UNESCO-NIGERIA TECHNICAL &
VOCATIONAL EDUCATION REVITALISATION
PROJECT-PHASE II

NATIONAL DIPLOMA IN

COMPUTER TECHNOLOGY

Introduction to Scientific Programming Using Java

COURSE CODE: COM121

PRACTICALBOOK

Version 1: December 2008

Table of contents

WEEK ONE PRACTICALS

Objectives:

a. Create a Java source file using a text editor e.g. note pad.
b. Know how to save a Java file using a text editor

Creating a Java program file using Note Pad.

A text editor is an application that allows text to be entered without any special formatting. Thus the document is stored as sequences of characters. When writing Java programs do not use a word processor such as Ms-Word because they include special formatting and header details that will result in errors if attempt is made to execute the file.

The operating system that was used for the implementation of this practical is Windows Vista, if the operating system in your institution is different, get the assistance of your Computer room Lab manager.

To activate Note Pad on windows follow these steps:

i. Click on the start button to open the windows start menu.

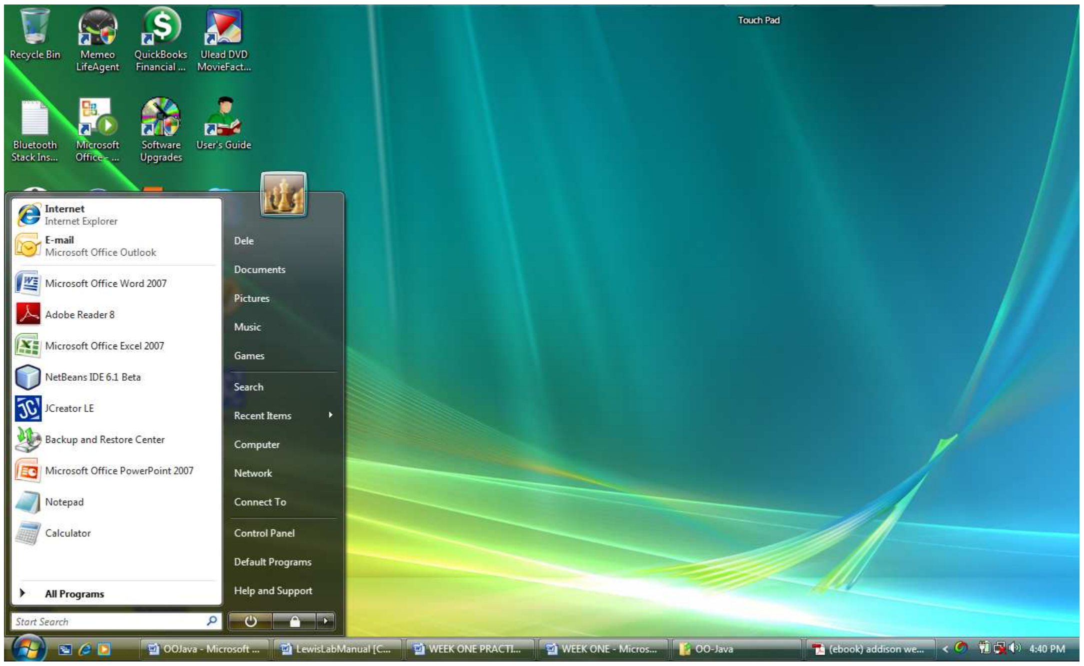


Figure 1.0 Opening the start menu

ii. Select All Programs from the menu.
iii. Select Accessories

iv. Click on Note Pad to load the program into memory.

On completion of these steps, you now proceed to enter the codes of Listing 1.0, reproduced below:

