Course Code and Title: CS27Q/COMP2160 – Object-Oriented Programming

Date: Friday, May 13, 2011

Duration: 2 Hours.

Time: 9:00 a.m. – 12:00 noon.

Paper No: 1 of 1

Materials required:

Answer booklet: Normal ☑ Special □ Not required □

Calculator: Programmable □ Non Programmable □

Multiple Choice answer sheets: numerical □ alphabetical ☑ 1-20 ☑ 1-100 □

Auxiliary/Other material(s) – Please specify: MCQ Cards

Candidates are permitted to bring the following items to their desks:

Instructions to Candidates: This paper has 11 pages & 3 questions.

DO ALL QUESTIONS.

QUESTION ONE (1) CONSISTS OF TEN (10) MULTIPLE CHOICE QUESTIONS, WHICH MUST BE ANSWERED ON MCQ CARDS.

Pages 10 and 11 contain a list of common Java calls.

STUDENTS MAY NOT REMOVE THE QUESTION PAPER FROM THE EXAM ROOM.

Candidates are reminded that the examiners shall take into account the proper use of the English Language in determining the mark for each response.

SEMESTER II 2010/2011
Question 1 [10 Marks]

MULTIPLE CHOICE. Choose the one alternative that best completes the statement or answers the question.

1.1) Which of the following will find the sum of the main diagonal (i.e. $a[0][0] + a[1][1] + ...$) for array $a$ for an $n$ by $n$ array?

A) public double sumDiagonal(double a[][], int n)
   {
       double sum = 0;
       for (int i = 0; i < n; i++)
           for (int j = i; j < n; j++)
               sum += a[i][j];
       return sum;
   }

B) public double sumDiagonal(double a[][], int n)
   {
       double sum = 0;
       for (int i = 0; i < n; i++)
           for (int j = 0; j < n; j++)
               sum += a[i][j];
       return sum;
   }

C) public double sumDiagonal(double a[][], int n)
   {
       double sum = 0;
       for (int i = 0; i < n; i++)
           sum += a[i][i];
       return sum;
   }

D) public double sumDiagonal(double a[][], int n)
   {
       double sum = 0;
       int j = n-1;
       for (int i = 0; i < n; i++, j--)
           sum += a[i][j];
       return sum;
   }

E) None of the above will work
1.2) What is printed by the main method in the following code?

```java
public abstract class Animal {
    abstract public void speak();
    public void locomote() {
        move();
    }
    public void move() {
        System.out.println("Move somehow");
    }
}

public abstract class Mammal extends Animal {
    public void speak() {
        System.out.println("Make Noise");
    }
    public void move() {
        System.out.println("Move with legs or swim");
    }
}

public class Cat extends Mammal {
    public void speak() {
        System.out.println("Meow!");
    }
}

public class Dog extends Mammal {
    public void speak() {
        System.out.println("Woof!");
    }
}

public class Inherit {
    public static void main(String[] args) {
        Mammal d = new Dog();
        Animal c = new Cat();
        c.speak();
        d.speak();
        d.locomote();
        c.move();
    }
}
```
A) Make Noise
   Woof!
   Move somehow
   Move with legs or swim
B) Meow!
   Woof!
   Move with legs or swim
   Move with legs or swim
C) Meow!
   Woof!
   Move somehow
   Move with legs or swim
D) Make Noise
   Woof!
   Move somehow
   Move somehow
E) None of the above

1.3) Given the following code, where $x = 0$ before the loop, what is the resulting value of $x$ after the for-loop terminates?

   ```
   for (int i=0; i<5; i++)
     for (int j=1; j<=2; j++)
       x += i;
   ```

   A) 8  B) 15  C) 30  D) 45  E) 10

1.4) Assume that count is 0, total is 20 and max is 1. The following statement will do which of the following?

   ```
   if (count := 0 || total/count > max) max = total / count;
   ```

   A) The condition short circuits so that there is no division by zero error when evaluating the condition, but the assignment statement causes a division by zero error
   B) The condition does not short circuit causing a division by zero error
   C) The condition short circuits and the assignment statement is not executed
   D) The condition will not compile because it uses improper syntax
   E) The condition does not short circuit and the assignment statement is executed causing a division by zero error

1.5) If you instantiate the subclass of an Abstract class, the object you wind up with

   A) is an object of a Concrete class
   B) is an Interface
   C) is a reference to an Object
   D) is also an object of an Abstract class
   E) can't exist unless the subclass implements all the abstract methods of the Abstract class

SEMESTER II 2010/2011
For Question 6, refer to the following incomplete definition of the Time class.

```java
public class Time implements Comparable {
    public int hours;    // 0 <= hours
    public int minutes;  // 0 <= minutes <= 59

    // constructor not shown

    public int compareTo(Object other) {
        Time oth = (Time) other;
        // missing code
    }

    // other methods not shown
}
```

1.6) Assume that the hours field is never negative, and the minutes field is always between 0 and 59. Which of the following can be used to replace the missing code in the body of the compareTo method to correctly complete that method?

