



# INDIAN SCHOOL MUSCAT

## Some Points to Keep in Mind .....



- Please keep your **MIC and WEBCAM in MUTE mode** until your teacher asks you to unmute it.
- Please take down notes.
- Ask doubts as and when it comes and write in the **CHAT** box.
- Don't post any non-academic matter in the chat box. Stringent action will be initiated.
- Some times, technology fails, don't panic, hold on - we will be back.



# INDIAN SCHOOL MUSCAT



## CLASS XI

### Information Technology(802)

### **Chapter :** Fundamentals of Java Programming

### **Teacher:** Mr. Saju Jagannath



WRITE

**Object-oriented programming (OOP)** is a programming paradigm based on the concept of "objects", which can contain data and code: data in the form of fields (often known as *attributes* or *properties*), and code, in the form of procedures (often known as methods).

A feature of objects is that an object's own procedures can access and often modify the data fields of itself (objects have a notion of this or **self**). In OOP, computer programs are designed by making them out of objects that interact with one another. OOP languages are diverse, but the most popular ones are class-based, meaning that objects are instances of classes, which also determine their types. Many of the most widely used programming languages (such as C++, Java, Python, etc.)





## JAVA PROGRAMMING LANGUAGE



**WRITE**

Java programming language was originally developed by Sun Microsystems which was initiated by James Gosling and released in 1995 as core component of Sun Microsystems' Java platform. With the advancement of Java and its widespread popularity, multiple configurations were built to suite various types of platforms. Ex: J2EE for Enterprise Applications, J2ME for Mobile Applications. Sun Microsystems has renamed the new J2 versions as Java Standard Edition(SE), Java Enterprise Edition(EE) and Java Micro Edition(ME) respectively. Java is guaranteed to be **Write Once, Run Anywhere.**



## JAVA PROGRAMMING LANGUAGE



### Characteristics of Java

- **Object Oriented:** In Java, everything is an Object. Java can be easily extended since it is based on the Object model.
- **Platform independent:** Unlike many other programming languages including C and C++, when Java is compiled, it is not compiled into platform specific machine, rather into platform independent byte code. This byte code is distributed over the web and interpreted by virtual Machine (JVM) on whichever platform it is being run.
- **Simple:** Java is designed to be easy to learn. If you understand the basic concept of OOP Java would be easy to master.



WRITE



## JAVA PROGRAMMING LANGUAGE



WRITE

- **Secure:** With Java's secure feature it enables to develop virus-free, tamper-free systems. Authentication techniques are based on public-key encryption.
- **Architectural-neutral** :Java compiler generates an architecture-neutral object file format which makes the compiled code to be executable on many processors, with the presence of Java runtime system.
- **Portable:** Being architectural-neutral and having no implementation dependent aspects of the specification makes Java portable. Compiler in Java is written in ANSI C with a clean portability boundary which is a POSIX subset.



## JAVA PROGRAMMING LANGUAGE



WRITE

- **Robust:** Java makes an effort to eliminate error prone situations by emphasizing mainly on compile time error checking and runtime checking.
- **Multithreaded:** With Java's multithreaded feature it is possible to write programs that can do many tasks simultaneously. This design feature allows developers to construct smoothly running interactive applications.
- **Interpreted:** Java byte code is translated on the fly to native machine instructions and is not stored anywhere. The development process is more rapid and analytical since the linking is an incremental and light weight process.



## JAVA PROGRAMMING LANGUAGE



WRITE

- **High Performance:** With the use of Just-In-Time compilers, Java enables high performance.
- **Distributed:** Java is designed for the distributed environment of the internet.
- **Dynamic:** Java is considered to be more dynamic than C or C++ since it is designed to adapt to an evolving environment. Java programs can carry extensive amount of run-time information that can be used to verify and resolve accesses to objects on run-time.





## JAVA PROGRAMMING LANGUAGE



WRITE

- **Byte code:** A byte code is machine instruction that the Java compiler generates and Java interpreter executes. When the compiler compiles a .java file, it produces a series of byte codes and stores them in a .class file. The Java interpreter (JVM) can execute the byte codes stored in the .class file.
- **JVM:** Java Virtual Machine (JVM) is a program which behaves as interpreter and translates the byte code into machine language as they go called just in time compilation.



# JAVA PROGRAMMING LANGUAGE



WRITE

- **Source Code:** The core program or text which is written in a language like C, C++ or Java is called source code.
- **Object Code:** The program which only is understood by the computer in the form of machine instructions or binary instructions called object code. In Java JVM is used to generate object code in the form of byte code.
- **RAD:** Rapid Application Development is software programming technique that allows quick development of software application.



# INTRODUCTION TO OBJECT ORIENTED PROGRAMMING



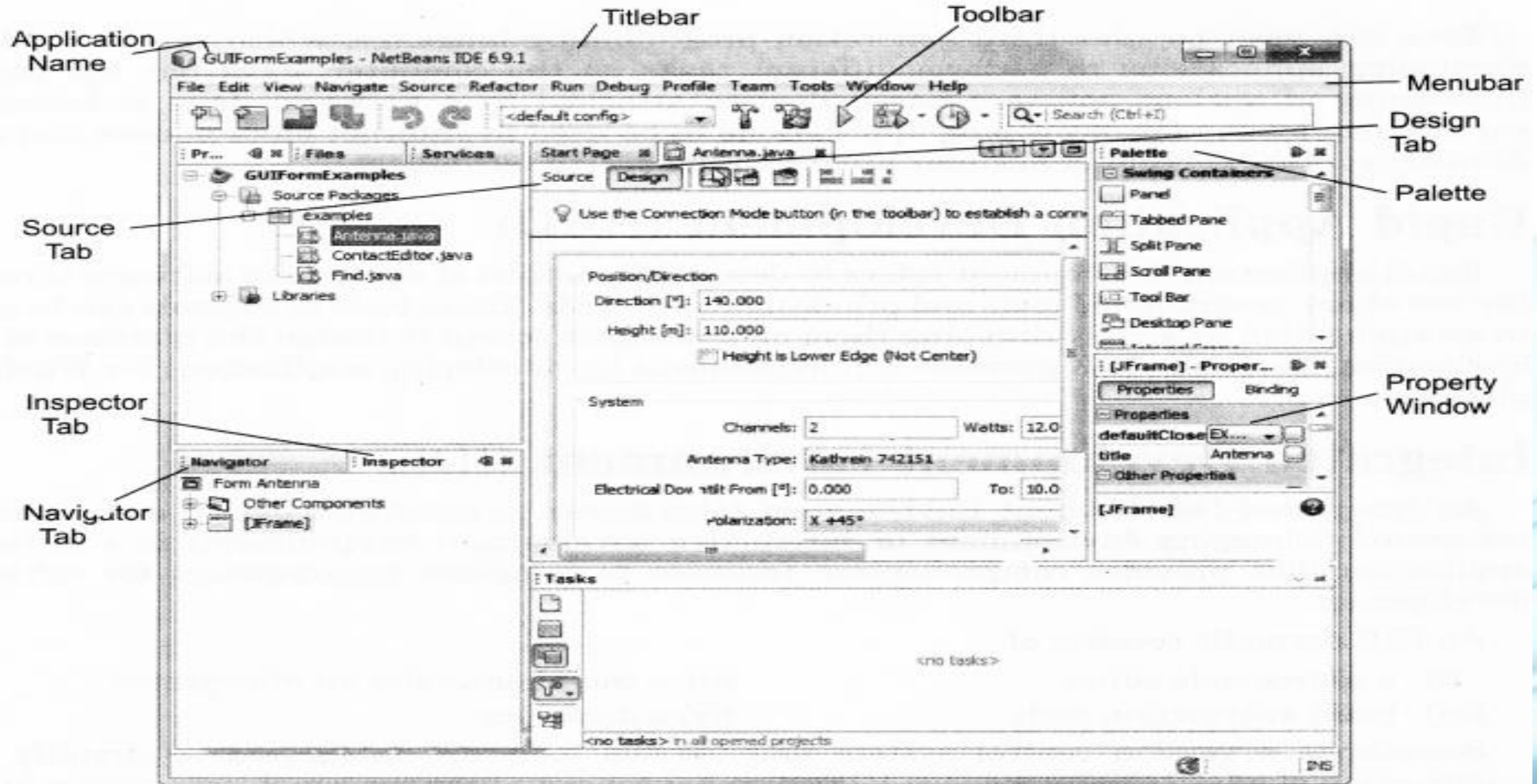
## About NetBeans IDE

WRITE

NetBeans IDE is a free, open source, integrated development environment (IDE) that enables you to develop desktop, mobile and web applications. The IDE supports application development in various languages, including Java, HTML5, PHP and C++. The IDE provides integrated support for the complete development cycle, from project creation through debugging, profiling and deployment. The IDE runs on Windows, Linux, Mac OS X, and other UNIX-based systems. NetBeans offers features such as: drag and drop GUI Creation, excellent editing, web services, excellent debugging, wizards, code generation and management tools. Java SDK along with NetBeans can be installed from

<http://java.sun.com/javase/downloads/netbeans.html>

# NETBEANS WINDOW



NetBeans IDE 6.9.1 Window



## NETBEANS WINDOW



**Titlebar** The titlebar is on the top of the window, containing the name of application.

WRITE

**Menubar** Below the title bar, there is a menu bar. It contains many options, i.e. pull down menu. These options can be used to perform various tasks.

**Toolbar** Tool bar contains many small icons for performing various tasks.

**GUI Builder** It is also known as Design Area or Design Space. It is an area to place components and construct GUI application visually. There are two types of views of the GUI builder, the Design View and the Source View



## NETBEANS WINDOW



WRITE

In Design View, you can see the user interface of your application while in Source View, you can add/edit the code for your application. We can switch over from one view to another by simply clicking on the source and design tabs directly above the Design Area.

**Palette** Palette contains controls or components used to create GUI applications.

**Inspector Window** This window is used to display a hierarchy of all the components or controls placed on the current form.