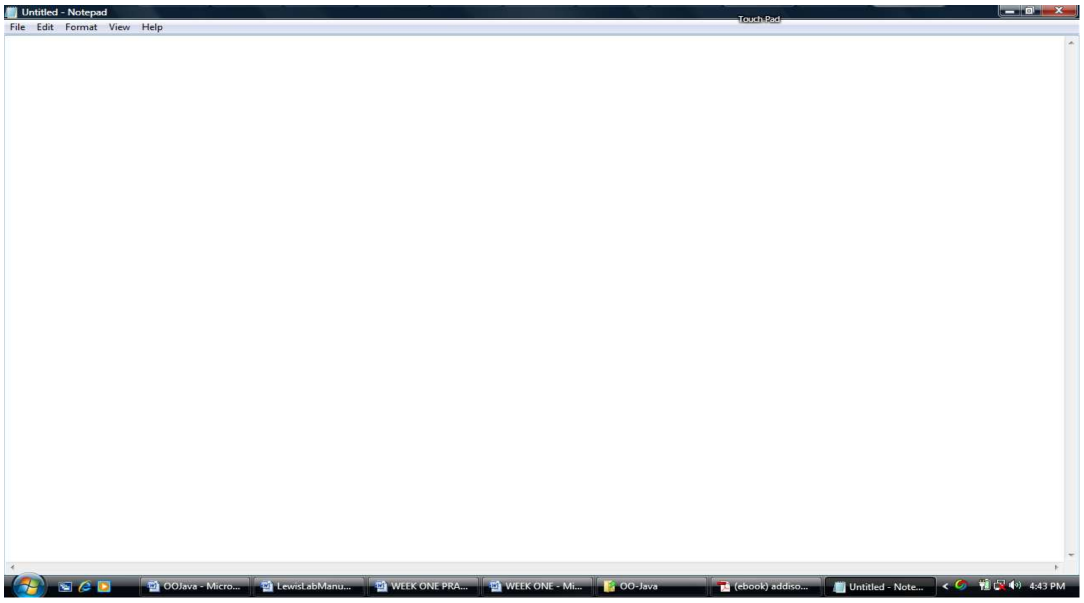


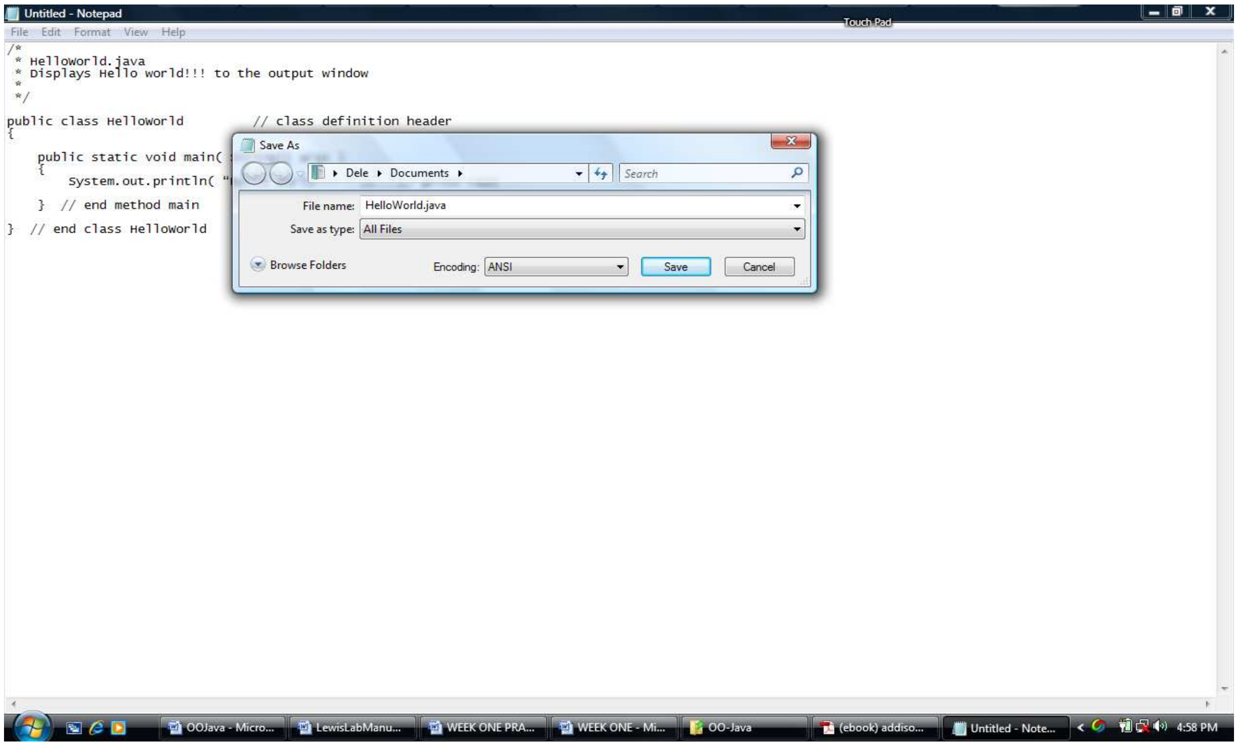
Figure 1.1 Note Pad Editor

```
/*
 * HelloWorld.java
 * Displays Hello world!!! to the output window
 *
 */

public class HelloWorld        // class definition header
{

    public static void main( String[] args )
    {
        System.out.println( “Hello World!!! “ );   // print text

    }  // end method main

}  // end class HelloWorld
```

Saving a Java program.

On completion of entering the program, follow these steps to save the program.

i. Open the File menu
ii. Select Save/Save As
iii. On the dialog box specify the folder you want to store your program e.g. JavaCodes.
iv. Type the name of the program in the File name text box: HelloWorld.java
v. Click the Save button to complete the process.



```
/*
 * HelloWorld.java
 * Displays Hello world!!! to the output window
 *
 */

public class HelloWorld          // class definition header
{

    public static void main(
    {
        System.out.println( "

    }  // end method main

}  // end class HelloWorld
```

File name: HelloWorld.java
Save as type: All Files
Encoding: ANSI

Figure 1.2 Saving the HelloWorld.java program.

Compiling a Java program.

Lab1.1

To compile a Java program follow these instructions:

i. Start a command prompt session and change to the directory / folder containing your Java source file. I have assumed that the file was saved in folder named JavaCodes, if the implementation on your computer is different substitute the folder name as appropriate.

C:\Java Codes>_

ii. Once this is done activate the Java compiler to convert the source code to byte codes.

    The Java compiler creates a corresponding file with an extension .class, e.g. HelloWorld.class

    C:\Java Codes>javac HelloWorld.java

    The java compiler is named javac the name of the java source code is HelloWorld.java. If the compilation was successful, the file HelloWorld.class will be created.

iii. To execute the byte codes activate the Java interpreter.

    C:\Java Codes>java HelloWorld

    The output produced should be:

    Hello World!!!

Practical for Week2

Specific Learning Objectives:

Objectives

Write a simple java program implementing input/output graphical boxes.

Key in the program below and answer the questions that follow by test running the program. Note that the line numbers are not part of the program. This is the same program presented in the lecture notes. Similar method will be applied for most of the practical work you will be required to complete in the subsequent weeks.

```java
 1 /*
 2  * HelloWolrdGUI.java
 3  *
 4  */
 5
 6
 7 import javax.swing.JOptionPane;
 8
 9 public class HelloWorldGUI {
10     public static void main(String[] args) {
11         String msg = "Hello Wolrd";
12         String ans ="";
13
14         JOptionPane.showMessageDialog(null, msg );
15
16         // accept the users name
17         ans = JOptionPane.showInputDialog( null, "Enter your Name Please"
);
18
19         // say hello to the user
20         JOptionPane.showMessageDialog(null, "Hello " + ans );
21
22     }  // end method main
23
24 }  // end of class HelloWorldGUI
```

Questions:

a. Describe the output of the program

b. Re-run the program and click on the ok button without entering a name. What is the output presented?

Week 3: Practicals.

