JOMO KENYATTA UNIVERSITY OF AGRICULTURE AND TECHNOLOGY

BACHELOR OF SCIENCE IN INFORMATION TECHNOLOGY /

BACHELOR OF BUSINESS AND INFORMATION TECHNOLOGY

BSC STAGE 2 SEMESTER 1 / BBT YEAR 2 SEMESTER 2

EXAMINATION 2013

ICS 2201 OBJECT ORIENTED PROGRAMMING II

DATE: APRIL 2013 **TIME: 2 HOURS**

Instructions: Answer question ONE (compulsory) and any other TWO questions

QUESTION ONE (30 marks)

a) State and explain four advantages of Java as an object oriented programming language.
(4 marks)

Portable - applications written in the Java programming language are compiled into machine-independent bytecodes, they run consistently on any Java platform (Java *Virtual Machine ported onto various hardware-based platforms)* hence allow the same program to be executed on multiple operating systems.

Robust - The *Java Application Programming Interface* (API) is a large collection of ready-made software components that provide many useful capabilities. It is grouped into libraries of related classes and interfaces; these libraries are known as *packages*. It offers a wide array of useful classes ready for use in your own applications. The API spans everything from basic objects, to networking and security, to XML generation and database access, and more.

Distributed - contain built-in support for using computer networks and is designed to execute code from remote sources securely. The JDK software provides standard mechanisms such as the Java Web Start software and Java Plug-In software for deploying your applications to end users.

Simple - The Java programming language is easy to learn, encourages good coding practices, and automatic garbage collection helps you avoid memory leaks. Program written in the Java can be four times smaller than the same program written in C++.

Multi-Threaded - A single Java program can have many different threads executing independently and continuously. Three Java applets on the same page can run together with each getting equal time from the CPU with very little extra effort on the part of the programmer.

b) By use of relevant examples, state the difference between

i. Statement and a block.

A ***statement*** forms a complete unit of execution e.g. System.out.println("Hello World!");
A ***block*** is a group of zero or more statements between balanced braces and can be used anywhere a single statement is allowed. e.g.

```
if (condition) { // begin block 1
                    System.out.println("Condition is true.");
                } // end block one
else { // begin block 2
                    System.out.println("Condition is false.");
    } // end block 2
  }
}
```

ii. Superclass and subclass

Inheritance is the process by which objects of one general class acquire the properties of objects of another (derived) class. A derived class is a class defined by adding instance variables and methods to an existing class. The existing class that the derived class is built upon is called the base class, or **superclass**. A base class is also called a superclass, a parent class, and an ancestor class. A derived class is also called a **subclass**, a child class, and a descendant class

iii. Function overloading and overriding (6 marks)

Overloading a method name means giving the same name to more than one method within a class. To do this, you must ensure that the different method definitions have something different about their parameter lists i.e. either have different numbers of parameters or have corresponding parameters with differing types.
Overriding - In a derived class, if you include a method definition that has the same name, the exact same number and types of parameters, and the same return type as a method already in the base class, this new definition replaces the old definition of the method when objects of the derived class receive a call to the method.

c) Explain the following terms: (6 marks)

i. Inheritance

Inheritance is the process by which objects of one class acquire the properties of objects of another class. A derived class is a class defined by adding instance variables and methods to an existing class. Inheritance should define a natural is-a relationship between two classes. Private instance variables and Private methods in a base class are not inherited by a derived class

ii. Encapsulation

The wrapping up of data (instance variables) and functions (methods) into a single unit called class is known as encapsulation. The data is not accessible to the outside world, and only those functions which are wrapped in the class can access it. This insulation of the data from direct access by the program is data hiding or information hiding

iii. Polymorphism

Polymorphism means ability to take more than one form. An operation may exhibit different behaviours in different instances. The behaviours depend on the types of data used in the operation. Polymorphism plays an important role in allowing objects having different internal structures to share the same external interface.

d) State four differences between Java applications and Java applets programming. (3 marks)

- **Application**
  - Trusted (i.e., has full access to system resources)
  - Invoked by Java Virtual Machine (JVM, java), e.g. java HelloWorld
  - Should contain a main method, i.e., public static void main(String[] args)

- **Applet**
  - Not trusted (i.e., has limited access to system resource to prevent security breaches)
  - Invoked automatically by a java enabled web browser
  - Should be a subclass of class java.applet.Applet

e) State and explain the usage of any three access modifiers in java programming language. (6 marks)

Default Access: class with default access has *no modifier preceding it in the* declaration. Default means it will be accessible only from the classes inside package in which it is defined.

