word);
  }
```

QUESTION Q2 CONTINUES ON THE NEXT PAGE

QUESTION Q2 CONTINUES FROM THE PREVIOUS PAGE

```
    public String removeWordAt(int index) {
      if (index<0) return "";
      return (String) _content.remove(index);
    }

    public int getWordSize(){
        return _content.size();
    }
}//Paragraph
```

The following is a sample run of the program:

```
0 = exit, 1 = undo, 2 = append word,
3 = delete last appended word
Enter option: 2

Append word: One
Paragraph Content: One

0 = exit, 1 = undo, 2 = append word,
3 = delete last appended word
Enter option: 2

Append word: Two
Paragraph Content: One Two

0 = exit, 1 = undo, 2 = append word,
3 = delete last appended word
Enter option: 3

Delete last appended word
Paragraph Content: One

0 = exit, 1 = undo, 2 = append word,
3 = delete last appended word
Enter option: 2

Append word: Three
Paragraph Content: One Three
```

QUESTION Q2 CONTINUES ON THE NEXT PAGE

QUESTION Q2 CONTINUES FROM THE PREVIOUS PAGE

```
0 = exit, 1 = undo, 2 = append word,
3 = delete last appended word
Enter option: 1

Undo Last Command
Paragraph Content: One

0 = exit, 1 = undo, 2 = append word,
3 = delete last appended word
Enter option: 1

Undo Last Command
Paragraph Content: One Two


0 = exit, 1 = undo, 2 = append word,
3 = delete last appended word
Enter option: 1

Undo Last Command
Paragraph Content: One

0 = exit, 1 = undo, 2 = append word,
3 = delete last appended word
Enter option: 1

Undo Last Command
Paragraph Content:

0 = exit, 1 = undo, 2 = append word,
3 = delete last appended word
Enter option: 1
No command to undo
Paragraph Content:

0 = exit, 1 = undo, 2 = append word,
3 = delete last appended word
Enter option: 0
```

QUESTION Q2 CONTINUES ON THE NEXT PAGE

QUESTION Q2 CONTINUES FROM THE PREVIOUS PAGE

James has shown his program to his supervisor, Alan. Alan recommended him to use the ***Command Pattern*** to handle the operations '**2 = append word**' and '**3 = delete last appended word**' on the menu. In addition, an ***undo*** function should be implemented to correct mistaken operations.

You are asked to help James to re-design the program. The program should support the ***undo*** operation. Draw a class diagram to show the new design of the program. The class diagram is only required to support the existing operations: append a word and delete last appended word from a paragraph. You must show all the necessary classes, interface(s), association(s) and constructors, methods and attributes of the ***command interface*** and classes implementing the ***command interface*** in your class diagram.

[Remark: attributes and methods for ***Paragraph*** *and* ***ParagraphEditor*** *classes* ***ARE NOT*** *required]*

[16 marks]

(b) With reference to the class diagram given in your answer to **Q2 part (a)**, write Java code for implementing the ***command interface*** and the class which implements the ***command interface*** and provides the ***delete last appended word*** operation.

[9 marks]

Q3 (a) Write a few sentences to describe the purpose of Adapter Design Pattern. [2 marks]

(b) State **ONE** difference between Class Adapter and Object Adapter. [2 marks]

(c) John implements the ***Book*** class and ***DisplayBook*** for storing the book details and displaying the book's information respectively. Later, he finds there is existing class ***Comic*** class that stores the comic book information. After that, he requests his subordinate Mary to apply the Adapter Design Pattern and implement the Adapter class to let the ***DisplayBook*** class to display a ***Comic*** object's information. However, Mary is not allowed to change the code of all existing classes. You are asked to help Mary and apply the Object Adapter Design Pattern to design a solution for allowing the ***DisplayBook*** class to display the information of a ***Comic*** object. Draw a class diagram to show your design. Show all classes, interface(s), methods of the classes/interface(s), and relationships between classes/interface(s) in your diagram.

[7 marks]