A) return (hours < oth.hours && minutes < oth.minutes);

B) return (hours < oth.hours || minutes < oth.minutes);

C) if (hours < oth.hours || minutes < oth.minutes) return -1;
   if (hours > oth.hours || minutes > oth.minutes) return 1;
   return 0;

D) if (hours < oth.hours) return -1;
   if (hours > oth.hours) return 1;
   if (minutes < oth.minutes) return -1;
   if (minutes == oth.minutes) return 0;
   return 1;

E) if ((hours < oth.hours) && (minutes < oth.minutes)) return -1;
   if ((hours == oth.hours) && (minutes == oth.minutes)) return 0;
   return 1;

1.7) All classes in Java are directly or indirectly subclasses of the _______ class.
    A) this    B) String    C) Object    D) Wrapper    E) Reference
1.8) Which of the following Java statements could be used to compute the total number of items sold?
   A) for (int j=0; j<12; j++) sum += sales[j];
   B) for (int j=0; j<=12; j++) sales[j] = sum;
   C) for (int j=0; j<12; j++) sum -= sales[j];
   D) for (int j=0; j<=12; j++) sum += sales[j];
   E) for (int j=0; j<12; j++) sales[j] += sum;

1.9) Which of the following is **not** essential when implementing Polymorphism?
   A) static methods
   B) overloading
   C) abstract classes
   D) abstract methods
   E) inheritance

1.10) Which of the following statements is true?
   i) If a static variable is changed, it changes for all objects of that class.
   ii) Attribute variables can be directly accessed by object methods and by static methods of that class.
   iii) Object methods can be directly invoked by static methods of that class.
   iv) A static variable can be used to share a value among objects of that class.

   A) i & ii
   B) i & iv
   C) ii & iii
   D) All of the above
   E) None of the above
**Question 2**  [20 Marks]

Refer to the following classes:

```java
public class Item
{
    private int itemNum; // item number
    private String name; // item name
    private double price; // item price
    private int qty; // quantity in stock

    public Item(int i, double p, int q, String n) {
        itemNum = i;
        price = p;
        qty = q;
        name = n;
    }

    public int getItemNo() { return itemNum; }
    public String getName() { return name; }
    public double getPrice() { return price; }
}

public class ItemLookupPanel
{
    private final int IMAX = 100; // maximum number of items
    private Item allItems[] = new Item[IMAX]; // items in stock
    private int nItems = 0; // number of items in the array

    JLabel itemLabel = new JLabel("Item");
    JLabel nameLabel = new JLabel("Name");
    JLabel priceLabel = new JLabel("Price");

    JTextField itemBox = new JTextField(10);
    JTextField nameBox = new JTextField(10);
    JTextField priceBox = new JTextField(10);

    JButton submitBtn;

    public ItemLookupPanel() {
        // add code for Q.2(c) here
    }

    private class SubmitListener implements ActionListener
    {
        public void actionPerformed(ActionEvent event) {
            // add code for Q.2(d) here
        }
    }
}
```
(a) Write a method `returnStock(int amt)` for class `Item`. This method allows the return of `amt` items to the quantity in stock.

(b) Write a method `findItem(int itemNum)` for class `ItemLookupPanel`, which returns the item object with the matching item number in the array `allItems`, if it is found. Otherwise, it returns null.

(c) The `ItemLookupPanel` displays the information for an item when the item number is entered (as shown in the following screenshot), and the submit button is clicked:

![Item Lookup Panel Screenshot]

Complete the missing section of the `ItemLookupPanel` constructor to create the interface, by placing each of the components (labels, text fields, buttons) on the panel properly, creating any objects needed, and ensuring that the button invokes the retrieval when clicked. Do not use the `FlowLayout` layout manager.