## NETBEANS WINDOW

WRITE

**Properties Window** Using this window, we can make changes in the properties of currently selected control on the form.

**Code Editor Window** It is the area where we write code for our Java applications. This window is displayed when we click on the source tab.

# NETBEANS WINDOW



## Components

Components are also known as Widgets (Windows gadgets). These are the basic interface elements. User interacts with JLabel, JButton, JTextField, etc. Components are placed on a container (like the JFrame).

There are two types of controls:

**Parent or Container Controls** They act as a background for other controls, e.g. Frame. When we delete a parent control, all its child controls also get deleted. When we move a parent control, all its child controls also move along with it.

**Child Controls Controls** placed inside a container control are called child controls, e.g. TextField, Label, Button, etc.

WRITE



# GRAPHICAL USER INTERFACE(GUI)



## BASICS OF A GUI

WRITE

GUI stands for **Graphical User Interface**. It refers to the windows, buttons, dialogs, menus and everything visual in modern application.

### Working of windows, Events and messages.

A window can be thought of as a simply rectangular region with its own boundaries. Examples are a document window or a dialog box etc. There are many other type of windows, a command button is also a window. Icons, text boxes, option buttons and menu bars are all windows.



# GRAPHICAL USER INTERFACE(GUI)



**WRITE**

The windows operating system manages all type of windows by assigning each one a unique id number (called the windows handle) The system continually monitors each of these windows for sign of activity or events.

**An event refers to the occurrence of an activity**

Each time an event occurs, it causes a message to be sent to the operating system.

**A message is the information/Request sent to the application.**

# GRAPHICAL USER INTERFACE(GUI) IN JAVA



**WRITE**

In Java, GUI features are supported via JFC (Java Foundation Classes) which encompass a group of features for building GUI's and adding rich graphics functionality and interactivity to Java applications.

One major feature of JFC is Swing API, which includes everything from buttons to tables and all functionality to create a GUI application. The components that you can use for creating GUI in JAVA are available through Swing API.

**A component is an object that defines a screen element such as a push button, text field, scroll bar, menu etc.**

# GRAPHICAL USER INTERFACE(GUI) IN JAVA



## Types of graphical components

WRITE

A control is also known as a widget (windows gadget)

The graphical controls that you put in GUI application are broadly of two types:

- 1. Container control:** It is a control that holds other controls within it. Eg: Panel or Panes or Frames
- 2. Child Control:** Controls inside a container are known as child controls. Child controls can exist completely inside their containers. Eg: Label, Button, Text field etc.

# GRAPHICAL CONTROLS OF SWING



WRITE

**1. JFrame** The JFrame is used to display a separate window with its own title bar.

**2. JLabel** The JLabel is used to display uneditable text. It means the user cannot change the information.

**3. JTextField** The JTextField is used to display a text box. In this text box, the user can input or edit the data.

**4. JPasswordField** does not directly display its content. Infact it uses a single character (usually an ASTERISK) to represent each character that it contains, so that it is possible to see how many characters have been typed, but not what they are. As its name suggests, JPasswordField is intended to be used as a simple way to accept a users password. JPasswordField is derived from JTextField.

# GRAPHICAL CONTROLS OF SWING



**5. JButton** The JButton displays a push button. The push button is used to generate the associated action event.

**6. JCheckBox** The JCheckBox is used to display a check box. The check box is used to allow the user to select multiple choices out of given choices. When the user selects a particular choice, a '√' is shown in front of it.

**7. JList** The JList displays a list for the application. We can select multiple elements from this list.

**8. JComboBox** The JComboBox provides a drop-down list of items. We can select an item from this list. Also, we can add a new item in the list. Infact, the combo box is a combination of a list and a text field.

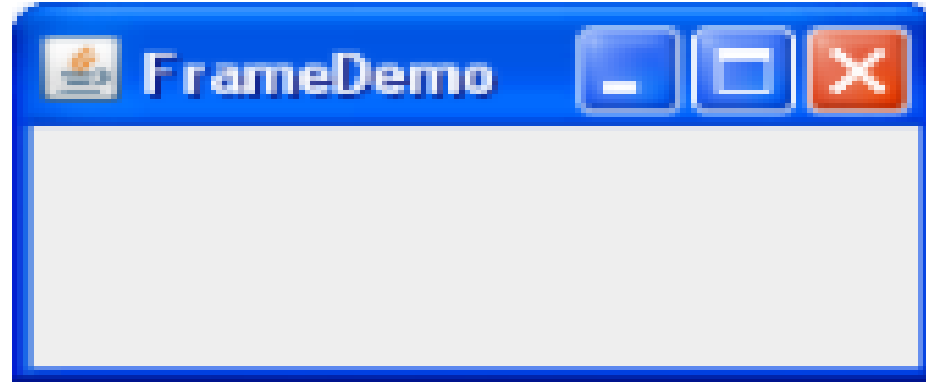
# GRAPHICAL CONTROLS OF SWING



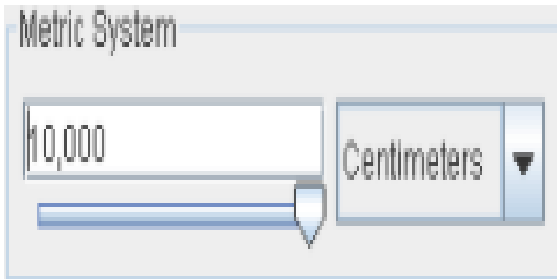
**WRITE**

**8. JPanel** The JPanel is used to organise the components in a GUI application. It is a supporting container, which cannot be displayed but can be added to another container.

**9. JRadioButton** The JRadioButton provides the radio buttons for the application. During the execution of a program, we can set these radio buttons either ON or OFF.



JFrame



JPanel

Host	User	Password	Last Modified
Biocca Games	Freddy	!#asf6Awwzb	Mar 16, 2006
zabble	ichabod	Tazbl34\$tZ	Mar 6, 2006
Sun Developer	fraz@hotmail.co...	AasW541!fbZ	Feb 22, 2006
Heirloom Seeds	shams@gmail...	blz[ADF78!	Jul 29, 2005
Pacific Zoo Shop	seal@hotmail.c...	vbAf124%z	Feb 22, 2006

JTable

*This is an editable JTextArea. A text area is a "plain" text component, which means that although it can display text in any font, all of the text is in the same font.*

JTextArea



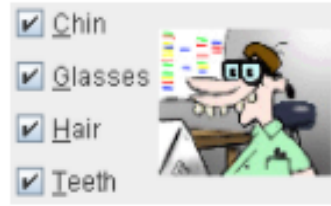


## Basic Controls

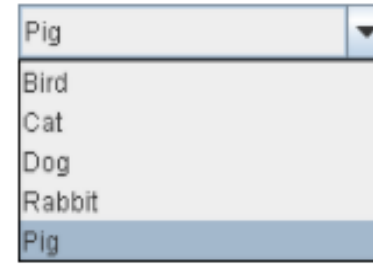
Simple components that are used primarily to get input from the user; they may also show simple state.



[JButton](#)



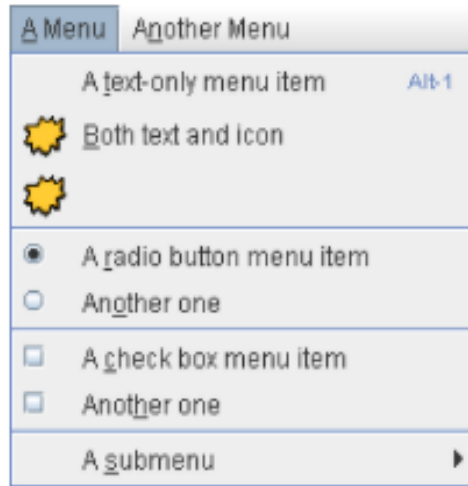
[JCheckBox](#)



[JComboBox](#)



[JList](#)



[JMenu](#)



[JRadioButton](#)



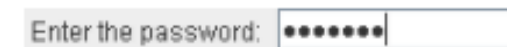
[JSlider](#)



[JSpinner](#)



[JTextField](#)



[JPasswordField](#)



Swing Controls Methods and Properties: These are the Swing Controls available with Netbean IDE and their concern methods and properties are given below.



Swing Controls	Methods	Properties
jButton	<ul style="list-style-type: none"><li>• getText()</li><li>• setText()</li></ul>	<ul style="list-style-type: none"><li>• Background</li><li>• Enabled</li><li>• Font</li><li>• Foreground</li><li>• Text</li><li>• Label</li></ul>
jLabel	<ul style="list-style-type: none"><li>• getText()</li></ul>	<ul style="list-style-type: none"><li>• Background</li><li>• Enabled</li><li>• Font</li><li>• Foreground</li><li>• Text</li></ul>
jTextField	<ul style="list-style-type: none"><li>• getText()</li><li>• isEditable()</li><li>• isEnabled()</li><li>• setText()</li></ul>	<ul style="list-style-type: none"><li>• Background</li><li>• Editable</li><li>• Enabled</li><li>• Font</li><li>• Foreground</li><li>• Text</li></ul>



jRadioButton	<ul style="list-style-type: none"><li>• getText()</li><li>• setText()</li><li>• isSelected()</li><li>• setSelected()</li></ul>	<ul style="list-style-type: none"><li>• Background</li><li>• Button Group</li><li>• Enabled</li><li>• Foreground</li><li>• Label</li><li>• Selected</li></ul>
jCheckBox	<ul style="list-style-type: none"><li>• getText()</li><li>• setText()</li><li>• isSelected()</li><li>• setSelected()</li></ul>	<ul style="list-style-type: none"><li>• Button Group</li><li>• Font</li><li>• Foreground</li><li>• Label</li><li>• Selected</li><li>• Text</li></ul>
jButtonGroup		<ul style="list-style-type: none"><li>• Add</li></ul>
jComboBox	<ul style="list-style-type: none"><li>• getSelectedItem()</li><li>• getSelectedIndex()</li><li>• setModel()</li></ul>	<ul style="list-style-type: none"><li>• Background</li><li>• ButtonGroup</li><li>• Editable</li><li>• Enabled</li><li>• Font</li><li>• Foreground</li><li>• Model</li><li>• SelectedIndex</li><li>• SelectedItem</li><li>• Text</li></ul>





**WRITE**

jList	<ul style="list-style-type: none"><li>• <code>getSelectedValue()</code></li><li>• <code>getSelectedValues()</code></li><li>• <code>getSelectedIndex()</code></li><li>• <code>getSelectedIndices()</code></li><li>• <code>setModel()</code></li></ul>	<ul style="list-style-type: none"><li>• Background</li><li>• Enabled</li><li>• Font</li><li>• Foreground</li><li>• Model</li><li>• SelectedIndex</li><li>• SelectedItem</li><li>• SelectedMode</li><li>• Text</li></ul>
jTable	<ul style="list-style-type: none"><li>• <code>addRow()</code></li><li>• <code>addModel()</code></li></ul>	<ul style="list-style-type: none"><li>• model</li></ul>
JOptionPane	<ul style="list-style-type: none"><li>• <code>showMessageDialog()</code></li></ul>	<ul style="list-style-type: none"><li>• <code>getRowCount()</code></li><li>• <code>RemoveRow()</code></li><li>• <code>addRow()</code></li></ul>





**Focus:** The control under execution is said to have the focus. The control having the focus obtains input from the user.

