JAVA FUNDAMENTALS

Module 1

Ms. Azenith R. Mojica

Faculty, CSD

Table of Contents

Gospel

Feast of the Exaltation of the Holy Cross

Lectionary: 638

Reading 1

NM 21:4B-9

With their patience worn out by the journey,
the people complained against God and Moses,
"Why have you brought us up from Egypt to die in this desert,
where there is no food or water?
We are disgusted with this wretched food!"

In punishment the LORD sent among the people saraph serpents,
which bit the people so that many of them died.
Then the people came to Moses and said,
"We have sinned in complaining against the LORD and you.
Pray the LORD to take the serpents from us."
So Moses prayed for the people, and the LORD said to Moses,
"Make a saraph and mount it on a pole,
and if any who have been bitten look at it, they will live."
Moses accordingly made a bronze serpent and mounted it on a pole,
and whenever anyone who had been bitten by a serpent
looked at the bronze serpent, he lived.

Learning Outcomes

At the end of the module, you will be able to:

1. Apply the different ways to have your Input/Output in a Java program.
2. Use Java's I/O Commands via command prompt using the InputStreamReader class and Scanner Class.
3. Use the JOptionPane class to develop programs using the dialog box.
4. Apply the concept of Event-driven programming using Java's basic GUI methods and Classes.
5. Understand Java's File Input/Output commands and methods.
6. Use File I/O in developing programming solutions with File management.

Format of a Java Application

```
<comment>
<import statement>
<class definition>
{
      <method definition>
      {
            <body of the program>
      }
}
```

Example of a Java Application:

```
//My First Java Program
/* Azenith R. Mojica
     SY 2020-2021 */
import  java.lang.System;
import java.lang.String;
public class Example
{
      public static void main(String[] args)
      {
            System.out.println("Hello World!");
      }
}
```

Part 1: I/O via Command Prompt

Output Statement

- In Java, output on the standard output device is accomplished by using the **standard output object** *System.out.*

- The System.out has access to two methods: print and println.

System.out.print(expression);

- print – leaves the insertion point after the last character of the value of expression.

System.out.println(expression);

- println – positions the insertion point at the beginning of the next line.

Notes:

- ✓ When using Java's Output statements, plus sign (+) acts as the concatenation operator.
- ✓ The concatenation operator concatenates (join) the operands or values to be printed.

Example:

```
System.out.println("The sum of  2 and 3 = " + 5);
```

Input Statement 1

- Allows the user to input values via the command prompt
- Needs the following import statements:

```
import java.io.InputStreamReader;
import java.io.BufferedReader;
import java.io.IOException;
```

- Include throws IOException in the main method definition so that when compiled, Java acknowledges potential failures or errors

InputStreamReader

- Enables the user to enter values in the command prompt

Syntax:

```
InputStreamReader object = new InputStreamReader(System.in);
```

Example:

```
InputStreamReader myInput = new InputStreamReader(System.in);
```

BufferedReader

- Stores the value entered by the user to the buffer (temporary storage)

Syntax:

```
BufferedReader object = new BufferedReader(object of InputStreamReader);
```

Example:

```
BufferedReader  myBuff = new BufferedReader(myInput);
```

readLine

- Reads/ accepts values entered by the user
- Variable to hold the value should be declared as type String

Syntax:

```
Variable = object of BufferedReader.readLine();
```

Example:

```
String num1;
num1 = myBuff.readLine();
```

Integer.parseInt()

- From the package java.lang

```
import java.lang.Integer;
```

- Convert String to integer values that can be used for calculation

Syntax:

```
Variable Integer = Integer.parseInt(Variable String);
```

Example:

```
String num1;
int number1;

num1   = myBuff.readLine();
number1 = Integer.parseInt (num1);
```

Input Statement 2

- To put data into variables from the standard input device, Java provides the class Scanner.
- Needs the following import statement:

```
import java.util.Scanner;
```

Syntax:

```
Scanner object= new Scanner (System.in);
```

Example:

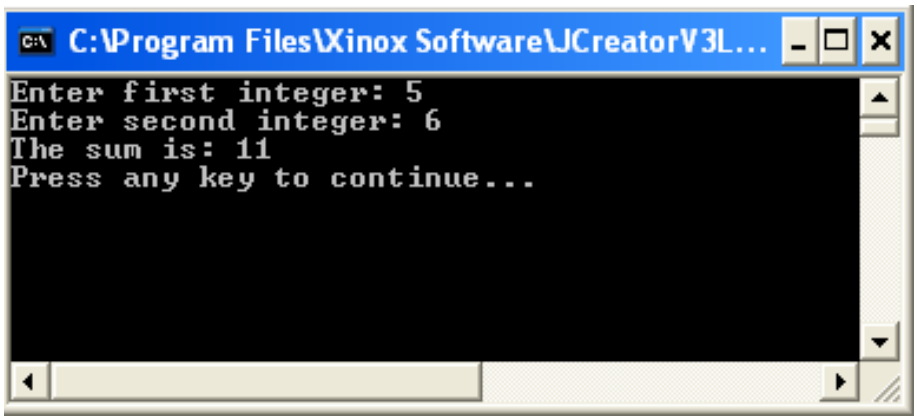
```
Scanner SCAN = new Scanner(System.in)
```

- If the next input token is an integer: **object.nextInt()**
- If the next input token is a floating-point number: **object.nextDouble()**
- If the next input token is a String: **object.next()**
- If the next input token is a String until the end of the line: **object.nextLine()**
- If the next input token is a single printable character: **object.next().charAt(0)**

Sample Code: Command Prompt

```
/* Example 1. USING STANDARD INPUT OBJECT
   Prepared by: AZENITH  ROLLAN- MOJICA */
import java.lang.System;
import java.lang.String;
import java.lang.Integer;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.IOException;
public class Example1
{
      public static void main(String args[]) throws IOException
      {
      InputStreamReader myInput = new InputStreamReader(System.in);
      BufferedReader myBuff = new BufferedReader(myInput);

      String num1, num2;
      int number1, number2, sum;
      System.out.print("Enter first integer: ");
      num1 = myBuff.readLine();
      number1 = Integer.parseInt(num1);
      System.out.print("Enter second integer: ");
      num2 = myBuff.readLine();
      number2 = Integer.parseInt(num2);
      sum = number1 + number2;
      System.out.println("The sum is: " + sum);
      }
}
```

OUTPUT:


C:\Program Files\Xinox Software\JCreatorV3L...
Enter first integer: 5
Enter second integer: 6
The sum is: 11
Press any key to continue...

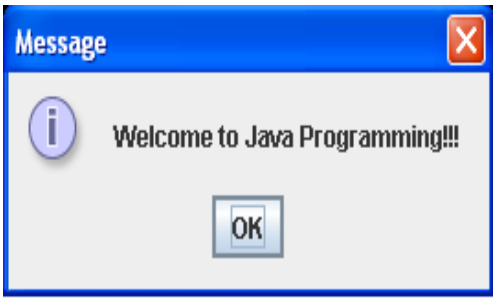
Part 2: I/O using the Dialog box

Output Statement

- Java provides the class ***JOptionPane*** which allows the programmer to use GUI (Graphical User Interface) components for I/O
- The class ***JOptionPane*** is contained in the package ***javax.swing***.
- The two methods of this class that we use are: ***showInputDialog*** and ***showMessageDialog***

showMessageDialog()

- The method showMessageDialog() allows the programmer to display information or results in the dialog box.

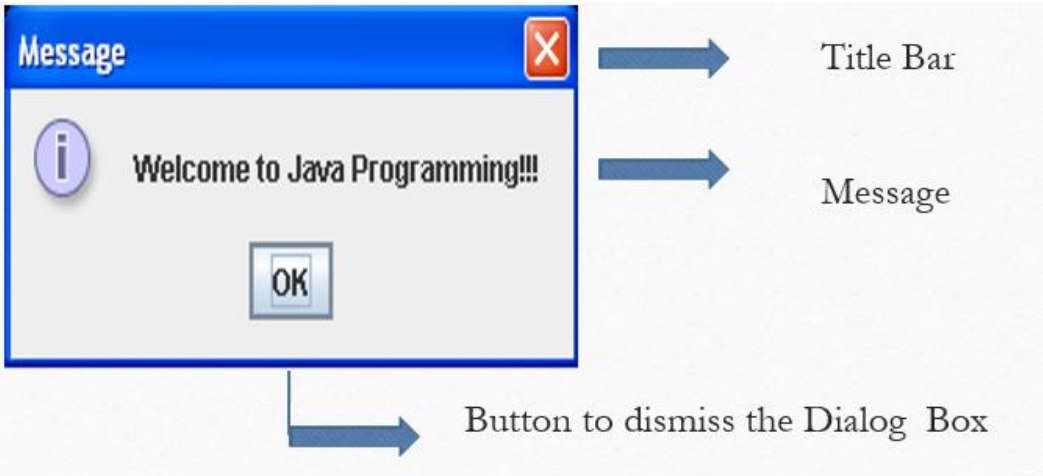


Syntax 1:

```
JOptionPane.showMessageDialog (null, "String to Display");
```

Example:

```
JOptionPane.showMessageDialog (null, "Welcome to Java Programming!!!");
```