Objectives

a. Implement Java Data Types in a simple addition program.
b. Implement data entry program that accepts data from the output window
c. Implement data entry program using input/output dialog boxes

In this practical exercise we will create two Java programs in order to see the difference between entering data from the windows prompt and using input/output dialog boxes. The first program is presented below:

Lab 3.1

```
/*
 * AddTwoNo.java
 * Add any two integer numbers
 */

import java.util.Scanner;

public class AddTwoNo {

    public static void main(String[] args) {
        // declare primitive variables
        int firstNumber = 10;
        int secondNumber = 20;
        int sum = 0;

        sum = firstNumber + secondNumber;

        system.out.println( "Sum of " + firstNumber + " and " +
                secondNumber + " is " + sum );

        // declare reference variable of type Scanner
        Scanner input = new Scanner( System.in );

        // Accept values from the user
        System.out.println( "Enter first integer number please ");
        firstNumber = input.nextInt();

        System.out.println( "Enter second integer number please ");
        secondNumber = input.nextInt();
        // calculate sum and display result
        sum = firstNumber + secondNumber;

        System.out.println( "Sum of " + firstNumber + " and " +
              secondNumber + " is " + sum );

    }  // end of method main

}  // end of class AddTwoNos
```

Listing 3.1 – AddTwoNos.java

Task:

a. Enter the program using a program editor (e.g Note Pad).
b. Debug any errors.
c. Using the values 10 and 11 to test the program. List the input and output interactive prompts and messages generated by the program.
d. Repeat task c, using the values -11 and 11.
e. Load the program and edit the declarative statements for the firstNumber and secondNumber variables to zero (0). Save and run the program. What is the first output results of computation displayed?
f. Do you thing any part of the program is now redundant? If yes edit out the redundant part otherwise give reasons for your answer.

Lab 3.2

```
/*
 * AddTwoNoDialog.java
 * Add any two integer numbers
 */

import javax.swing.JOptionPane;

public class AddTwoNoDialog {

    public static void main(String[] args) {
        // declare primitive variables
        int firstNumber = 10;
        int secondNumber = 20;
        int sum = 0;

        String input;   // for accepting input from the user
        String output;  // for displaying output

        // Accept values from the user
        input = JOptionPane.showInputDialog( null, "Enter first integer number",
              "Adding Integers", JOptionPane.QUESTION_MESSAGE );

        firstNumber = Integer.parseInt( input );

        input = JOptionPane.showInputDialog(null, "Enter second integer number",
              "Adding Integers", JOptionPane.QUESTION_MESSAGE );

        secondNumber = Integer.parseInt( input );

        // calculate sum and display result
        sum = firstNumber + secondNumber;

        // build output string
        output = "Sum of " + firstNumber + " and " + secondNumber + " is "
              + sum;

        // display output
        JOptionPane.showMessageDialog( null, output, "Adding two integers",
              JOptionPane.INFORMATION_MESSAGE );


    }  // end of method main

}  // end of class AddTwoNos
```

Listing 3.2 AddTwoNoDialog.java

Enter the program listing for Lab 3.2, then apply the questions in Lab 3.1 with this program.

WEEK FOUR – PROGRAM DEVELOPMENT PRACTICALS

Objectives

d. Understand the concept developmental techniques of program development.

Your first task is to enter the following program and compile it. Ensure that it is free from any errors and then proceed to answer the following questions.

```
/*
 * AverageThreeIntegers
 * Calculates the sumand average of any three integer numbers
 */

import javax.swing.JOptionPane;

public class AverageThreeIntegers
{
   public static void main( String args[] )
   {
       int firstNumber;    // first integer number
       int secondNumber;   // second integer number
       int thirdNumber;    // third integer number
       int sum;            // sum of the three numbers

       double average;     // average of the three numbers

       String input;       // input values
       String result;      // output generating string

       // Accept inteher numbers from he user
       input = JOptionPane.showInputDialog( null, "Enter first number: " );
       firstNumber = Integer.parseInt( input );    // wrap input to integer

       input = JOptionPane.showInputDialog( null, "Enter second number: " );
       secondNumber = Integer.parseInt( input );    // wrap input to integer

       input = JOptionPane.showInputDialog( null, "Enter third number: " );
       thirdNumber = Integer.parseInt( input );    // wrap input to integer

       // Calculate sum
       sum = firstNumber + secondNumber + thirdNumber;

       // Calculate average
       average = sum/3.0;
```

```
        // Build output string and display output
        result = "Average of " + firstNumber + ", " + secondNumber + " and " +
              thirdNumber + " is = " + average;

        JOptionPane.showMessageDialog( null, result, "Average of 3 Integers",
              JOptionPane.INFORMATION_MESSAGE );

    }  // end method main

}  // end class AverageThreeIntegers
```

Listing 4.1 AverageThreeIntegers.java

Questions:

a. Run the program using the integer numbers 3, 4 and 5 as input. What is the output?
b. Run the program use the integer numbers 9, 5 and 11 as input. What is the output? Is the output integer or floating point?
c. Amend the program and alter the statement: average = sum/3.0;
   Change the statement to **average = sum/3;** then re-run the program with the set of data supplied in questions a and b. What do you observe?
d. Run the program using the integer values -3, 3 and 0. What is/are the output generated?

WEEK FIVE: UNDERSTAND INSATIABLE CLASSES - PRACTICALS

Objectives

e. Know what an insatiable class is.
f. Know the difference between public and private data
g. Know the difference between local variables and instance variables
h. Understand parameter passing
i. Understand method return types

Lab 5.1

In this practical session you are expected to key in the programs presented and answer the questions presented at the end.

```
/*
 * Triangle.java
 *
 */

import javax.swing.JOptionPane;

public class Triangle
{
   // Declare instance variables - fields
   double base;     // base of triangle
   double height;   // height of triangle

   // No-arguement default constructor
   public Triangle()
   {
      base = 0;
      height = 0;
   }  // end no-argument default constructor

   // Programmer declared constructor
   public Triangle( double newBase, double newHeight )
   {
      base = newBase;
      height = newHeight;
   }  // end programmer declared constructor

   // Calcuate area of triangle
```

```
public double calculateArea()
{
    double area = 0;
    area = 0.5 * base * height;
    return area;
}  // end method calculateArea

// Dispay the area of the triangle
public void showArea()
{
    String output;
    output = "Base = " + base + "\nHeight = " + height +
            "\nArea = " + calculateArea();

    JOptionPane.showMessageDialog( null, output, "Calculate Area",
            JOptionPane.INFORMATION_MESSAGE );

}  // end method showArea

// Change the state of the base attribute of a Triangle object
public void setBase( double newBase )
{
    base = newBase;

}  // end method endBase

// Get the current state of the base of the Triangle object
public double getBase()
{
    return base;

}  // end method getBase

// Change the state of the height attribute of a Triangle object
public void setHeight( double newHeight )
{
    height = newHeight;

}  // end method endBase

// Get the current state of the height of the Triangle object
public double getHeight()
{
    return height;

}  // end method getBase
} // end class Triangle
```

Listing 5.1 – Triangle.java

Tasks:

a. Compile the above program and correct any compilation errors.
b. Attempt to run the program. What is the outcome?