**WRITE**

**getText():** getText() method is used to obtain the text from a jTextField during the run time.

**setText():** setText() method is used to set or change the text of a jTextField during run time.



**Parse Methods:** parse() methods helps to parse string into different numeric types. These are:

Method	Syntax	Usage
parseByte()	Byte.parseByte(string)	To convert a string value to byte type
parseShort()	Short.parseShort(string)	To convert a string value to type short
parseInt()	Integer.parseInt(string)	To convert a string value to Integer type
parseLong()	Long.parseLong()	To convert a string value to Long type
parseFloat()	Float.parseFloat()	To convert a string value to Float type
parseDouble()	Double.parseDouble()	To convert a string value to Double type



**WRITE**



## Java Character Set

Character set refers to a set of valid characters that the language can recognise. In Java, Unicode characters are used.



**WRITE**

### Token

The token refers to the smallest unit of a program. In Java language, many tokens are used. These are keywords, identifiers, literals, punctuators and operators.

**(i) Keywords** The keywords are the reserved words defined in the language compiler. These reserved words are used for special purpose and cannot be used for any other purpose such as an identifier name.



Commonly used keywords are as follows:



**WRITE** →

abstract	class	extends	import	private	switch
boolean	const	final	instanceof	protected	this
break	continue	finally	int	public	throw
byte	default	float	interface	return	try
case	do	for	long	short	void
catch	double	if	new	static	volatile
char	else	implements	package	super	while





**(ii) Identifiers** The identifiers are used for naming different items of the programs such as variables, characters, classes, functions, arrays, etc.



**WRITE** →

The identifiers are subject to following rules:

1. The identifier cannot be a reserved word.
2. The identifier cannot start with a digit.
3. It must start with an alphabet, dollar sign or an underscore.
4. It can contain alphabets, digits, underscore and dollar sign.
5. There is no limit on the length of the identifier.

**Valid identifiers:**

Myfile    \_sum    MYSALARY    Avg\_sal



**WRITE**

**Note** Java is a case sensitive language and we can use both uppercase and lowercase characters for naming an identifier. Here, a name written in uppercase is different from the same name written in lowercase (e.g. a and A are two different identifiers).

### **Invalid identifiers:**

- |                 |   |
|-----------------|---|
| <b>DATA-REC</b> | - It contains special character –(hyphen) |
| <b>29CLCT</b>   | - Starting with digit                     |
| <b>break</b>    | - reserved keyword                        |
| <b>My.file</b>  | - contains special character              |



**(iii) Literals** The literals or the constants are those items, whose values cannot be changed during the execution of the program. There are number of literals used in a Java program. These are integer literals, floating literals, boolean literals, character literals, string literals and null literals.

**WRITE** →

**(a) Integer Literals** The integer literals are the whole numbers without any fractional part. We can enter the integers as decimal numbers (base 10), octal numbers (base 8) and hexadecimal numbers (base 16).

The decimal number is entered as a number which begins with a non-zero digit (e.g. 35), for an octal number we have to enter the number that begins with a zero (e.g. 026) and a hexadecimal number must start with Ox or OX. (e.g. 0x2B).





**(b) Floating Literals** The floating literals or the real numbers are those numbers which have fractional part. The numbers can be written either in fractional form or in exponential form.

e.g. 35.45. To write  $5.3 \times 10^3$ , we have to write 5.3E03 or 5.3e03.

**(c) Boolean Literals** The boolean type literals can have a boolean type value i.e. either true or false.

**(d) Character Literals** The character literal is one character enclosed in a single quotes, e.g. 'a'.

**(e) String Literals** When multiple characters are entered such as a name of a person or a place. The string literals are enclosed by double quotes, e.g. "xyz".

**(f) Null Literals** The null type literal has only null value. It is formed by a literal null.

**WRITE**





sequences. An escape sequence is represented by characters. Following table (Table 4.1) gives a listing of escape sequences :

**Table 4.1** *Escape Sequences in Java*

<i>Escape Sequence</i>	<i>Nongraphic Character</i>
<code>\a</code>	Audible bell (alert)
<code>\b</code>	Backspace ✓
<code>\f</code>	Formfeed ✓
<code>\n</code>	Newline or linefeed ✓
<code>\r</code>	Carriage Return ✓✓
<code>\t</code>	Horizontal tab
<code>\v</code>	Vertical tab
<code>\\</code>	Backslash
<code>\'</code>	Single quote
<code>\"</code>	Double quote
<code>\?</code>	Question mark
<code>\0n</code>	Octal number (0n represents the number in octal)
<code>\xHn</code>	Hexadecimal number (Hn represents the number in hexadecimal)
<code>\uHn</code>	Unicode character represented through its hexadecimal code Hn.
<code>\0</code>	Null ✓

In the above table, you see sequences representing `\, ', "` and `?` also. Though these characters





**WRITE**

**(iv) Punctuators (or Separators)** A punctuator is a token that has syntactic and semantic meaning to the compiler, but the exact significance depends on the context. A punctuator can also be a token that is used in the syntax of the preprocessor.

**(v) Operators** There are several kinds of operators. Operators are used in expression to describe operations involving one or more operands. In the expression  $a + b$ , '+' is an operator involving two operands (a and b).



**Operators:** Operators are special symbols that perform specific operations on one, two, or three operands, and then return a result.

Operators	Precedence
Postfix	expr++ expr
Unary	++expr expr+expr expr~ !
Multiplicative	* / %
Additive	+ -
Shift	<< >> >>>
Relational	<> <= >= instanceof
Equality	== !=
Bitwise AND	&
Bitwise	exclusive OR ^
Bitwise Inclusive	OR
Logical	AND &&
Logical	OR
Ternary	? :
Assignment	= += *= /= %= &=  = <<= >>= >>>=

**WRITE** 



## DATA TYPES

Java like any other programming language provides ways and facilities to handle different types of data by providing data types.

**Data types are means to identify the type of data and associated operations of handling it.**

**Java data types are of two types.**

### 1. Primitive Data Types:

The Java programming language is statically typed, which means that all variables must first be declared before they can be used.

A primitive type is predefined by the language and is named by a reserved keyword. The eight primitive data types supported by the Java programming language are:

- **byte:** The byte data type is an 8 bit signed two's complement integer. It has a minimum value of -128 and a maximum value of 127 (inclusive).



WRITE





WRITE

## DATA TYPES



- **short:** The short data type is a 16 bit signed two's complement integer. It has a minimum value of 32,768 and a maximum value of 32,767 (inclusive).
- **int:** The int data type is a 32 bit signed two's complement integer. It has a minimum value of 2,147,483,648 and a maximum value of 2,147,483,647 (inclusive).
- **long:** The long data type is a 64 bit signed two's complement integer. It has a minimum value of 9,223,372,036,854,775,808 and a maximum value of 9,223,372,036,854,775,807 (inclusive).
- **float:** The float data type is a single precision 32 bit IEEE 754 floating point.
- **double:** The double data type is a double precision 64 bit IEEE 754 floating point.



WRITE

## DATA TYPES



■ **boolean:** The boolean data type has only two possible values: true and false. Use this data type for simple flags that track true/false conditions.

■ **char:** The char data type is a single 16 bit Unicode character. It has a minimum value of '\u0000' (or 0) and a maximum value of '\uffff' (or 65,535 inclusive).

2. **Reference Data Types:** These are constructed by using primitive data types, as per user need.

Reference data types store the memory address of an object.

Class, store the memory address of an object.

Class, Interface and Array are the example of Interface Reference Data types.



## Text Interaction Methods in Java

For text interaction in Java usually following types of methods are used:

**(i) getText() Method** get text from a GUI component.

The `getText( )` method is used to return the text stored in the text based GUI component such as a text box. e.g. if the variable name is `jTextField1`, then to obtain text from the `jTextField1` text field. We need to type following statement:

```
String Name = jTextField1.getText();
```

**(ii) setText() Method** set text into a GUI component.

A `setText( )` method is used to store or change text in a text field, e.g. to change a text in a text field named `stuName` we need to write: `jTextField1.setText("Kritika")`:



WRITE



## Non-GUI Output Methods

**WRITE**

For displaying an output on console window (or terminal window) in a Java program, we can use either of two methods i.e. `System.out.print()` or `System.out.println()`.

**1. `System.out.print( )`** -This method displays the text and keeps the cursor in the same line.

**e.g.** `System.out.print("Welcome to Java Programming");`

**2. `System.out.println( )`** -This method displays the text and then moves the cursor to the next line.

**e.g.** `System.out.println("Java Programming is fun");`

# VARIABLES



Variables represent named storage locations, whose value can be manipulated during program run. For instance, to store the name of a student and marks of a student during a program run, we require storage locations that too named so that these can be distinguished easily.