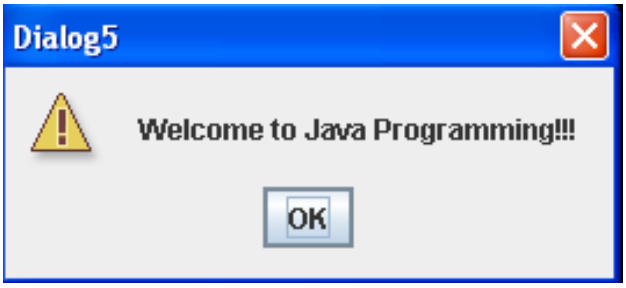


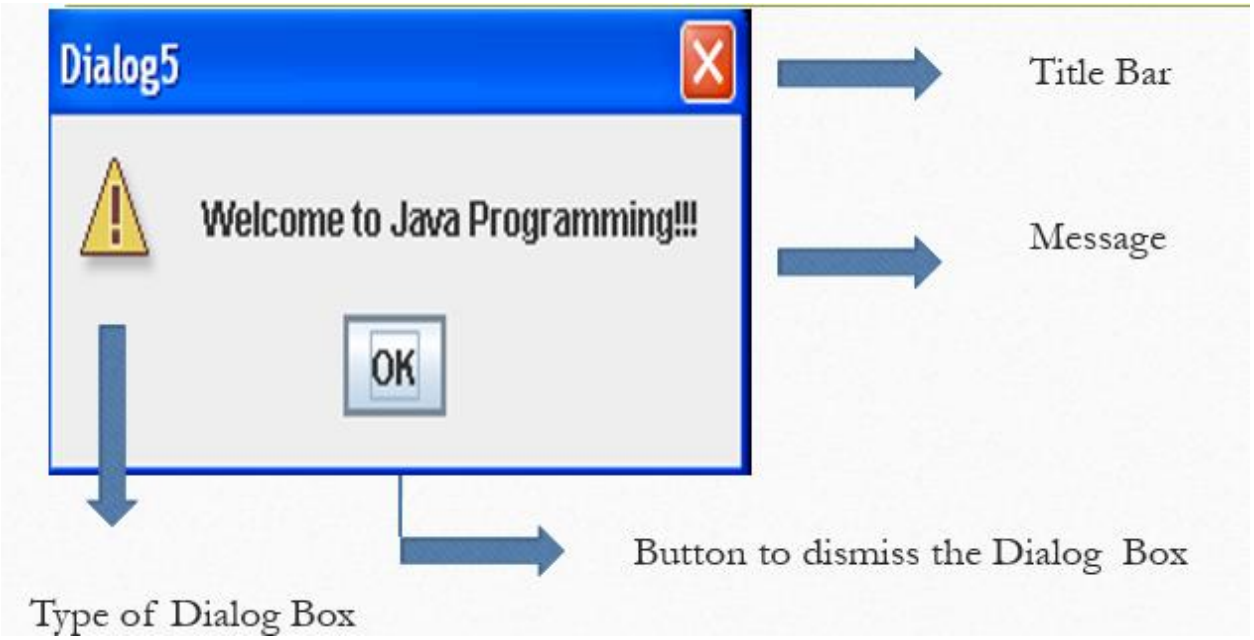
Syntax 2:

```
JOptionPane.showMessageDialog (null, "String to Display", "Text in
                Title Bar", Type of Message Dialog);
```

Example:

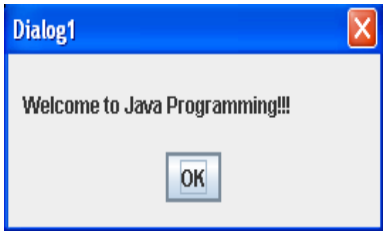
```
JOptionPane.showMessageDialog(null, "Welcome to Java Programming!!
                !","Dialog5", JOptionPane.WARNING_MESSAGE);
```