Lab 5.2

```
/*
 * TriangleTest.java
 *
 */

import javax.swing.JOptionPane;

public class TriangleTest
{

    public static void main(String[] args) {
        // Declare local variables
        String answer;
        String result;

        double h = 0;   // height of triangle
        double b = 0;   // base of triangle

        // Create instance of class Triangle
        Triangle triangle1 = new Triangle();
        Triangle triangle2 = new Triangle( 4, 8 );

        //Display the area of triangle1
        triangle1.showArea();


        // Display area of triangle2
        triangle2.showArea();

        // accept new base and height for triangle1
        answer = JOptionPane.showInputDialog( null, "Enter Base", "Triangle1",
                JOptionPane.QUESTION_MESSAGE );
        b = Double.parseDouble( answer );

        answer = JOptionPane.showInputDialog( null, "Enter Height", "Triangle1",
            JOptionPane.QUESTION_MESSAGE );
        h = Double.parseDouble( answer );

        // set new values for triangle1
        triangle1.setBase( b );
        triangle1.setHeight( h );

        // Display new area
        result = "New Base = " + triangle1.getBase() +
            "\nNew Height = " + triangle1.getHeight() +
            "\nNew Area = " + triangle1.calculateArea();

        JOptionPane.showMessageDialog( null, result, "New Area - Triangle1",
            JOptionPane.INFORMATION_MESSAGE );

    }  // end method main

}  // end class TriangleTest
```

Listing 5.2 TriangleTest.java

Tasks:

a. Compile the above program and eliminate any compilation error.
b. Run the program and give a brief report the outcomes.
c. Write the instructions to alter the base and height values of triangle2 object to 3.4 and 10 respectively then display its area. Apply these instructions as the last instructions in the main method.
d. The base and height values were interchanged in tasks ‘c’ above will the output be the same? Try it out.
e. Update the Triangle.java class code (listing 5.1) and include a method for calculating the hypotenuse **– calculateHypothenus**. Recompile the class and ensure that there are no errors.
f. Write instructions to display graphically the base, height and hypotenuse of the triangle1 and triangle2 objects.

Practical for Week six

Specific Learning Objectives:

j. Create and execute JApplet class
k. Use the Appletviewer to execute a JApplet
l. Execute a JApplet using a web browser

Lab 6.1

In this practical session you are required to enter the program listing below and then create an html file to execute the JApplet. Thereafter answer the questions that follow:

```
/*
 * MyFirstApplet.java
 *
 */

import java.awt.Graphics;
import javax.swing.JOptionPane;
import javax.swing.JApplet;

public class MyFirstApplet extends JApplet
{
   public void init()
   {
      showStatus( "we are in init() method " );
      JOptionPane.showMessageDialog( null,
            "Check the status bar we are in init() " );
   }  // end method init

   public void start()
   {
      showStatus( "we are in start() method ");
      JOptionPane.showMessageDialog( null,
            "Check the status bar we are in start() " );
   }  // end method start

   public void paint(Graphics g)
   {
      super.paint( g );
      showStatus( "we are in paint() method ");
      JOptionPane.showMessageDialog( null,
            "Check the status bar we are in paint() " );
   }  // end method paint
```

```
    public void stop()
    {
        showStatus( "we are in stop() method ");
        JOptionPane.showMessageDialog( null,
                "Check the status bar we are in stop() " );
    }  // end method stop

    public void destroy()
    {
        showStatus( "we are in destroy() method ");
        JOptionPane.showMessageDialog( null,
                "Check the status bar we are in destroy() " );
    }  // end method destroy

}  // end class MyFirstJApplet
```

To create an html file follow these instructions.

i. Create a text file using note pad and store it in the folder where you have stored the MyFirstApplet.java file.
ii. Enter the following code:

```
<applet code=”MyFirstApplet.class” width=350 height=175>
</applet>
```

iii. Save the file as myfirstApplet.html
iv. To run the applet, open the windows explorer and double click on the file: myfirstapplet.html

Tasks:

a. Respond to the dialog boxes as they are being displayed.
b. When the paint() method prompt is displayed, does the next displayed when you respond to the dialog box?
c. Click the close button on the applet, what happened now?
d. Click the close button to end the applet program.

Lab 6.2

```
/*
 * SumTwoNumbers.java
 *
 */

import javax.swing.JApplet;
import java.awt.Graphics;
import javax.swing.JOptionPane;
public class SumTwoNumbers extends JApplet {

    public double sum;

    public void init() {
        double firstNumber = 0;
        double secondNumber = 0;

        String input;
        String output;

        // Prompt user for first number
        input = JOptionPane.showInputDialog( null, "Enter First Number: " );
        firstNumber = Double.parseDouble( input );

        // Prompt user for second number
        input = JOptionPane.showInputDialog( null, "Enter Second Number: " );
        secondNumber = Double.parseDouble( input );

        // calculate sum
        sum = firstNumber + secondNumber;
    }  // end method init

    public void paint( Graphics g )
    {

        super.paint( g );   // ensure that window is displayed properly

        String output;
        output = "The Sum is: " + sum;

        // Draw rectangle
        g.drawRect( 15, 10, 270, 20 );

        // Display the sum inside the rectangle
        g.drawString( output, 25, 25 );
    }  // end method paint

}  // end class SumTwoNumbers
```

Listing 6.2

Task:

a. Key in the program above – listing 6.2
b. Compile and ensure that it error free.
c. Create an appropriate html file and run the program
d. Using the appletviewer execute the file – to do this
   type: **appletviewer SumTwoNumbers**

   What differences do you observed from the execution of the applet using the appletviewer compared to using a web browser?