Access Levels

Modifier	Class	Package	Subclass	World
public	Y	Y	Y	Y
protected	Y	Y	Y	N
no modifier	Y	Y	N	N
private	Y	N	N	N

f) Java exception handling is managed by five keywords: state and discuss the role of each of the keywords. (5 Marks)

Program statements to monitor are contained within a **try** block, if an exception occurs within the **try** block, it is thrown. Code within **catch** block catches the exception and handles it. To manually throw an exception, the keyword **throw** is used. Any exception that is thrown out of a method must be specified as such by a **throws** clause. Any code that absolutely must be executed before a method returns is put in a **finally** block

QUESTION TWO (20 marks)

a) Distinguish between abstract classes and interfaces (4 marks)

An **abstract class** can never be instantiated. Its sole purpose is to be extended (subclassed).If even a single method is abstract, the whole class must be declared abstract. However, you can put non-abstract methods in an abstract class.

An **interface** is a contractor what a class can do, without saying anything about how the class will do it. Interfaces can be implemented by any class, from any inheritance tree. An interface can have only abstract methods. All interface methods are implicitly public and abstract. All variables defined in an interface must be public, static, and final i.e. interfaces can declare only constants, not instance variables. An interface can extend one or more other interfaces. An interface cannot implement another interface or class.

b) Explain the difference between the AWT components and the Swing Components using examples

- AWT – Abstract Windows Toolkit (java.awt package)
  - The older version of the components
  - Rely on "peer architecture"…drawing done by the OS platform on which the application/applet is running
  - Considered to be "heavy-weight"
- Swing (javax.swing package)
  - Newer version of the components
  - No "peer architecture"…components draw themselves
  - Most are considered to be "lightweight"

(6 marks)

c) Differentiate between the following layout managers (6 Marks)

i. GridLayout

Arranges components into rows and columns
In Frame's constructor:

```
setLayout(new GridLayout(rows,columns))
        OR
setLayout(new GridLayout(rows,columns,hgap,vgap))
```

Components will be added in order, left to right, row by row
Components will be equal in size
As container is resized, components will resize accordingly, and remain in same grid arrangement

ii. BorderLayout

Arranges components into five areas: North, South, East, West, and Center
In the constructor:

```
setLayout(new BorderLayout())
        OR
setLayout(new BorderLayout(hgap,vgap))
```

for each component:

```
add (the_component, region)
```

do for each area desired:
BorderLayout.EAST, BorderLayout.SOUTH, BorderLayout.WEST, BorderLayout.NORTH, or BorderLayout.CENTER
Behavior: when the container is resized, the components will be resized but remain in the same locations.
NOTE: only a maximum of five components can be added and seen in this case, one to each region.

iii. FlowLayout

Places components sequentially (left-to-right) in the order they were added
Components will wrap around if the width of the container is not wide enough to hold them all in a row.
Default for applets and panels, but not for frames
Options: left, center (this is the default), or right

Typical syntax: in your Frame class's constructor
setLayout(new FlowLayout(FlowLayout.LEFT)) OR
setLayout(new FlowLayout(FlowLayout.LEFT,hgap,vgap))

d) Using a code snippet show how to set the layout of a user interface. (4 marks)

Typical syntax: in your Frame class's constructor

```
setLayout(new FlowLayout(FlowLayout.LEFT))
 OR
setLayout(new FlowLayout(FlowLayout.LEFT,hgap,vgap))
```

QUESTION THREE (20 marks)

a) Explain the procedure for converting an applet to an application. (6 marks)

- Build a Java JApplet (or Applet). (Let's assume you name the class AppletToApplication).
- Change the name of your applet's init() method to be the same as the name of your class (in this case: AppletToApplication). This is now the constructor method for the class.
- Delete the word void in the header of your new AppletToApplication constructor, since a constructor has no return type.
- Alter the class header so that it extends Frame rather than Applet (or JApplet) .
- Create a new method called main.
- Delete the import for the class Applet, since it is now redundant.
- Add window methods (e.g., windowClosing to handle the event which is the user clicking on the close window button, and others).
- Make sure that the program does not use any of the methods that are special to the Applet class – methods including getAudioClip , getCodeBase , getDocumentBase , and getImage

b) State and briefly describe the four methods that define the structure of an Applet. (8 marks)