```
public class Book {
  private String bookTitle;
  private String bookSubject;
  public Book (String title, String subject) {
    this.bookTitle = title;
    this.bookSubject = subject;
  }
  public Book () { this("Dummy","Dummy"); }
  public String getBookTitle ( )  { return bookTitle; }
  public String getBookSubject ( )  { return bookSubject; }
}//Book

public class DisplayBook {
  public static void displayBook(Book bk) {
    System.out.println("Book info : [" +
    bk.getBookTitle() + " / " + bk.getBookSubject() + "]");
          }
}//DisplayBook

public class Comic{
  private String title;
  private String subject;
  public Comic (String title, String subject) {
    this.title = title;
    this.subject = subject;
  }
  public String getTitle ( )  { return title; }
  public String getSubject ( )  { return subject; }
}//Comic
```

QUESTION Q3 CONTINUES ON THE NEXT PAGE

QUESTION Q3 CONTINUES FROM THE PREVIOUS PAGE

(d) Write Java code for implementing the adapter class in your answer to **Q3 part (c)**. [9 marks]

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

Q4 (a) Briefly describe the Open-Closed Principle. [3 marks]

(b) Jason is a junior programmer working in a chocolate factory, which only sells two types of chocolate: coffee and hazelnut. The price and discount of each type are different. Jason has written a program to calculate the individual price for each type of chocolate and print out total in a sales invoice. The program code is shown below:

```
public class Chocolate {
  private int type, qty;
  private String code;
  public static final int HAZELNUT = 0;
  public static final int COFFEE = 1;
  public Chocolate(int type, int qty, String code) {
    this.type = type; this.qty = qty; this.code = code;
  }
  public String getCode() { return code; }
  public double getAmt() {
   if (type == HAZELNUT) {
       double price = 20;
       if (qty < 10) { // no discount
         return price * qty;
       } else if (qty >= 10 && qty < 100) {
         return price * qty * .9;
       } else {
         return price * qty * .8;
       }
    } else if (type == COFFEE) {
       double price = 25;
       return  qty <= 100 ? price * qty : price * qty * .85;
    }else { return 0;}
  }
}// Chocolate

import java.util.List;
public class ChocStore {
    public static void checkout(List<Chocolate> items) {
        double total = 0;
        for (Chocolate c : items) {
            double amt = c.getAmt();
            System.out.println("Price for " + c.getCode() +
               " is HKD " + amt);
            total += amt;
        }
        System.out.println("Total Price is HKD " + total);
    }
} // ChocStore
```

QUESTION Q4 CONTINUES ON THE NEXT PAGE

QUESTION Q4 CONTINUES FROM THE PREVIOUS PAGE

```
import java.util.ArrayList;
public class Main {
  public static void main(String[] args) {
    Chocolate item1 = new Chocolate(Chocolate.COFFEE,  30, "item1");
    Chocolate item2 = new Chocolate(Chocolate.HAZELNUT, 40, "item2");
    ArrayList<Chocolate> items = new ArrayList<Chocolate>();
    items.add(item1);
    items.add(item2);
    ChocStore.checkout(items);
  }
}// Main
```

The following is a sample run of the program:

```
C:\>java Main
Price for item1 is HKD 750.0
Price for item2 is HKD 720.0
Total Price is HKD 1470.0
```

His supervisor, Alan, tells Jason that new type of chocolate will promote to the market soon. Alan recommends him to modify the design of the program in order to handle any new type of chocolate, which has an own price and the discount calculation method. Briefly explain why a new design is needed. Alan suggests using abstract class for the new design. [2 marks]

(c) Apply the Open-Closed Principle to redesign the program in **Q4 part (b)**. Draw a class diagram to show your new design. Show all necessary constructor(s) (with parameter types), methods (with parameter types and return types), classes, interface(s) and relationships between the classes or interface(s) in your class diagram. [8 marks]

(d) Write the Java code for implementing the classes **except** the *Main* class which have been changed or added in the new class diagram given in your answer to **Q4 Part (c)**. [10 marks]

(e) For the new design described in the class diagram of your answer to **Q4 Part (c)**, write Java code for the implementation of the ***Main*** Class [2 marks]

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