Practical: Week7

Specific Objectives:

a. Write programs to apply selection statements

Lab 7.1

Enter the program below and answer the questions below.

```
/*
 * Guess.java
 *
 */

package MyTrig;
import java.util.Random;
import javax.swing.JOptionPane;

public class Guess
{
   public static void main(String args[])
   {
      // Declare variables
      String answer;
      String output = "";

      int guessedNumber;

      // generate a random integer number between 1 and 10
      Random random = new Random();
      int randomNumber = random.nextInt( 10 ) + 1;

      // Accept an integer number from the user
      answer = JOptionPane.showInputDialog(null,
            "Guess an integer number between 1 and 10 inclusive",
            "Guessing Game", JOptionPane.QUESTION_MESSAGE );
      guessedNumber = Integer.parseInt( answer );

      if(guessedNumber != randomNumber )
         output = "Not Lucky! The number is " + randomNumber;
      else
         output = "You are correct!!!";
```

```
        JOptionPane.showMessageDialog(null, output, "Guessing Game",
              JOptionPane.INFORMATION_MESSAGE );


   }  // end method main

}  // end class Guess
```
Listing 7.1

Questions:

a. Compile the program and make sure it is error free.
b. What is the program doing?
c. Amend the program such that it allow the user to guess integer numbers from 10 to 20.

Practicals for: Week8

Specific Objectives:

b. Apply the while statement
c. Apply the do statement
d. Write simple programs to implement the while and do statements
e. Develop algorithms for solving simple repetitive problems – counter controlled and sentinel-controlled algorithms.
f. Applies a JTextArea and a JScrollPane class to display the numbers.

Lab 8.1

```
/*
 * TenNos.Java
 * Generates the First Ten Numbers and Calculate their Sum
 *
 */

import javax.swing.JOptionPane;
import javax.swing.JTextArea;
import javax.swing.JScrollPane;

public class TenNos
{

    public static void main(String[] args)
    {
        int sum = 0;
        int counter = 1;
        int n = 10;
        String nos = "";

        // Create JTextArea for displaying numbers
        JTextArea output = new JTextArea( 5, 20 );

        // Generate numbersn
        while(counter <= n ){
            output.append( counter + "\n" );    // add numbers to the JTextarea
            sum += counter;                     // calculate sum
            counter++;                          // increament counter by one
        }  // end while i <= n

        nos = "\nSum = " + sum;
        output.append(nos);

        // Append a JScrollpane to the JTextArea object
        JScrollPane outputArea = new JScrollPane( output );

        // Display numbers and their sum
        JOptionPane.showMessageDialog(null, outputArea,
                "Generate and Sums Numbers 1 - 10",
                JOptionPane.INFORMATION_MESSAGE);

    }  // end method main

}  // end of class TenNos
```

Questions:

a. Enter the program above, compile it and ensure that it is error free.
b. What does the program do?
c. Alter the statement JTextArea output = new JTextArea(5, 20); such that the actual parameter 5 and 20 becomes 20 and 5. Compile and run the program. What is the effect of the alteration?

Practicals for: Week9

Specific Objectives:

g. Understand the concepts of recursion
h. Write simple recursive methods

In this practical session will be required to enter the program below test it and answer the questions beneath the codes. You are advised to enter the codes in week 9 of the lecture notes and test them as well.

Lab 9.1a

```
/*
 * RecursiveNos.java
 *
 */

public class RecursiveNos
{
   public void display1To10( int startNumber )
   {
      if( startNumber == 10 ) {
         System.out.printf("\n%d ", startNumber );

      }  // number is equal to ten - base case
      else {
         System.out.printf("\n%d ", startNumber );  // number
         display1To10( startNumber + 1);

      }  // end else - number less than ten

   }  // end method dislay1To10

}  // end class RecursiveNos
```

Lab 9.1b

```
/*
 * RecursiveNosTest.java
 *
 */


public class RecursiveNosTest
{

    public static void main(String[] args) {
        RecursiveNos myNumbers = new RecursiveNos();

        System.out.println("Print number starting from one");
        myNumbers.display1To10( 1 );  // display the numbers starting from 1

        System.out.println("Print number starting from six");
        myNumbers.display1To10( 6 );  // display numbers starting from 6

    }  //end method main

}  // end class RecursiveNosTest
```

Questions:

a. What does the program do?
b. Write the instruction: myNumbers.display1To10(-2);

   What do you observe? Make suitable suggestions and implement same based on your findings.

Practical for Week10

Specific Objectives:

a. Describe and manipulate character type data
b. Differentiate between string and string buffer classes
c. Differentiate between equivalence and equality for string objects
d. Show how objects are passed to and returned from methods

Lab 10.1

Enter the program below and write the outputs displayed.

```
// Fig. 10.1 StringConstructors.java
// String class constructors.

public class StringConstructors
{
   public static void main( String args[] )
   {
     char charArray[] = { 'b', 'i', 'r', 't', 'h', ' ', 'd', 'a' , 'y' };
      String s = new String( "hello" );

      // use String constructors
      String s1 = new String();
      String s2 = new String( s );
      String s3 = new String( charArray );
      String s4 = new String( charArray, 6, 3);

      System.out.printf(
         "s1 = %s\ns2 = %s\ns3 = %s\ns4 = %s\n",
         s1, s2, s3, s4 ); // display strings
   } // end main
} // end class StringConstructors
```

Lab 10.2

Enter the program below and produce its output.

```
// Fig. 10.2: StringMiscellaneous.java
// This application demonstrates the length, charAt and getChars
// methods of the String class.

public class StringMiscellaneous
{
   public static void main( String args[] )
   {
      String s1 = "hello there";
      char charArray[] = new char[ 5 ];
```

```java

      System.out.printf( "s1: %s", s1 );

      // test length method
      System.out.printf( "\nLength of s1: %d", s1.length() );

      // loop through characters in s1 with charAt and display reversed
      System.out.print( "\nThe string reversed is: " );

      for ( int count = s1.length() - 1; count >= 0; count-- )
         System.out.printf( "%s ", s1.charAt( count ) );

      // copy characters from string into charArray
      s1.getChars( 0, 5, charArray, 0 );
      System.out.print( "\nThe character array is: " );

      for ( char character : charArray )
         System.out.print( character );

      System.out.println();
   } // end main
} // end class StringMiscellaneous
```