Types of Message Dialogbox

JOptionPane.PLAIN_MESSAGE



JOptionPane.WARNING_MESSAGE

JOptionPane.ERROR_MESSAGE

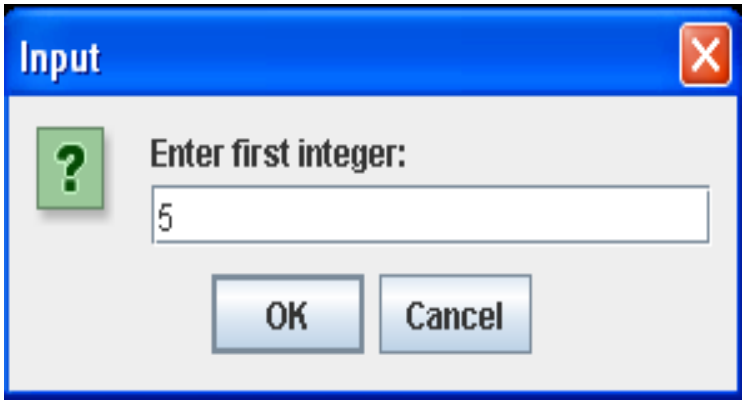
JOptionPane.INFORMATION_MESSA

JOptionPane.QUESTION_MESSAGE

Input Statement

showInputDialog()

- The method showInputDialog() allows the user to input a String from the keyboard.



Syntax:

```
Variable= JOptionPane.showInputDialog(null, "String to Display");
```

Example:

```
String num1;
int number1;
num1 = JOptionPane.showInputDialog(null,"Enter first
integer:");
number1 = Integer.parseInt(num1);
```

System.exit(0);

- Use the method exit of the class system to terminate a Java Application
- Required in any Java Application that display a GUI component
- Zero (0) indicates a successful termination

setDefaultLookAndFeelDecorated()

- From the package javax.swing and class JDialog

```
import javax.swing.JDialog;
```

- The method setDefaultLookAndFeelDecorated() is used to enhance the look of dialog boxes
- JDialog.setDefaultLookAndFeelDecorated(true);

Output Statement

JTextArea()

- A GUI component capable of displaying lines of text
- From the package javax. swing

```
import javax.swing.JTextArea;
```

Syntax:

```
JTextArea object = new JTextArea (rows, columns);
```

Example:

```
JTextArea outputArea = new JTextArea(5, 10);
```

Note:

JTextArea object should be attached to the JOptionPane object when displaying the dialog box.

append()

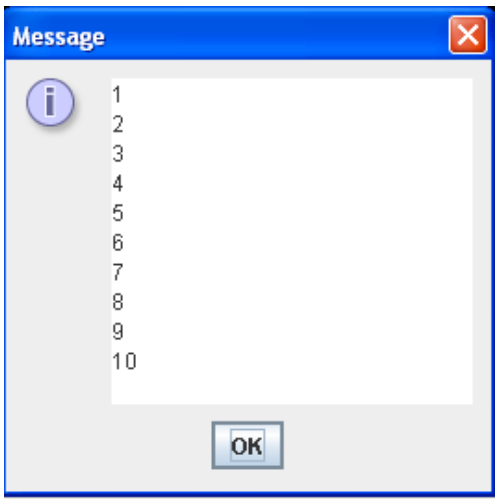
- Used to add more text to the end of the string that is already displayed in the JTextArea

Syntax:

```
object of JTextArea.append("text to print");
```

Example:

```
JTextArea outputArea = new JTextArea(5, 10);

for (x = 1; x<=10; x++)

   outputArea.append(x+"\n");

JOptionPane.showMessageDialog(null, outputArea);
```



setText()

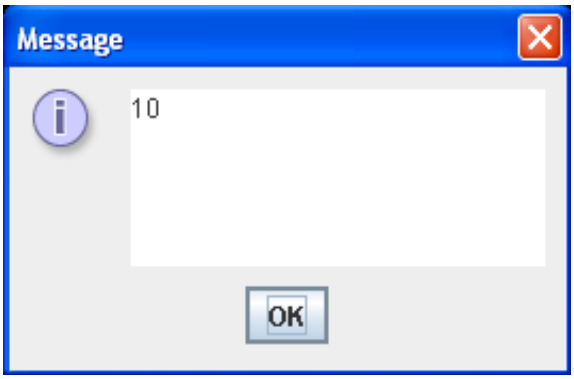
- Used to replace the text in the JTextArea with a new text printed by .setText()