**WRITE**

**A Variable is a named memory location, which holds a data value of a particular data type.**

**Declaration of a variable:**

**Syntax :**

**type variable name ;**

**Eg: (declaring an integer with the variable name age)**

**int age ;**

# VARIABLES



## Initialization of variables

The first value given to a variable at the time of declaration itself is known as initialization of a variable.

Eg:

```
int val = 100 ;
```

```
String name = "Rohan" ;
```

```
float amount = 5000.75 ;
```

## Dynamic Initialization

The expression that initializes a variable can be an expression with

```
int a = 5 , b = 10 ;
```

```
double c = Math.sqrt(a*a + b*b) ; // here variable 'c' is  
initialized by the return value of the square root method of  
the built-in Math class
```

**WRITE**



## Constant

To make a variable as a constant, final keyword is used. Such value cannot be changed during the execution of the program.

e.g. `final float PI = 3.14;`

## Text Interaction Methods in Java

For text interaction in Java usually following types of methods are used:

WRITE



## Operators in Java

An operator on one or more variables and perform on the all tasks related to programming language. It consists of words or symbols. In Java language, many operators are used. We can categorise these operators as follows:

**1. Arithmetic Operators** In Java language, following five arithmetic operators are used:

Operator	Use	Description
+ (plus)	$a + b$	Addition operator
- (minus)	$a - b$	Subtraction operator
* (asterisk)	$a * b$	Multiplication operator
/ (slash)	$a / b$	Division operator
% (percent)	$a \% b$	Modulus operator, which finds the remainder of a division

**WRITE**





**WRITE**

**2. Increment/Decrement Operators** There are two forms of increment operator (`++`) and decrement operator (`--`) as follows:

**(i) Prefix Form** In prefix form, the operator `++` or `--` is used before the operand. In such form, first there will be increment or decrement in the value of the variable and then it is used. This form follows the change-then-use rule:

**e.g.**

```
int a = 5, b;
```

```
b = ++a;
```

Here, the value of `b` will be 6, also the last value of `a` will be 6.



**(ii) Postfix Form** In postfix form, the operator `++` or `--` is used after the operand. In the postfix form, first the value of the variable is used, then there will be increment or decrement in the value of the variable. This form follows the use-then-change rule:

e.g.

```
int a = 5, b;
```

```
b = a++;
```

Here, the value of `b` will be 5 and the last value of `a` will be 6.

**WRITE** 



**3. Logical Operators** The logical operators are used to make a decision on two conditions. Logical operators are mainly used to control program flow. There are three logical operators used in Java.

These are as follows:

- **AND Operator (&&)** This operator combines two given conditions and evaluates the result to true, if both of the given conditions are true.
- **OR Operator (||)** This operator combines two given conditions and evaluates the result to true, if either of the given conditions or both of the conditions are true.
- **NOT Operator (!)** The NOT operator is an unary operator. It negates the given condition.

**WRITE**



## AND ( && ) OPERATOR EXAMPLE



WRITE

$(6 == 3) \ \&\& \ (4 == 4)$  results into false because first expression is false.  
 $(4 == 4) \ \&\& \ (8 == 8)$  results into true because both expressions are true.  
 $6 < 9 \ \&\& \ 4 > 2$  results into true because both expressions are true.  
 $6 > 9 \ \&\& \ 5 < 2$  results into false because both expressions are false.

Because logical AND operator (&&) has lower precedence than the relational operators, we don't need to use parentheses in these expressions.



**WRITE** →

## OR ( || ) OPERATOR EXAMPLE

$(4==4) || (5 ==8)$

results into true because first expression is true.

$1 == 0 || 0 >1$

results into false because neither expression is true (both are false).

$5 > 8 || 5 < 2$

results into false because both expressions are false.

$1 < 0 || 8 > 0$

results into true because second expression is true.

The operator || (logical OR) has lower precedence than the relational operators, thus, we don't need to use parenthesis in these expressions.





**WRITE**

**5. Relational Operators** The relational operators are used to compare two operands and determines the relationship between them. Most of relational operators are used in "if" statement and inside "loop" statement in order to check truthness or falseness of condition. The relational operators are binary operators and returns boolean value. The relational operators and their uses are given below:

Operator	Use	Description
<code>==</code>	<code>a == b</code>	a is equal to b
<code>&gt;</code>	<code>a &gt; b</code>	a is greater than b
<code>&gt;=</code>	<code>a &gt;= b</code>	a is greater than or equal to b
<code>&lt;</code>	<code>a &lt; b</code>	a is less than b
<code>&lt;=</code>	<code>a &lt;= b</code>	a is less than or equal to b
<code>!=</code>	<code>a != b</code>	a is not equal to b



Table 4.7 Relational Operators

$p$	$q$	$p < q$	$p \leq q$	$p == q$	$p > q$	$p \geq q$	$p != q$
0	1	t	t	f	f	f	t
1	0	f	f	f	t	t	t
3	3	f	t	t	f	t	f
2	6	t	t	f	f	f	t

t represents *true* and f represents *false*.

WRITE



**WRITE**

## Shorthand Assignment Operators

In Java language, five shorthand assignment operators are used, these are +=, -=, \*=, /= and %=.

e.g.

`x = x + 5` can be written as `x += 5;`

`x = x - 5` can be written as `x -= 5;`

`x = x * 5` can be written as `x *= 5;`

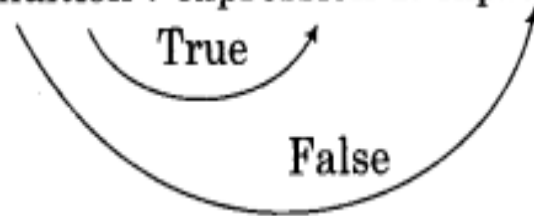
`x = x / 5` can be written as `x /= 5;`

`x = x % 5` can be written as `x %= 5;`

7. Conditional Operator It is used when one condition and two expressions are given.

### Syntax

`Condition ? expression 1: expression 2;`







When the value of condition is true then first expression will be returned. If the condition is false, then second expression will be returned whichever value is returned is depend on the condition.

**WRITE**

Operators can also be divided into three parts, on the basis of number of operands:

1. **Unary Operator** An unary operator requires only one operand. Example of unary operators are unary +, unary -, ++ and --. These operator performs task only on a single operand.
2. **Binary Operator** A binary operator requires two operands. Examples of binary operators are + and %.
3. **Ternary Operator** A ternary operator requires three operands. Example of ternary operator is ?: (the conditional operator).

supports.

**Table 4.10** Other Operators

Operator	Description
? :	Shortcut if-else statement
[ ]	Used to declare arrays, create arrays, and access array elements
.	Used to form qualified names
(params)	Delimits a comma-separated list of parameters
(type)	Casts (converts) a value to the specified type
new	Creates a new object or a new array
instanceof	Determines whether its first operand is an instance of its second operand
shift operator (>>, <<, >>>)	Perform bit manipulation on data by shifting bits
bitwise operators (  , &, ^)	Perform bitwise comparisons
Complement operator (~)	Interts each bit of operand.

WR

In the following lines, we are discussing some of these operators. Discussion of all these

**Q. Design an application to accept two integers in the text boxes and find its sum and display the result in a text box as shown below. The result is displayed when you click the “Calculate Sum” Button.**

The screenshot displays the NetBeans IDE 6.5.1 interface. The main workspace is in Design mode, showing a graphical user interface for a Java Swing application. The UI consists of three text input fields and one button. The first two fields are labeled "ENTER FIRST NUMBER" and "ENTER SECOND NUMBER". The third field is labeled "SUM". Below these fields is a button labeled "CALCULATE SUM".

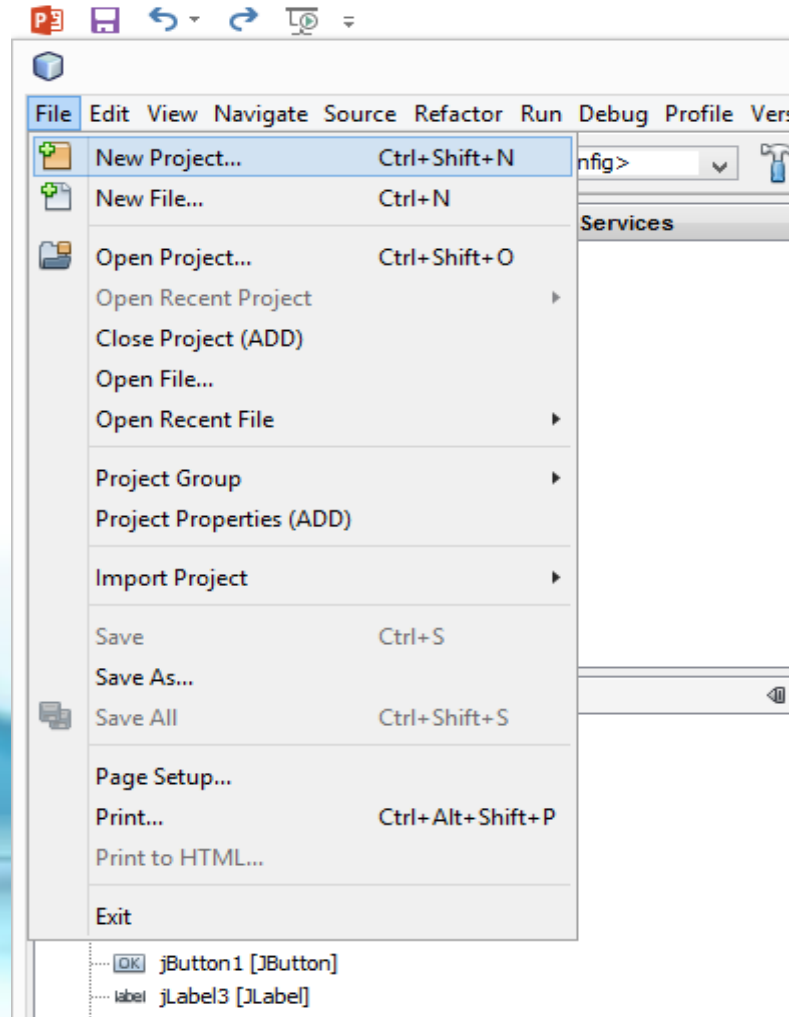
The interface includes a menu bar (File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Versioning, Tools, Window, Help), a toolbar, and several panels:

- Projects:** Shows a project named "ADD" with source packages, test packages, and libraries.
- Navigator:** Shows the "Form NewJFrame" component.
- Inspector:** Shows the selected component, "[JFrame]".
- Palette:** Lists Swing Containers (Panel, Split Pane, Tool Bar, Internal Frame) and Swing Controls (Label, Toggle Button, Radio Button, Combo Box, Text Field, Scroll Bar, Tabbed Pane, Scroll Pane, Desktop Pane, Layered Pane, OK Button, Check Box, Button Group, List, Text Area, Slider).
- [JFrame] - Properties:** Shows the "defaultCloseOperation" property set to "EXIT\_ON\_CLOSE".
- Output:** An empty area for displaying program output.
- Tasks:** An empty area for displaying task-related information.