Lab 10.3

Enter the program below and write out its output

Figure 10.3. `String` comparisons.

```java
// Fig. 10.3: StringCompare.java
// String methods equals, equalsIgnoreCase, compareTo and regionMatches.

public class StringCompare
{
   public static void main( String args[] )
   {
      String s1 = new String( "hello" ); // s1 is a copy of "hello"
      String s2 = "goodbye";
      String s3 = "Happy Birthday";
      String s4 = "happy birthday";

      System.out.printf(
         "s1 = %s\ns2 = %s\ns3 = %s\ns4 = %s\n\n", s1, s2, s3, s4 );

      // test for equality
      if ( s1 equals( "hello" ) ) // true
         System.out.println( "s1 equals \"hello\"" );
      else
         System.out.println( "s1 does not equal \"hello\"" );

      // test for equality with ==
      if ( s1 == "hello" ) // false; they are not the same object
         System.out.println( "s1 is the same object as \"hello\"" );
      else
```

```
         System.out.println( "s1 is not the same object as \"hello\"" );

      // test for equality (ignore case)
      if ( s3.equalsIgnoreCase( s4 ) ) // true
         System.out.printf( "%s equals %s with case ignored\n", s3, s4 );
      else
         System.out.println( "s3 does not equal s4" );

      // test compareTo
      System.out.printf(
         "\ns1.compareTo( s2 ) is %d", s1.compareTo( s2 ) );
      System.out.printf(
         "\ns2.compareTo( s1 ) is %d", s2.compareTo( s1 ) );
      System.out.printf(
         "\ns1.compareTo( s1 ) is %d", s1.compareTo( s1 ) );
      System.out.printf(
         "\ns3.compareTo( s4 ) is %d", s3.compareTo( s4 ) );
      System.out.printf(
         "\ns4.compareTo( s3 ) is %d\n\n", s4.compareTo( s3 ) );

      // test regionMatches (case sensitive)
      if ( s3.regionMatches( 0, s4, 0, 5 ) )
         System.out.println( "First 5 characters of s3 and s4 match" );
      else
         System.out.println(
            "First 5 characters of s3 and s4 do not match" );

      // test regionMatches (ignore case)
      if ( s3. regionMatches( true, 0, s4, 0, 5 ) )
         System.out.println( "First 5 characters of s3 and s4 match" );
      else
         System.out.println(
            "First 5 characters of s3 and s4 do not match" );
     } // end main
} // end class StringCompare
```

Practicals for: Week11

Specific Objectives:

Write simple program that implement single-dimension arrays

Lab11.1

Enter the program below and answer the questions hat follow:

```
// Fig. 11.2: InitArray.java
// Creating an array.

public class InitArray
{
  public static void main( String args[] )
  {
    int array[]; // declare array named array

   array = new int[ 10 ]; // create the space for array

   System.out.printf( "%s%8s\n", "Index", "Value" ); //column headings

   // output each array element's value
   for ( int counter = 0; counter < array.length; counter++ )
     System.out.printf( "%5d%8d\n", counter, array[ counter ] );
  } // end main
} // end class InitArray
```

Questions:

a. What is the output of the program?
b. Re-write the program such that the values generated will be in reverse order.

Lab 11.2

```
// Fig. 7.3: InitArray.java
// Initializing the elements of an array with an array initializer.
```

```
public class InitArray
{
   public static void main( String args[] )
   {
      // initializer list specifies the value for each element
      int array[] = { 32, 27, 64, 18, 95, 14, 90, 70, 60, 37 };

     System.out.printf( "%s%8s\n", "Index", "Value" ); //column headings

       // output each array element's value
      for ( int counter = 0; counter < array.length; counter++ )
          System.out.printf( "%5d%8d\n", counter, array[ counter ] );
    } // end main
 } // end class InitArray
```

Questions:

a. What is the output of the program?

Practical for Week12

Specific Objectives:

i. Manipulate a set of data values using array of arrays
j. Declare and use array of arrays of primitive types

Lab 11.1

Enter the program below and answer the questions listed below.

```
// Fig. 12.2: InitArray.java
// Initializing two-dimensional arrays.

public class InitArray
{
   // create and output two-dimensional arrays
   public static void main( String args[] )
   {
      int array1[][] = { { 1, 2, 3 }, { 4, 5, 6 } };
      int array2[][] = { { 1, 2 }, { 3 }, { 4, 5, 6 } };

      System.out.println( "Values in array1 by row are" );
      outputArray( array1 ); // displays array1 by row

      System.out.println( "\nValues in array2 by row are" );
      outputArray( array2 ); // displays array2 by row
   } // end main

   // output rows and columns of a two-dimensional array
   public static void outputArray( int array[][] )
   {
      // loop through array's rows
      for ( int row = 0; row < array.length; row++ )
      {
         // loop through columns of current row
         for ( int column = 0; column < array[ row ].length; column++ )
            System.out.printf( "%d  ", array[ row ][ column ] );

         System.out.println(); // start new line of output
      } // end outer for
   } // end method outputArray
} // end class InitArray
```

Questioins:

a. What is the output of the program.

b. Write a program to print the contents of the array in reverse order.

Practical for Week 13

Specific Objectives:

Write simple event drive programs

Key in the programs below and run them

Lab 13.1

```
// Fig. 11.6: LabelFrame.java
// Demonstrating the JLabel class.
import java.awt.FlowLayout; // specifies how components are arranged
import javax.swing.JFrame; // provides basic window features
import javax.swing.JLabel; // displays text and images
import javax.swing.SwingConstants; // common constants used with Swing
import javax.swing.Icon; // interface used to manipulate images
import javax.swing.ImageIcon; // loads images

public class LabelFrame extends JFrame
{
  private JLabel label1; // JLabel with just text
   private JLabel label2; // JLabel constructed with text and icon
   private JLabel label3; // JLabel with added text and icon

   // LabelFrame constructor adds JLabels to JFrame
   public LabelFrame()
   {
      super( "Testing JLabel" );
      setLayout( new FlowLayout() ); // set frame layout

      // JLabel constructor with a string argument
      label1 = new JLabel( "Label with text" );
      label1.setToolTipText( "This is label1" );
      add( label1 ); // add label1 to JFrame

      // JLabel constructor with string, Icon and alignment arguments
      Icon bug = new ImageIcon( getClass().getResource( "bug1.gif" ) );
      label2 = new JLabel( "Label with text and icon", bug,
        SwingConstants.LEFT );
      label2.setToolTipText( "This is label2" );
      add( label2 ); // add label2 to JFrame

      label3 = new JLabel(); // JLabel constructor no arguments
      label3.setText( "Label with icon and text at bottom" );
      label3.setIcon( bug ); // add icon to JLabel
```

```
          label3.setHorizontalTextPosition( SwingConstants.CENTER );
          label3.setVerticalTextPosition( SwingConstants.BOTTOM );
          label3.setToolTipText( "This is label3" );
          add( label3 ); // add label3 to JFrame
       } // end LabelFrame constructor
    } // end class LabelFrame
```