Syntax:

object of JTextArea.setText("text to print");

Example:

```
 JTextArea outputArea = new JTextArea(5, 10);

 for (x = 1; x<=10; x++)

        outputArea.setText(x+"\n");

JOptionPane.showMessageDialog(null, outputArea);
```



JScrollPane()

- GUI component with a scrolling functionality
- From the package javax. swing
  import javax.swing.JScrollPane;
- Attached to the JTextArea

Syntax:

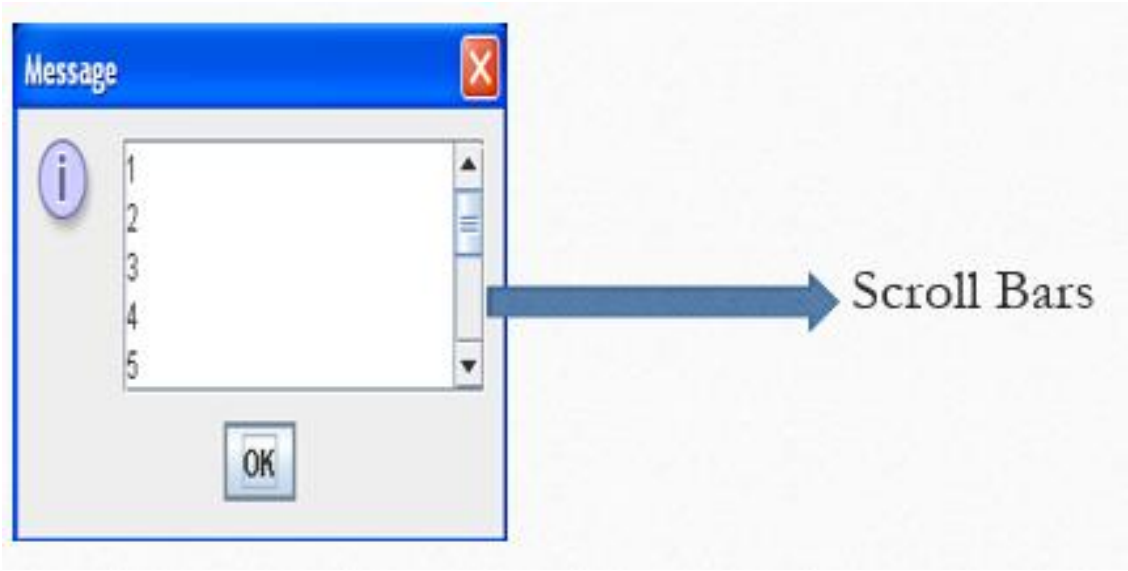
```
JScrollPane object = new JScrollPane (object of JTextArea);
```

Example:

```
JTextArea outputArea = new JTextArea(5, 10);
JScrollPane scroll = new JScrollPane (outputArea);
```

Note:

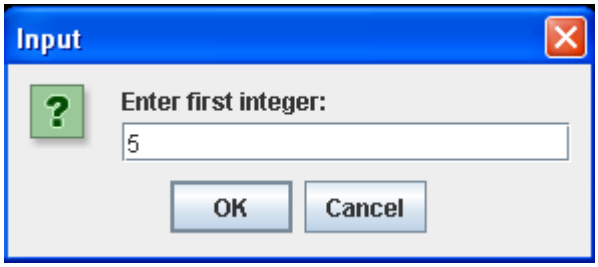
JScrollPane object should be attached to the JOptionpane object when displaying the dialog box.

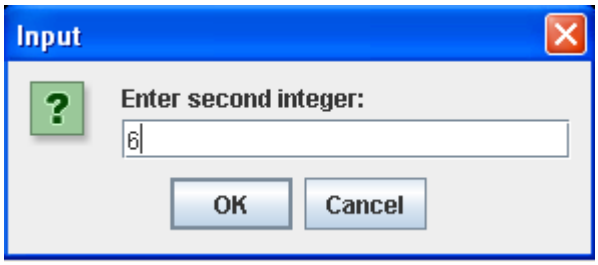


Sample Code: Dialog Box

```
/* Example 2. USING JOPTIONPANE FOR INPUT AND OUTPUT
   Prepared by: AZENITH  ROLLAN- MOJICA */

 import java.lang.System;
 import java.lang.String;
 import javax.swing.JOptionPane;
  public class Example2
 {
     public static void main(String args[])
     {
     String num1, num2;
     int number1, number2, sum;

     num1 = JOptionPane.showInputDialog(null,"Enter first integer:");
     number1 = Integer.parseInt(num1);

     num2 = JOptionPane.showInputDialog(null,"Enter second integer:");
     number2 = Integer.parseInt(num2);

     sum = number1 + number2;

     JOptionPane.showMessageDialog(null,"The sum is " + sum);

     System.exit(0);
     }
 }
```