(d) Complete the missing section of class `submitListener` to perform the retrieval and to display the item information. If the item number input by the user is not a valid number, display "error in number entered" via a pop-up message. If the item number is not in the array, display an appropriate pop-up message. You do not need to rewrite the `submitListener` code given.
Question 3 [20 Marks]

(a) Write a method `writeItems(String fileName)` for class `ItemLookupPanel` (defined in Q.2), which stores items from the array `allItems` into the text file `fileName`.

The number of items currently in the array is to be written as the first line of the file. The rest if the file is to be written in a format of one item per line, as follows:

```
itemNumber itemName price qtyInStock orderLevel
```

where items are separated by tabs. Don’t use the `toString()` method. Print an appropriate error message if a file error/exception occurs. Assume that all necessary imports have already been declared. [6]

(b) Write the body of the `compareTo(Object otherItem)` method below to implement the `Comparable` interface in the `Item` class. It should compare two `Item` objects using their names and returns an appropriate integer depending on the comparison. [3]

(c) Write a subclass of `Item` called `TaxableItem` that has a static variable `get`, initialised to 12.5%, and a (non-static) integer variable `reorderLevel`. It must contain a constructor:

```
public TaxableItem(int itemNo, double price, int qty, String name, int reorder)
```

Do not write any other methods for the `TaxableItem` class, apart from `reorderStock()` below. [4]

ii) Write a boolean method `reorderStock()` for `TaxableItem` that is true if the quantity in stock is below the `reorderLevel`. [2]

iii) Write a static method `setGCT(double newTax)` for `TaxableItem` that changes the `get` to `newTax`. [1]

(d) Write a method `totalStock()` for the `ItemLookupPanel` class that computes and returns the total dollar value of all the items in stock (i.e. total of all price*quantity) [4]
Java Library Calls and Sample Code that may be useful

// Sample class with import statements and main method
import java.util.*; // Scanner, other utility classes ...
import java.awt.*; // For GUI
import java.awt.event.*; // For GUI
import javax.swing.*; // For GUI

public class SampleApplicationClass
{
    public static void main(String[] args)
    {
        // code
    }
}

// Some I/O methods and classes
System.out.println(String s);
System.out.printlnln(String s);
Scanner scan = new Scanner(System.in);
int i = scan.nextInt();
double d = scan.nextDouble();
String s = scan.next(); // reads the next word
String s = scan.nextLine(); // reads the rest of the line

// opens a scanner on text file fileName.
Scanner scan = new Scanner(new File(String fileName));
scan.hasNextInt(); // returns true if another int exists
scan.hasNext(); // returns true if another character exists

// Some String methods
str.charAt(int i); // returns the ith character of str

// returns the starting index of aStr within str, or -1 if
// aStr is not found
str.indexOf(aStr);

// searching from position i, returns the starting index of
// aStr within str, or -1 if aStr is not found
str.indexOf(i, aStr);

// returns a new string of characters, from index i1 to i2-1
str.subStr(i1, i2);
str.toUpperCase(); // returns an uppercase copy of a string
str.trim(); // returns string without leading, trailing blanks

// Some Array and List methods
array.length; // returns the size of an array
arrayList.size(); // returns the number of items in a list
arrayList.add(item); // adds an item to an array list
arrayList.get(index); // returns the item at index, if it exists

Semester II 2010/2011
// Useful conversion methods (may throw exceptions)
Integer.parseInt(String s); // convert string to integer
Double.parseDouble(String s); // convert string to double
Integer.toString(int i); // convert integer to String
Double.toString(double d); // convert double to String

// Some GUI methods
panel.add(Component comp) // adds a component to a container

// Makes panel have a grid layout of 6 rows and 2 columns
panel.setLayout(new GridLayout(6, 2));

// Adds a titled border to a panel
panel.setBorder(BorderFactory.createTitledBorder("Title"));

// Makes a component or panel scrollable, and adds it to a
// panel
panel.add(new JScrollPane(componentOrPanel));

// Create a JList and make it display an array
JList aList = new JList(anArray);

// adds an ActionListener to a component, e.g. a Button
component.addActionListener(new ActionListener()
{
   public void actionPerformed(ActionEvent e)
   {
      // handle action event
      // add code ...
   }
}, Listener alistener);

// Action Listener format
private class MyListener implements ActionListener
{
   public void actionPerformed(ActionEvent e)
   {
      // handle action event
      // add code ...
   }
}

// returns the object that sent the event
event.getSource();

// Some JLabel and JTextField methods
labelOrField.setText(String s); // assign string value
labelOrField.getText(); // return contents

// Dialog Box
JOptionPane.showMessageDialog(null, "Some Message");

END OF QUESTION PAPER