**Q. Design an application to accept two integers in the text boxes and find its sum and display the result in a text box as shown below. The result is displayed when you click the “Calculate Sum” Button.**

**Step1: Open the NetBeans, click File - > new project**



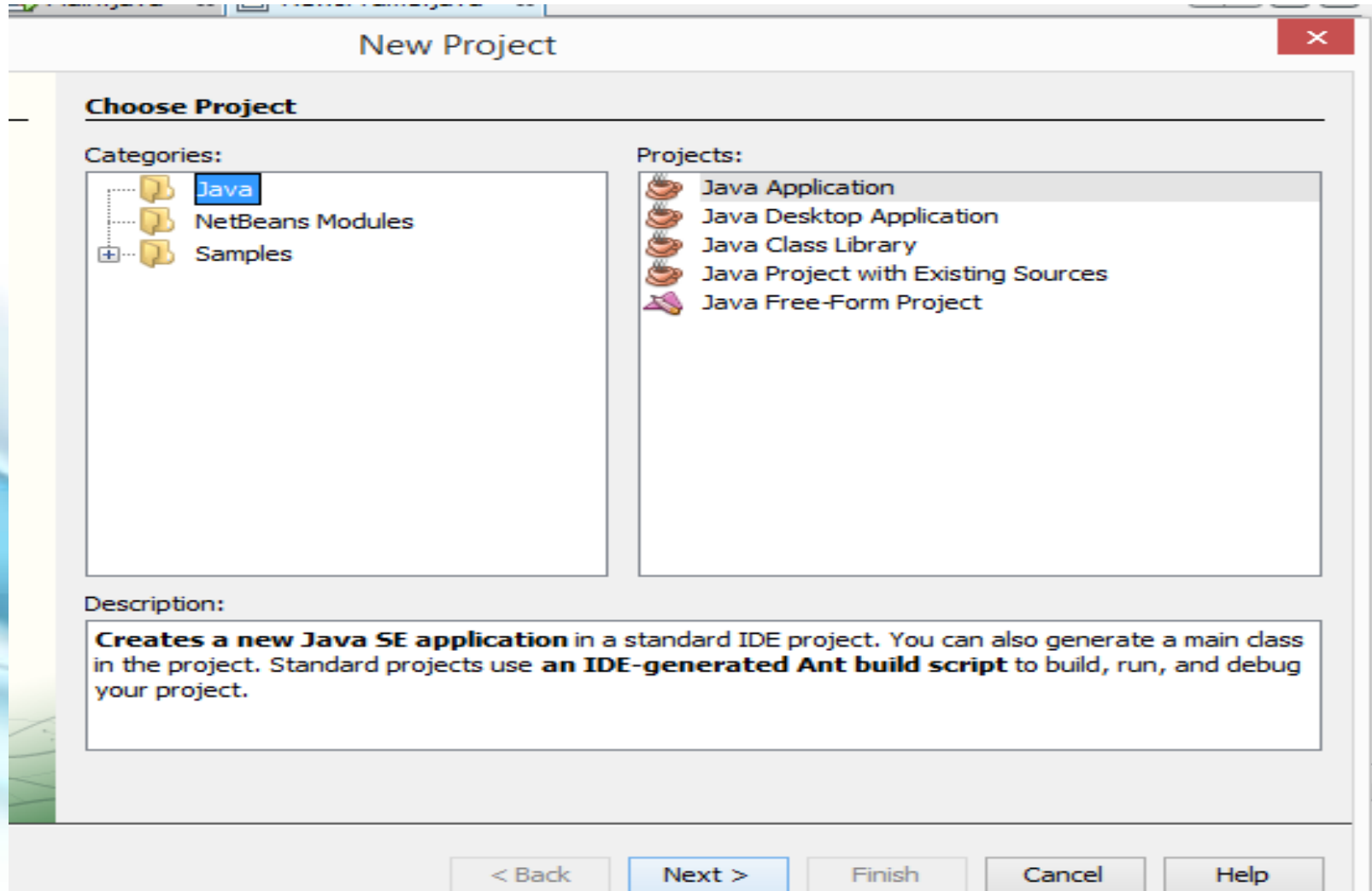


**Q. Design an application to accept two integers in the text boxes and find its sum and display the result test box as shown below. The result is displayed when you click the “Calculate Sum” Button.**

**Step 2: click Java on categories and select Java Application on Projects and then click next**



**WRITE**



**Step 3: Give the project name as ADDNUMBERS . Disable the tick mark on “Create Main Class” at the bottom of the page. Click the Finish button**



bility Mode] - PowerPoint DRAWING TOOLS

ADD - NetBeans IDE 6.5.1

Help

Main.java NewJFrame.java

### New Java Application

**Name and Location**

Project Name:

Project Location:  Browse...

Project Folder:

Use Dedicated Folder for Storing Libraries

Libraries Folder:  Browse...

Different users and projects can share the same compilation libraries (see Help for details).

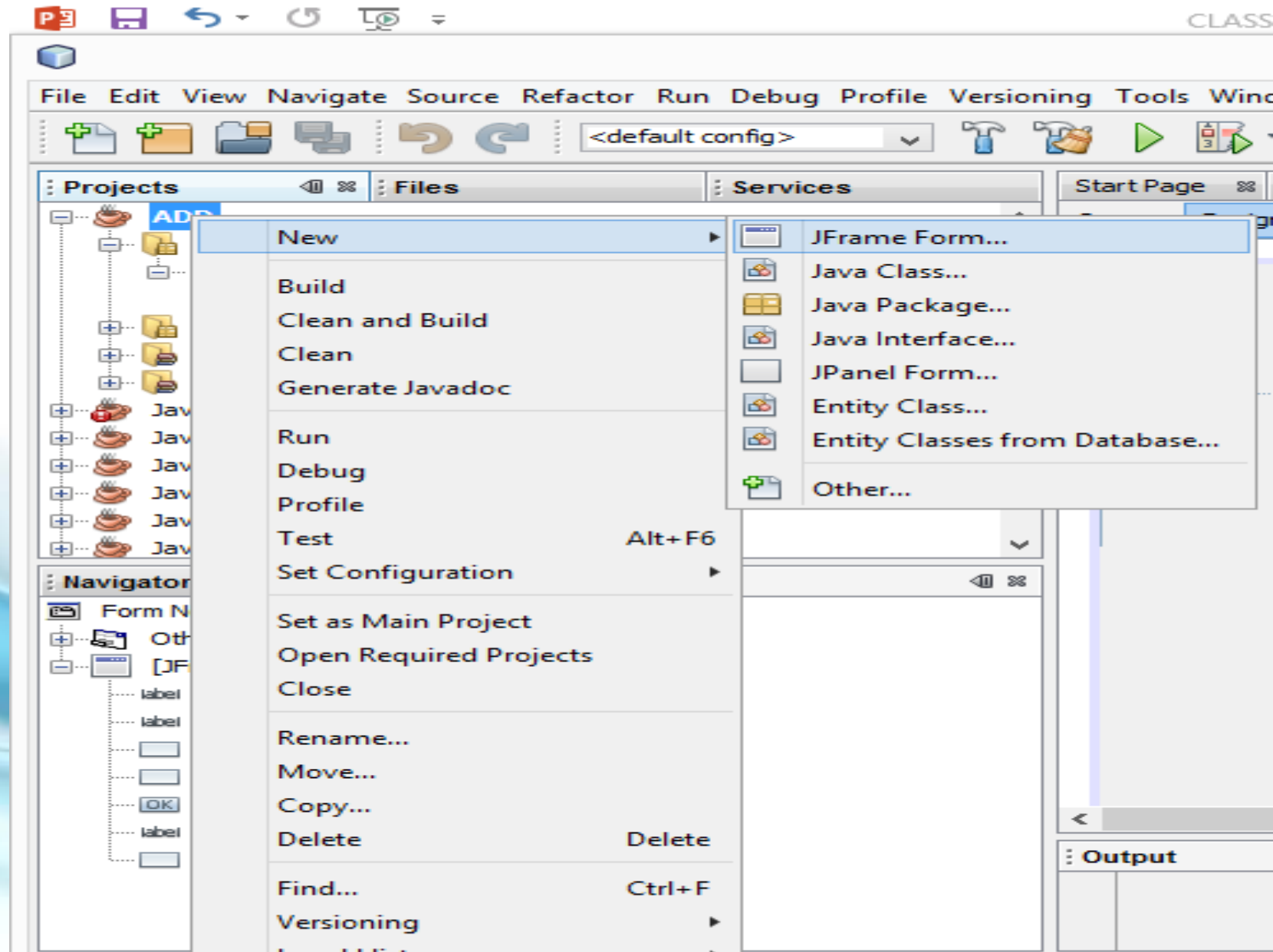
Create Main Class

Set as Main Project

< Back Next > Finish Cancel Help



**Step 4: Select ADDNUMBERS available in the Projects window , Right click and select JFrame Form**



## Step 5: You will see now the design window as shown below

The screenshot displays the NetBeans IDE 6.5.1 interface. The title bar reads "ADD - NetBeans IDE 6.5.1". The menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Versioning, Tools, Window, and Help. The toolbar contains icons for file operations, configuration, and execution. The main workspace is divided into several panels:

- Projects:** Shows a project named "ADD" with source packages containing "NewJFrame.java" and "NewJFrame1.java".
- Navigator:** Displays the "Form NewJFrame1" structure, including "Other Components" and a "[JFrame]" component.
- Design Window:** The central area is in "Design" mode, showing a large, empty rectangular frame representing the visual structure of the JFrame.
- Palette:** Lists various Swing Containers (Panel, Split Pane, Tool Bar, Internal Frame, Layered Pane) and Swing Controls (Label, Button, Toggle Button, Check Box, Radio Button, Button Group, Combo Box, List, Text Field, Text Area, Scroll Bar, Slider).
- [JFrame] - Properties:** Shows the "Properties" tab for the selected JFrame component, with the "defaultCloseOperation" property set to "EXIT\_ON\_CLOSE".
- Output and Tasks:** Empty panels at the bottom of the IDE.



# Step 6: Drag 3 labels and 3 textboxes and an OK button control from the swing controls to the form as shown below

The screenshot displays the NetBeans IDE 6.5.1 interface. The main window, titled "ADD - NetBeans IDE 6.5.1", is in Design mode. The central canvas shows a form layout with three rows of controls: a label (jLabel1) and a text field (jTextField1) in the first row, a label (jLabel2) and a text field (jTextField2) in the second row, and a label (jLabel3) and a text field (jTextField3) in the third row. A fourth row contains a single button (jButton1) centered below the text fields. The left sidebar contains the "Projects" view showing a project named "ADD" with source packages and files, and the "Navigator" view showing the "Form NewJFrame1" with a list of components: JLabel1, JLabel2, JLabel3, JTextField1, JTextField2, JTextField3, and JButton1. The right sidebar contains the "Palette" view with "Swing Containers" and "Swing Controls" sections. The "Swing Controls" section includes Label, Toggle Button, Radio Button, Combo Box, Text Field, Scroll Bar, Button, Check Box, Button Group, List, Text Area, and Slider. The "Properties" view for the selected JButton1 shows the "defaultCloseOperation" set to "EXIT\_ON\_CLOSE". The bottom status bar shows "Output" and "Tasks" tabs.



ADD - NetBeans IDE 6.5.1

File Edit View Window Help

Start Page Main.java NewJFrame.java

Source Design

ENTER FIRST NUMBER

ENTER SECOND NUMBER

SUM

CALCULATE SUM

Palette

- Swing Containers
  - Panel
  - Split Pane
  - Tool Bar
  - Internal Frame
  - Tabbed Pane
  - Scroll Pane
  - Desktop Pane
  - Layered Pane
- Swing Controls
  - Label
  - Toggle Button
  - Radio Button
  - Combo Box
  - Text Field
  - Scroll Bar
  - Button
  - Check Box
  - Button Group
  - List
  - Text Area
  - Slider

[JFrame] - Properties

Properties Binding Events Code

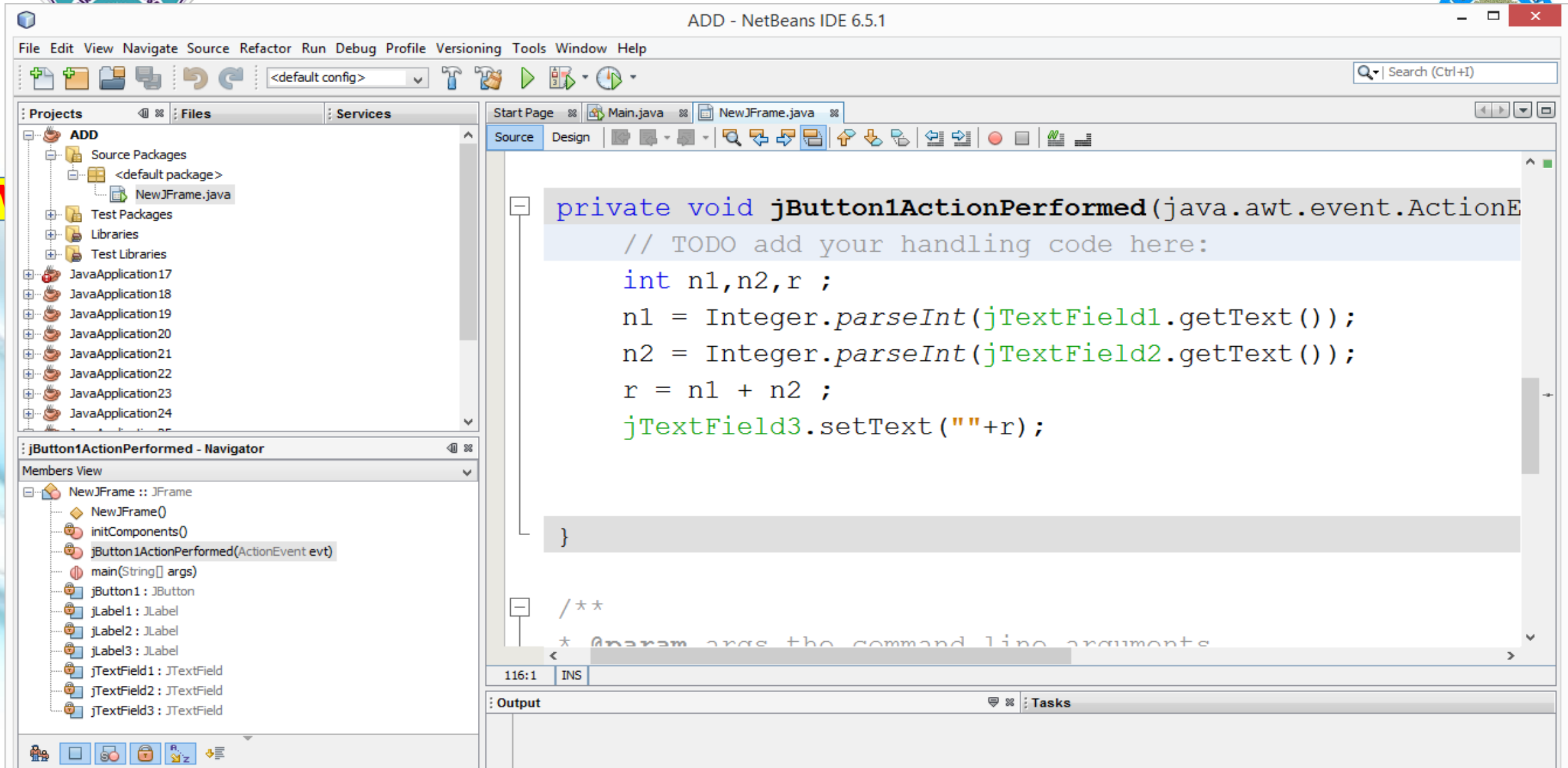
defaultCloseOperation EXIT\_ON\_CLOSE

[JFrame]

Output Tasks



## Step 7: Type the following code under jButtonActionPerformed



The screenshot displays the NetBeans IDE 6.5.1 interface. The main editor window shows the source code for `NewJFrame.java`. The code implements the `jButtonActionPerformed` method, which takes an `ActionEvent` parameter and performs the following actions:

```
private void jButtonActionPerformed(java.awt.event.ActionEvent  
    // TODO add your handling code here:  
    int n1,n2,r ;  
    n1 = Integer.parseInt(jTextField1.getText());  
    n2 = Integer.parseInt(jTextField2.getText());  
    r = n1 + n2 ;  
    jTextField3.setText(""+r);  
}
```

The IDE also shows the `Projects` and `Services` panes on the left. The `Projects` pane shows a project named `ADD` with a `NewJFrame.java` file. The `Services` pane shows the `Members View` for `NewJFrame :: JFrame`, listing various methods and fields, including `jButtonActionPerformed(ActionEvent evt)`.

## Step 8: Go to the Run menu and select Run File

The screenshot shows the NetBeans IDE 6.5.1 interface. The 'Run' menu is open, and the 'Run File' option (Shift+F6) is selected. The code editor displays the following Java code:

```
public void jButton1ActionPerformed(java.awt.event.ActionEvent  
// TODO add your handling code here:  
int n1, n2, r ;  
n1 = Integer.parseInt(jTextField1.getText());  
n2 = Integer.parseInt(jTextField2.getText());  
r = n1 + n2 ;  
jTextField3.setText(""+r);  
}
```

The 'jButton1ActionPerformed - Navigator' window shows the class structure for 'NewJFrame :: JFrame', with the 'jButton1ActionPerformed(ActionEvent evt)' method highlighted. The 'Output' window is empty.

## Step 9: Enter 2 integers in the first and second text boxes and click the button Calculate Sum

**WRITE**

ADD - NetBeans IDE 6.5.1

File Edit View Navigate Source Refactor Run Debug Profile Versioning Tools Window Help

Projects Files Services Start Page Main.java NewJFrame.java

ADD

- Source Pack
- <default>
- New
- Test Packag
- Libraries
- Test Librarie
- JavaApplication:
- JavaApplication:
- JavaApplication:
- JavaApplication:
- JavaApplication:
- JavaApplication:
- JavaApplication:
- JavaApplication:

Members View

- NewJFrame :: JF
- NewJFrame(
- initCompon
- jButton1Acti
- main(String[
- jButton1 : JF
- jLabel1 : JLab
- jLabel2 : JLab
- jLabel3 : JLab

ENTER FIRST NUMBER 10

ENTER SECOND NUMBER 20

SUM 30

CALCULATE SUM

Create the following GUI form Accept the title, First name, Last name Class and Section and display the information as shown below in a TextArea after clicking the Generate button at run time.