OUTPUT:


Input
Enter first integer:
5
OK
Cancel


Input
Enter second integer:
6
OK
Cancel


Message
The sum is 11
OK

Part 3: I/O using Basic GUI

JLabel

- Contains a string of character to display on the screen.
- Normally indicates the purpose of another GUI component on the screen.

Syntax:

```
JLabel object_name;          // declaration
object_name  =  new JLabel ("String to display");
```

Jlabel Example

```
JLabel label1;
label1 = new JLabel ("Enter a value");
```

- To attach this to the content pane, use add method:

Syntax:

```
Object of Container. add(object of JLabel);
```

Example, if c is the content pane, to add the new JLabel label1, use:

```
c.add (label1);
```

JLabel Example

JTextField

- Used to get a single line of information from the user at the keyboard or display information on screen.

Syntax:

```
JTextField object_name;          // declaration
object_name = new JTextField (size);
```

JTextField Example

```
JTextField field1;
field1 = new JTextField (10);
```

- To attach this to the content pane, use add method:

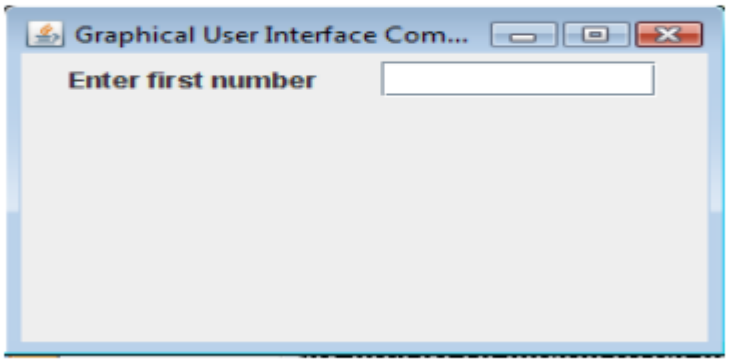
Syntax:

```
Object of Container. add(object of JTextField);
```

Example, if c is the content pane, to add the new JTextField field1, use:

```
c.add (field1);
```

JTextField Example



JButton

- Rectangular object that the user can press to perform an action

Syntax:

```
JButton object_name;            // declaration
object_name = new JButton ("Text on Button");
```

JButton Example

```
JButton button1;
button1 = new JButton ("Click here");
```

- To attach this to the content pane, use add method:

Syntax:

```
Object of Container. add(object of JButton);
```

Example, if c is the content pane, to add the new JButton button1, use:

```
c.add (button1);
```

JButton Example

.addActionListener(this);

- Specifies that the program should listen for events
- The keyword this enables the program to refer to itself
- When the user interacts with a GUI component an *event* is sent to the program

Example:

```
button1.addActionListener (this);
```

When the user press the button, an event is sent to the applet indicating that an action was performed by the user and calls the method **actionPerformed ( )** to process user interaction.