Lab 13.2

```
   // Fig. 11.7: LabelTest.java
   // Testing LabelFrame.
    import javax.swing.JFrame;

    public class LabelTest
    {
       public static void main( String args[] )
       {
          LabelFrame labelFrame = new LabelFrame(); // create LabelFrame
          labelFrame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
          labelFrame.setSize( 275, 180 ); // set frame size
          labelFrame.setVisible( true ); // display frame
       } // end main
    } // end class LabelTest
```

Practical for Week 14

Specific Objectives:

a. Understand the concepts of event driven programs
b. Understand how to place objects on a frame
c. Write simple event drive programs

Lab 14.1

In this practical session we will develop an Employee inheritance hierarchy; this will involve us writing several programs. At the end of this practical, you will be required to develop an inheritance hierarchy of your choice and submit the full working version.

```java
// Fig. 14.1 CommissionEmployee.java
// CommissionEmployee class represents a commission employee.

public class CommissionEmployee extends Object
{
   private String firstName;
   private String lastName;
   private String socialSecurityNumber;
   private double grossSales; // gross weekly sales
   private double commissionRate; // commission percentage

  // five-argument constructor
  public CommissionEmployee( String first, String last, String ssn,
     double sales, double rate )
  {
     // implicit call to Object constructor occurs here
     firstName = first;
     lastName = last;
     socialSecurityNumber = ssn;
     setGrossSales( sales ); // validate and store gross sales
     setCommissionRate( rate ); // validate and store commission rate
  } // end five-argument CommissionEmployee constructor

  // set first name
  public void setFirstName( String first )
  {
     firstName = first;
  } // end method setFirstName

  // return first name
  public String getFirstName()
  {
     return firstName;
  } // end method getFirstName

  // set last name
  public void setLastName( String last )
  {
     lastName = last;
  } // end method setLastName

  // return last name
  public String getLastName()
  {
     return lastName;
  } // end method getLastName

  // set social security number
  public void setSocialSecurityNumber( String ssn )
  {
     socialSecurityNumber = ssn; // should validate
  } // end method setSocialSecurityNumber

  // return social security number
  public String getSocialSecurityNumber()
  {
     return socialSecurityNumber;
  } // end method getSocialSecurityNumber

  // set gross sales amount
  public void setGrossSales( double sales )
  {
     grossSales = ( sales < 0.0 ) ? 0.0 : sales;
  } // end method setGrossSales

  // return gross sales amount
  public double getGrossSales()
  {
     return grossSales;
  } // end method getGrossSales

  // set commission rate
  public void setCommissionRate( double rate )
  {
     commissionRate = ( rate > 0.0 && rate < 1.0 ) ? rate : 0.0;
  } // end method setCommissionRate

  // return commission rate
  public double getCommissionRate()
  {
     return commissionRate;
  } // end method getCommissionRate

   // calculate earnings
   public double earnings()
   {
      return commissionRate * grossSales;
   } // end method earnings

   // return String representation of CommissionEmployee object
   public String toString()
   {
      return String.format( "%s: %s %s\n%s: %s\n%s: %.2f\n%s: %.2f",
         "commission employee", firstName, lastName,
         "social security number", socialSecurityNumber,
         "gross sales", grossSales,
         "commission rate", commissionRate );
   } // end method toString
} // end class CommissionEmployee
```

Lab 14.2

```java
// Fig. 14.2: CommissionEmployeeTest.java
// Testing class CommissionEmployee.

public class CommissionEmployeeTest
{
   public static void main( String args[] )
   {
      // instantiate CommissionEmployee object
      CommissionEmployee employee = new CommissionEmployee(
         "Sue", "Jones", "222-22-2222", 10000, .06 );

      // get commission employee data
      System.out.println(
         "Employee information obtained by get methods: \n" );
      System.out.printf( "%s %s\n", "First name is",
         employee.getFirstName()  );
      System.out.printf( "%s %s\n", "Last name is",
         employee.getLastName()  );
      System.out.printf( "%s %s\n", "Social security number is",
         employee.getSocialSecurityNumber()  );
      System.out.printf( "%s %.2f\n", "Gross sales is",
         employee.getGrossSales() );
      System.out.printf( "%s %.2f\n", "Commission rate is",
         employee.getCommissionRate() );

      employee.setGrossSales( 500 ); // set gross sales
      employee.setCommissionRate( .1 ); // set commission rate

      System.out.printf( "\n%s:\n\n%s\n",
         "Updated employee information obtained by toString", employee );
   } // end main
} // end class CommissionEmployeeTest
```

Lab 14.3

```java
// Fig. 14.3: CommissionEmployee3.java
// CommissionEmployee3 class represents a commission employee.

public class CommissionEmployee3
{
   private String firstName;
   private String lastName;
   private String socialSecurityNumber;
   private double grossSales; // gross weekly sales
   private double commissionRate; // commission percentage

   // five-argument constructor
   public CommissionEmployee3( String first, String last, String ssn,
      double sales, double rate )
```

```java
     {
        // implicit call to Object constructor occurs here
        firstName = first;
        lastName = last;
        socialSecurityNumber = ssn;
        setGrossSales( sales ); // validate and store gross sales
        setCommissionRate( rate ); // validate and store commission rate
     } // end five-argument CommissionEmployee3 constructor

     // set first name
     public void setFirstName( String first )
     {
        firstName = first;
     } // end method setFirstName

    // return first name
    public String getFirstName()
    {
       return firstName;
    } // end method getFirstName

    // set last name
    public void setLastName( String last )
    {
       lastName = last;
    } // end method setLastName

    // return last name
    public String getLastName()
    {
       return lastName;
    } // end method getLastName

    // set social security number
    public void setSocialSecurityNumber( String ssn )
    {
       socialSecurityNumber = ssn; // should validate
    } // end method setSocialSecurityNumber

    // return social security number
    public String getSocialSecurityNumber()
    {
       return socialSecurityNumber;
    } // end method getSocialSecurityNumber

    // set gross sales amount
    public void setGrossSales( double sales )
    {
       grossSales = ( sales < 0.0 ) ? 0.0 : sales;
    } // end method setGrossSales

    // return gross sales amount
    public double getGrossSales()
    {
       return grossSales;
    } // end method getGrossSales