W

Screenshot Labels

The screenshot shows a Java Swing window titled "Data Form". It contains the following components:

- Title (Mr/Ms):** A text field containing "Ms."
- First Name:** A text field containing "Rabia"
- Last Name:** A text field containing "Khan"
- Class:** A text field containing "XI"
- Section:** A text field containing "B"
- Generate:** A button located below the input fields.
- Student Details:** A text area at the bottom of the window displaying the output: "Name: Ms. Rabia Khan" and "Class: XI B".

Text Fields

Button

Text Area



## Generate Button coding

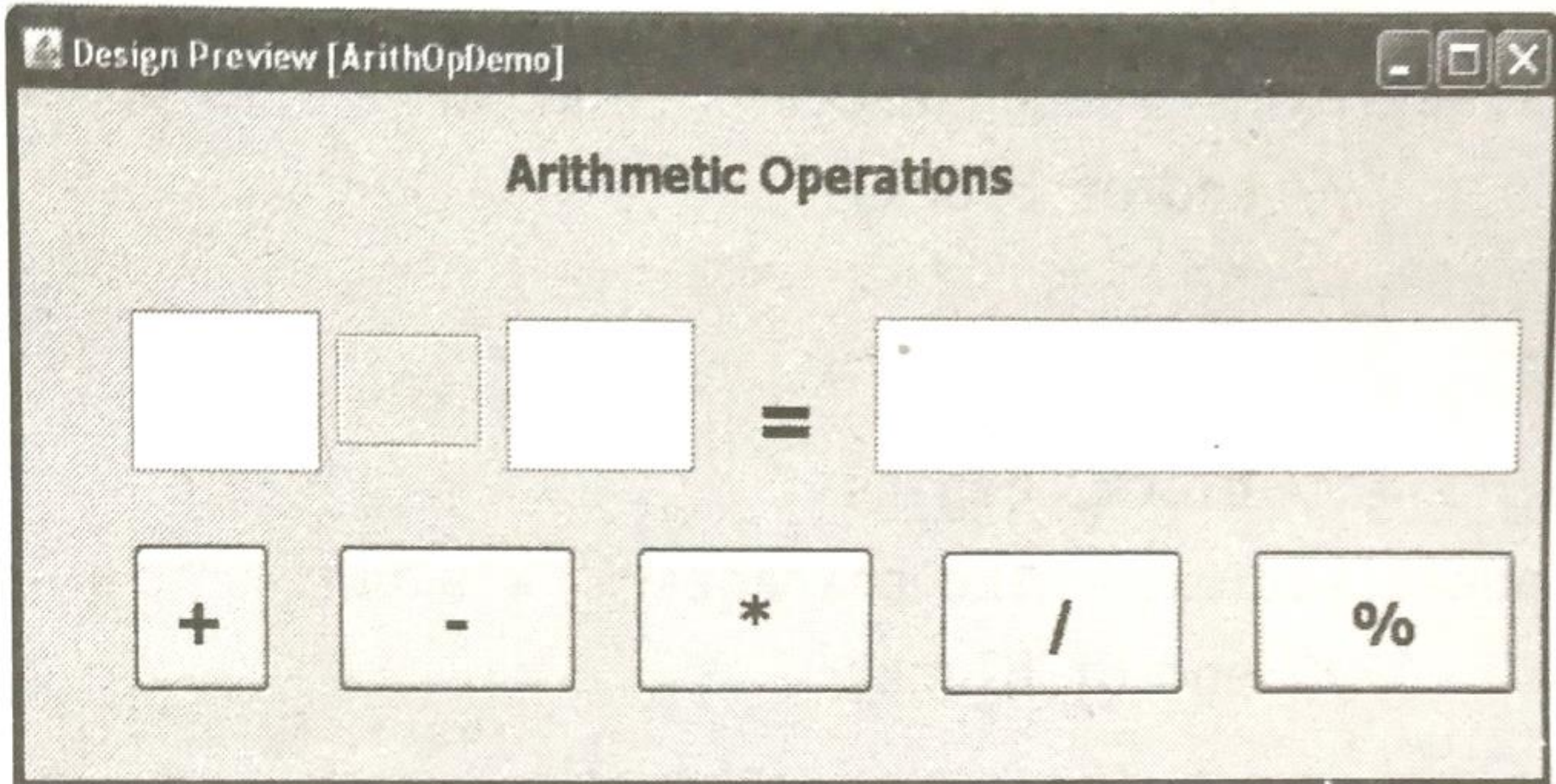


**WRITE** →

```
String text1 = jTextField1.getText( );  
String text2 = jTextField2.getText( );  
String text3 = jTextField3.getText( );  
String text4 = jTextField4.getText( );  
String text5 = jTextField5.getText( );  
jTextArea1.append("Student Details" + "\n");  
jTextArea1.append("Name: " +text1+ " " +text2+ " " +text3+"\n");  
jTextArea1.append("Class: " +text4+ " " +text5) ;
```



Design an application that the shows all arithmetic operations



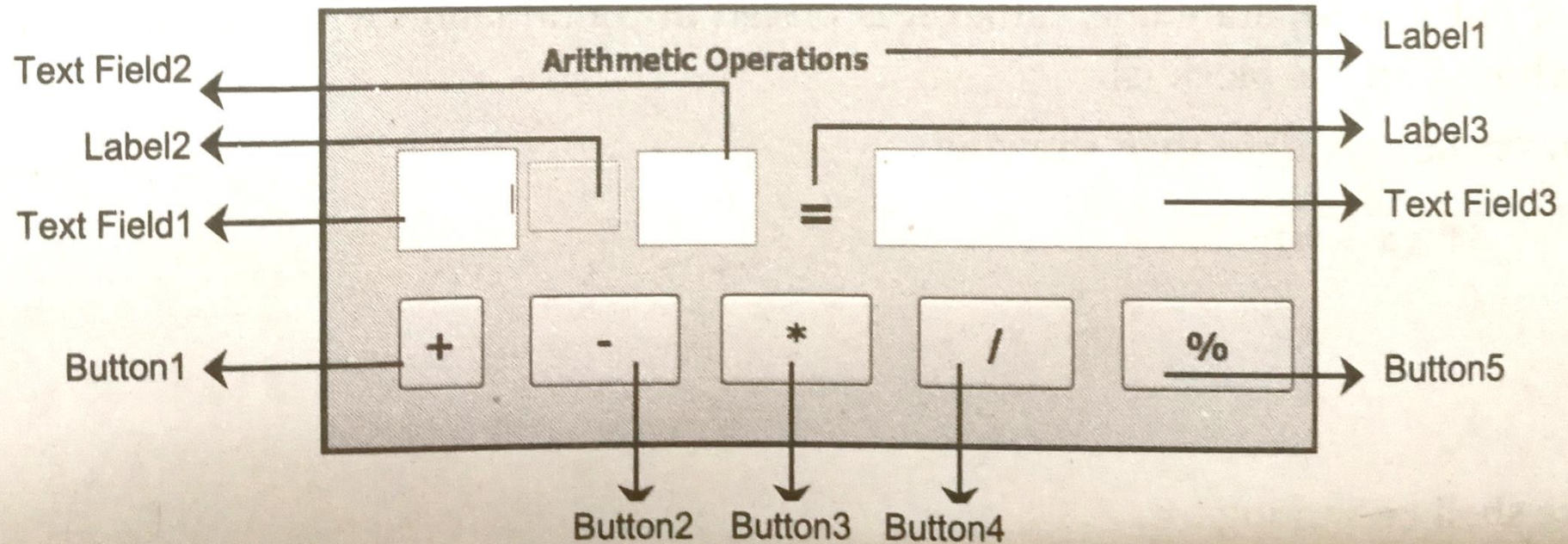




## Design an application that the shows all arithmetic operations



1. Start NetBeans and create a new project. Name the project as *Wks4\_1*. Also add new frame to it and name this frame as *ArithOpDemo*.
2. Now to achieve the desired result, design the frame having controls arranged as shown below :



WRITE



## CODE for '+' Button

WRITE

```
jLabel2.setText( "+" ) ;  
double num1 = Double.parseDouble(jTextField1.getText()) ;  
double num2 = Double.parseDouble(jTextField2.getText()) ;  
double num3 = num1 + num2 ;  
jTextField3.setText( " "+num3) ;
```



## CODE for '-' Button

**WRITE**

```
jLabel2.setText( "-" );  
double num1 = Double.parseDouble(jTextField1.getText()) ;  
double num2 = Double.parseDouble(jTextField2.getText()) ;  
double num3 = num1 - num2 ;  
jTextField3.setText( "" + num3 ) ;
```



## CODE for '\*' Button

WRITE

```
jLabel2.setText( "*" );  
double num1 = Double.parseDouble(jTextField1.getText()) ;  
double num2 = Double.parseDouble(jTextField2.getText()) ;  
double num3 = num1* num2 ;  
jTextField3.setText(" "+num3) ;
```



## CODE for '/' Button

**WRITE**

```
jLabel2.setText( "/");  
double num1 = Double.parseDouble(jTextField1.getText()) ;  
double num2 = Double.parseDouble(jTextField2.getText()) ;  
double num3 = num1 / num2 ;  
jTextField3.setText(" "+num3) ;
```



## CODE for '%' Button

WRITE

```
jLabel2.setText( "%");  
double num1 = Double.parseDouble(jTextField1.getText()) ;  
double num2 = Double.parseDouble(jTextField2.getText()) ;  
double num3 = num1 % num2 ;  
jTextField3.setText(“ “+num3) ;
```



## EXPRESSIONS

**WRITE**

An expression in JAVA is any valid combination of operators, constants and variable.