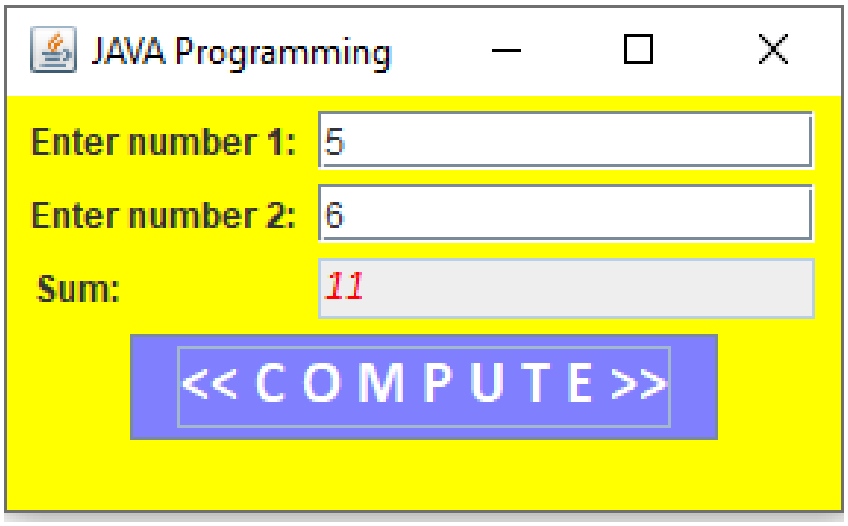
Sample Code: Basic GUI

```
/* Example 3. USING BASIC GUI
   Prepared by: AZENITH  ROLLAN- MOJICA */

import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.JButton;
import java.awt.Container;
import java.awt.FlowLayout;
import java.awt.Color;
import java.awt.Font;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

public class Example3 extends JFrame implements ActionListener
{
      JLabel L1, L2, L3;
      JTextField F1, F2, F3;
      JButton B;
      Font F = new Font("Calibri", Font.BOLD, 20);
      Font Fa = new Font("Calibri", Font.ITALIC, 14);

      public Example3 ()
      {
            Container C = getContentPane();
            C.setLayout(new FlowLayout());
            C.setBackground(Color.yellow);
```

```
L1 = new JLabel("Enter number 1: ");
C.add(L1);

F1 = new JTextField(15);
C.add(F1);
F1.setToolTipText("1st number!");

L2 = new JLabel("Enter number 2: ");
C.add(L2);

F2 = new JTextField(15);
C.add(F2);
F2.addActionListener(this);



L3 = new JLabel("Sum:                    ");
C.add(L3);

F3 = new JTextField(15);
C.add(F3);
F3.setEditable(false);
F3.setForeground(Color.red);
F3.setFont(Fa);

B = new JButton ("<< C O M P U T E >>");
C.add(B);
B.addActionListener(this);
B.setToolTipText("Click to display the sum...");
B.setBackground(new Color(128,128,255));
B.setForeground(new Color(255,255,255));
```

```
        B.setFont(F);

        setSize(300, 180);
        setTitle("JAVA Programming");
        setVisible(true);
    }



    public void actionPerformed (ActionEvent e)
    {
        int n1, n2, sum;

        n1 = Integer.parseInt(F1.getText());
        n2 = Integer.parseInt(F2.getText());

        sum = n1 + n2;
        F3.setText(String.valueOf(sum));

    }

    public static void main (String args[])
    {
        Example3 M = new Example3 ();
        M.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

}
```

OUTPUT:


JAVA Programming
Enter number 1:
5
Enter number 2:
6
Sum:
11
<< COMPUTE >>

Part 4: I/O using Files

FILES

"A **file** is an area in secondary storage used to hold information."

Input:

By using a file as a source of input data, you can prepare data before running a program, and the program can access the data each time it runs.

Output:

Saving output to a file allows the output to be saved and distributed to others, and the output produced by one program can be used as input to other programs

INPUT FROM A FILE

To input data from a file, you use the class Scanner and FileReader.

You need the following import statements:

```
import java.util.Scanner;
import java.io.FileReader;
```

Initialize a Scanner object to input a source from the class FileReader:

Scanner object = new Scanner (new FileReader("input file"));

Example:

Scanner inFile= new Scanner (new FileReader("input.txt"));

Scanner inFile= new Scanner (new FileReader("h:\\input.txt"));

INPUT FROM A FILE

Use the object of the Scanner class to input the data from the file, just like the way to input data from the standard input device using the methods next(), nextInt(), nextDouble(), etc.

Example:

```
Scanner inFile= new Scanner (new FileReader("input.txt"));

String name;
double grade;

name = inFile.next();
grade = inFile.nextDouble();
```

OUTPUT TO A FILE

To send the output to a file, use the class PrintWriter.

```
import java.io.PrintWriter;
```

Declare a PrintWriter variable and associate the variable with the destination, that is, the file where the output will be stored.

```
PrintWriter variable = new PrintWriter("output  file");
```

Example:

```
PrintWriter outFile= new PrintWriter("output.txt");
```

OUTPUT TO A FILE

Use the variable of PrintWriter class with the methods print and println .

Example:

```
PrintWriter outFile= new PrintWriter("output.txt");
outFile.println("The grade of     " + name + "is" + grade);
```

OUTPUT TO A FILE

Notes:

1) An output file does not have to exist before it is opened. If the output file does not exist, the computer prepares an empty file for output.