```

```java
   // set commission rate
   public void setCommissionRate( double rate )
   {
      commissionRate = ( rate > 0.0 && rate < 1.0 ) ? rate : 0.0;
   } // end method setCommissionRate

   // return commission rate
   public double getCommissionRate()
   {
      return commissionRate;
   } // end method getCommissionRate

   // calculate earnings
   public double earnings()
   {
      return getCommissionRate() * getGrossSales();
   } // end method earnings

   // return String representation of CommissionEmployee3 object
   public String toString()
   {
      return String.format( "%s: %s %s\n%s: %s\n%s: %.2f\n%s: %.2f",
         "commission employee", getFirstName(), getLastName(),
         "social security number", getSocialSecurityNumber(),
         "gross sales", getGrossSales(),
         "commission rate", getCommissionRate() );
    } // end method toString
} // end class CommissionEmployee3
```

Lab 14.4

```java
// Fig.  14.4: BasePlusCommissionEmployee4.java
//BasePlusCommissionEmployee4 class inherits from CommissionEmployee3 and
// accesses CommissionEmployee3's private data via CommissionEmployee3's
// public methods.

public class BasePlusCommissionEmployee4 extends CommissionEmployee3
{
   private double baseSalary; // base salary per week

   // six-argument constructor
   public BasePlusCommissionEmployee4( String first, String last,
      String ssn, double sales, double rate, double salary )
   {
      super( first, last, ssn, sales, rate );
      setBaseSalary( salary ); // validate and store base salary
   } // end six-argument BasePlusCommissionEmployee4 constructor

   // set base salary
   public void setBaseSalary( double salary )
   {
      baseSalary = ( salary < 0.0 ) ? 0.0 : salary;
   } // end method setBaseSalary

   // return base salary
   public double getBaseSalary()
   {
      return baseSalary;
   } // end method getBaseSalary

   // calculate earnings
   public double earnings()
   {
      return getBaseSalary() + super.earnings();
   } // end method earnings

   // return String representation of BasePlusCommissionEmployee4
   public String toString()
   {
      return String.format( "%s %s\n%s: %.2f", "base-salaried",
         super.toString(), "base salary", getBaseSalary() );
   } // end method toString
} // end class BasePlusCommissionEmployee4
```

Practical for: Week15

Specific Objectives:

a. Understand the concepts of polymorphism using class hierarchy
b. Know how to create abstract classes
c. Write abstract methods
d. Write simple programs implementing polymorphism

Lab 15.1

Enter the programs below and compile them until they are error free. At the end of this practical session, you will be expected to apply the skills acquired and develop Java programs implementing polymorphic principles.

```java
// Fig. 15.1: PolymorphismTest.java
// Assigning superclass and subclass references to superclass and
// subclass variables.

public class PolymorphismTest
{
   public static void main( String args[] )
   {
      // assign superclass reference to superclass variable
      CommissionEmployee3 commissionEmployee = new CommissionEmployee3(
         "Sue", "Jones", "222-22-2222", 10000, .06 );

      // assign subclass reference to subclass variable
      BasePlusCommissionEmployee4 basePlusCommissionEmployee =
         new BasePlusCommissionEmployee4(
         "Bob", "Lewis", "333-33-3333", 5000, .04, 300 );

      // invoke toString on superclass object using superclass variable
      System.out.printf( "%s %s:\n\n%s\n\n",
        "Call CommissionEmployee3's toString with superclass reference ",
         "to superclass object", commissionEmployee.toString() );

      // invoke toString on subclass object using subclass variable
      System.out.printf( "%s %s:\n\n%s\n\n",
         "Call BasePlusCommissionEmployee4's toString with subclass",
         "reference to subclass object",
         basePlusCommissionEmployee.toString() );

      // invoke toString on subclass object using superclass variable
      CommissionEmployee3 commissionEmployee2 =
         basePlusCommissionEmployee;
      System.out.printf( "%s %s:\n\n%s\n",
         "Call BasePlusCommissionEmployee4's toString with superclass",
         "reference to subclass object", commissionEmployee2.toString() );
   } // end main
} // end class PolymorphismTest
```

Lab 15.2

```
// Fig. 15.4: Employee.java
// Employee abstract superclass.

public abstract class Employee
{
   private String firstName;
   private String lastName;
   private String socialSecurityNumber;

   // three-argument constructor
   public Employee( String first, String last, String ssn )
   {
      firstName = first;
      lastName = last;
      socialSecurityNumber = ssn;
   } // end three-argument Employee constructor

   // set first name
   public void setFirstName( String first )
   {
      firstName = first;
   } // end method setFirstName

   // return first name
   public String getFirstName()
   {
      return firstName;
   } // end method getFirstName

   // set last name
   public void setLastName( String last )
   {
      lastName = last;
   } // end method setLastName

   // return last name
   public String getLastName()
   {
      return lastName;
   } // end method getLastName

   // set social security number
   public void setSocialSecurityNumber( String ssn )
   {
      socialSecurityNumber = ssn; // should validate
   } // end method setSocialSecurityNumber

   // return social security number
   public String getSocialSecurityNumber()
   {
      return socialSecurityNumber;
   } // end method getSocialSecurityNumber

   // return String representation of Employee object
   public String toString()
   {
      return String.format( "%s %s\nsocial security number: %s",
         getFirstName(), getLastName(), getSocialSecurityNumber() );
   } // end method toString

   // abstract method overridden by subclasses
   public abstract double earnings(); // no implementation here
} // end abstract class Employee
```