- init: This method is intended for whatever initialization is needed for your applet. It is called after the param tags inside the applet tag have been processed.
- start: This method is automatically called after the browser calls the init method. It is also called whenever the user returns to the page containing the applet after having gone off to other pages.
- stop: This method is automatically called when the user moves off the page on which the applet sits. It can, therefore, be called repeatedly in the same applet.
- destroy: This method is only called when the browser shuts down normally. Because applets are meant to live on an HTML page, you should not normally leave resources behind after a user leaves the page that contains the applet.
- paint: Invoked immediately after the start() method, and also any time the applet needs to repaint itself in the browser. The paint() method is actually inherited from the java.awt.

c) Write an applet that displays the string “This is my first applet” on a web browser on clicking a button. (6 Marks)

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class FirstApplet extends Applet implements ActionListener {
    private JLabel msg;
    private final String ButtonText = "Press Me";
    public void init() {
        Container contentHolder = getParent();
        contentHolder.setLayout(new BorderLayout(18, 18));
        msg = new JLabel("");
        contentHolder.add(msg, BorderLayout.CENTER);
        JPanel buttonPanel = new JPanel();
        JButton myButton = new JButton(ButtonText);
        myButton.addActionListener(this);
        buttonPanel.add(myButton);
        contentHolder.add(buttonPanel, BorderLayout.SOUTH);

    }
    public void actionPerformed(ActionEvent evt) {
        String command = evt.getActionCommand();
        if (ButtonText.equals(command)) {
            msg.setText("This is my first applet");
        }

    }
}
```

QUESTION FOUR (20 marks)

a) Write a program in java that accepts two user input numbers: number1 and number2, the program then compares the two numbers and outputs: number1 equals number2 or number1 is less than number2 or number1 is greater than number2 depending on the outcome. (5 marks)

```
package numbers;
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        int number1, number2;
        Scanner keyboard = new Scanner(System.in);
        System.out.println("Enter two integer numbers");
        number1 = keyboard.nextInt();
        number2 = keyboard.nextInt();

        if(number1 == number2)
        {
             System.out.println("The two numbers are equal");
        }
        else if(number1 < number2)
        {
            System.out.println("number1 is less than number2");
        }
         else

            System.out.println("number1 is less than number2");
        }
}
```

b) Write a program in java that accepts a user input number and checks if the number is even or odd; the program then outputs the appropriate message. (4 Marks)

```
package javaapplication13;
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        // TODO code application logic here
        int number;
        Scanner keyboard = new Scanner(System.in);
        System.out.println("Input a number ");
        number = keyboard.nextInt();
        if((number % 2)==0)
        {
            System.out.println("The number is even");
        }
        else
                System.out.println("The number is odd");
    }

}
```

c) Use an if-else statement to write a program in Java that awards students grades based on the following guidelines; A = 70-100, B = 60 – 69, C = 50-59, D = 40-49, F = 0-39. The program should request the user to input the mark attained by the student, and should return “an invalid input” error in case the user tries to input any value that is not in the range 0-100. (6 Marks)

```java
import java.util.Scanner;

public class Grades {

    public static void main(String[] args){
        Scanner input = new Scanner(System.in);
        int m;
        System.out.print("Enter the cat marks: ");
        m = input.nextInt();
            if (m>100||m<0){System.out.println("Invalid");}
            else if (m>=70){System.out.println("A");}
            else if (m>=60){System.out.println("B");}
            else if (m>=50){System.out.println("C");}
            else if (m>=40){System.out.println("D");}
            else if (m<=39){System.out.println("F");}

        }
}
```

d) Write a recursive program that accepts a user input number and calculates the factorial of the number.

(5 Marks)

```java
import java.util.Scanner;
public class Main {

   public static void main(String[] args) {
        int number;
        Scanner keyboard = new Scanner(System.in);
        System.out.println("Input the number to calculate its factorial");
        number = keyboard.nextInt();

        System.out.println(fact(number));
    }
    static int fact(int numb)
    {
        if(numb == 1)
            return 1;
        else
            return numb * fact(numb-1);
    }
}
```