2) If the designated output file already exists, by default, the old contents are erased (lost).

3) If the program is not able to create or access the output file, it throws a FileNotFoundException. To solve, include throws FileNotFoundException in the main method definition.

```
import java.io.FileNotFoundException;
public static void main (String args[]) throws FileNotFoundException
```

CLOSING FILES

Close the input and output files.

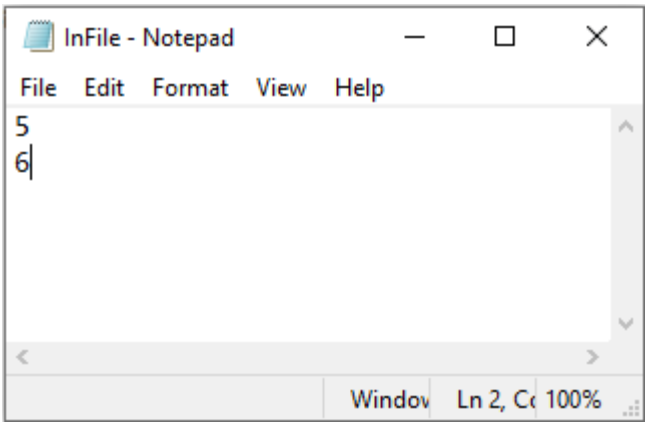
Use the close method in closing both the input and output files.

Example:

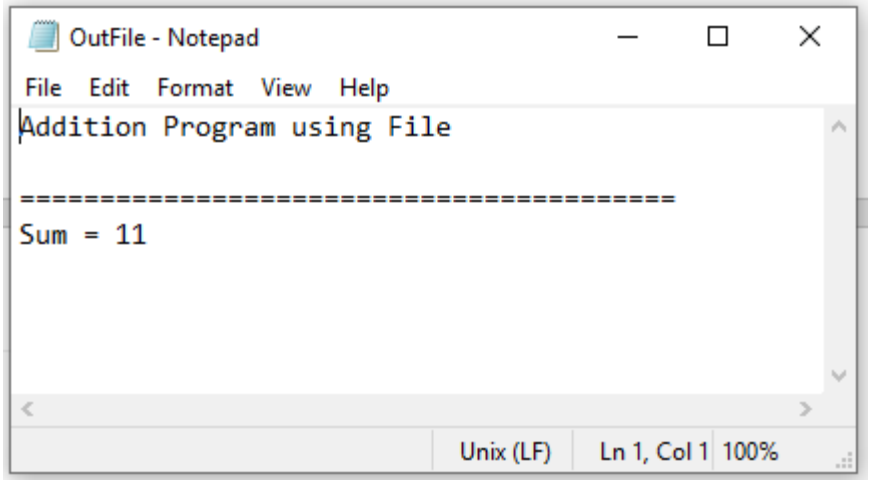
```
Scanner inFile= new Scanner (new FileReader("input.txt"));
PrintWriter outFile= new PrintWriter("output.txt");

inFile.close();
outFile.close();
```

Sample Code: File

Input File: InFile.txt



Output File: OutFile.txt



```
/* Example 4. USING FILE
   Prepared by: AZENITH  ROLLAN- MOJICA */

import java.io.FileReader;
import java.io.PrintWriter;
import java.util.Scanner;
import java.io.FileNotFoundException;

public class Example4
{
      public static void main (String args[]) throws FileNotFoundException
      {
            Scanner in = new Scanner (new FileReader("InFile.txt"));
            PrintWriter out = new PrintWriter("OutFile.txt");
            int x,y, sum;

            x = in.nextInt();
            y = in.nextInt();
            sum = x+y;

            out.println("Addition Program using File\n");
            out.println("========================================");
            out.println("Sum = "+ sum);
            in.close();
            out.close();

      }
}
```

REFERENCES AND ADDITIONAL READINGS

Java Tutorial | Learn Java

https://www.javatpoint.com/java-tutorial

Java Tutorial: Learn Java with Examples

https://beginnersbook.com/java-tutorial-for-beginners-with-examples/

Java Tutorial for Beginners

https://www.youtube.com/watch?v=eIrMbAQSU34

Java Programming All-in-One Tutorial Series (6 HOURS!)

https://www.youtube.com/watch?v=r3GGV2TG_vw

Java Full Course | Java Tutorial for Beginners | Java Online Training | Edureka

https://www.youtube.com/watch?v=hBh_CC5y8-s