The expressions in JAVA can be of any type:

- Arithmetic expressions
- Compound expressions
- Relational(or Logical) expressions



## EXPRESSIONS



- **Arithmetic expressions**

**WRITE**

Arithmetic expressions can either be pure integer expressions or pure real expressions, sometimes a mixed expressions can also be formed.

**Integer expressions** are formed by connecting integer constants or integer variable using arithmetic operators.

```
final int count = 30 ;  
int I,J,K,X,Y,Z ;
```

Valid integer expressions:

i) I                      ii)  $K - X$                       iii)  $K + X - Y + \text{count}$                       iv)  $-J + K * Y$





## EXPRESSIONS



- Arithmetic expressions

**WRITE**

**Real expressions** are formed by connecting real constants or real variable using arithmetic operators.

final float bal = 250.53f ;  
float qty, amount, value ;  
double fin, inter ;

Valid real expressions:

- i) qty/amount      ii) qty \* value      iii) (amount + qty\* value) – bal  
iv) fin + qty \* inter      v) inter –(qty \* value )+ fin

## Some useful Math Functions available through Math Class



Functions	Action
<code>pow(x, y)</code>	This function returns x raised to $y$ ( $x^y$ )
<code>exp(x)</code>	This function returns e raised to $x$ ( $e^x$ )
<code>log(x)</code>	This function returns the natural logarithm of x
<code>sqrt(x)</code>	This function returns the square root of x
<code>ceil(x)</code>	This function returns the smallest whole number greater than or equal to x. (Rounded up)
<code>floor(x)</code>	This function returns the largest whole number less than or equal to x (Rounded down)
<code>rint(x)</code>	This function returns the rounded value of x
<code>abs(a)</code>	This function returns the absolute value of a
<code>max(a, b)</code>	This function returns the maximum of a and b
<code>min(a, b)</code>	This function returns the minimum of a and b.

**WRITE** →

It may be ints, longs, floats



**Write the corresponding JAVA expressions for the following mathematical expressions**



**WRITE**

(i)  $\sqrt{a^2 + b^2 + c^2}$

(ii)  $2 - ye^{2y} + 4y$

(iii)  $p + q / (r + s)^4$

(iv)  $|e^x - x|$

Solution.

(i) `Math.sqrt (a * a + b * b + c * c)`

(ii) `2 - y * Math.exp (2 * y) + 4 * y`

(iii) `p + q / Math.pow ((r + s), 4)`

(iv) `Math.abs (Math.exp (x) - x)`



## TYPE CONVERSION



The process of converting one predefined type into another is called **Type Conversion**.

**WRITE**

**JAVA** facilitates the type conversion in two forms:

- **Implicit type conversion**
- **Explicit type conversion**

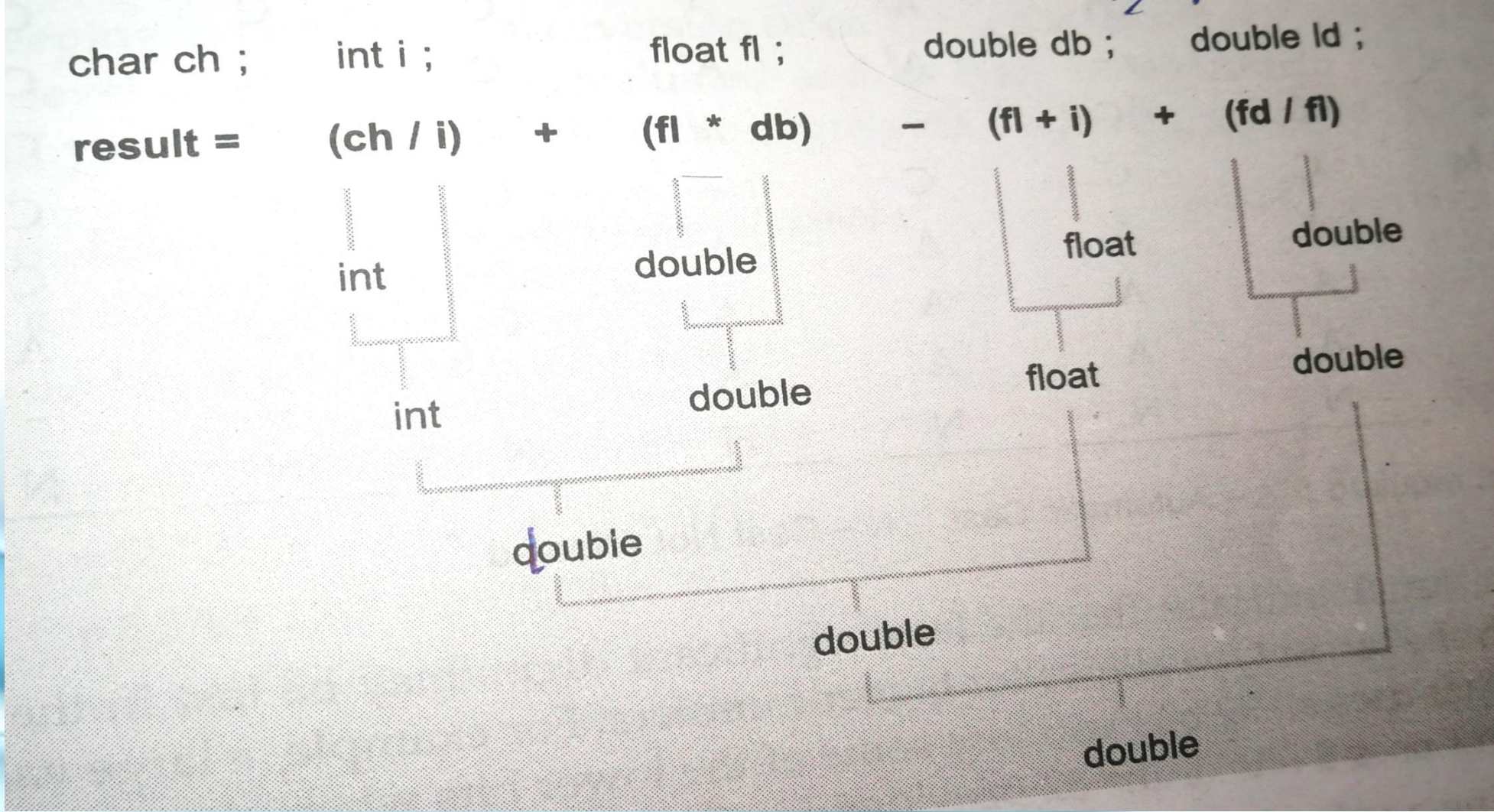
### **Implicit type conversion**

An implicit type conversion is performed by the compiler without the programmer's intervention. An implicit conversion is applied generally whenever differing data types are intermixed in an expression.

The implicit type conversion wherein data types are promoted is known as **Coercion**.



**WRITE** →





## TYPE CONVERSION

**WRITE**

### Explicit type conversion

An explicit type conversion is user defined that forces an expression to be of a specific type.

The explicit conversion of an operand to a specific type is called **Type Casting**.

**Example:** To make sure that  $(x + y/2)$  evaluates to type float, write it as:

**$(\text{float}) (x + y/ 2)$**



## EXPRESSION EVALUATION

**WRITE**

Expressions can either be pure expressions or mixed expressions. Pure expressions have all operands of same data types, mixed expressions have operands of mixed data types.

### **Evaluating pure expressions**

```
int a = 5, b = 2, c ;
```

```
a + b ; // will produce result 7 of int type
```

```
a/b ; // will produce result 2 of int type(it will not produce 2.5)
```

# EXPRESSION EVALUATION



## Evaluating mixed expressions

In JAVA when a mixed expression is evaluated, it is first divided into component sub-expressions upto the level of two operands and a operator. Then the type of sub-expression is decided keeping in mind general conversion rules.

Evaluate the following Java expression:

```
int a, mb = 2, k = 4 ;  
a = mb * 3/4 + k/4 + 8 - mb + 5/8
```

Solution:

```
a = mb * 3/4 + k/4 + 8 - mb + 5/8  
= (2 * 3) / 4 + 4/4 + 8 - 2 + 5/8  
= 6/4 + 1 + 8 - 2 + 0  
= 1 + 1 + 8 - 2 + 0  
= 10 - 2 = 8
```

**WRITE**





## Boolean (Logical) Expression

**WRITE**

The expressions that result into false or true are called Boolean expressions. The Boolean expressions are combinations of constants, variables and logical and relational operators.

**The following are examples of some valid Boolean expressions:**

- i)  $x > y$
- ii)  $(y + z) >= (x/z)$
- iii)  $(a + b) > c \ \&\& \ (c + d) > a$
- iv)  $(y > x) \ || \ (z < y)$
- v)  $x \ || \ y \ \&\& \ z$
- vi)  $(x)$
- vii)  $(-y)$
- viii)  $(x - y)$
- ix)  $(x - y) \ \&\& \ (!y < z)$
- x)  $x <= !y \ \&\& \ z$



**WRITE**

## Compound Expressions

A compound expression is the one which is made up by combining two or more simple expressions with the help of operator.

$(a + b) / (c + d)$  is a compound expression

$(a > b) || (b > c)$  is another compound expression



## JAVA STATEMENTS

**WRITE**

A statement in Java forms a complete unit of execution. The following types of expressions can be made into a statement by terminating the expression with a semicolon(;

- Assignment statement
- Any use of ++ or –
- Method calls
- Object creation expressions



## JAVA STATEMENTS

**WRITE**

**Here are some examples:**

```
x = 345.666 ; // assignment statement
y++ ; // increment statement
System.out.print(x) ; // method call
Integer integerobject = new Integer(4) ; // Object creation statement
double z = 345.768 ; // declaration statement
```



# JAVA STATEMENTS



**WRITE**

## **Block**

A block is a group of zero or more statements between balanced braces and can be used anywhere , a single statement is allowed.

Eg:

```
if(a < b)
{
.....
.....
.....
}
```



# JAVA STATEMENTS



## Null or Empty Statement

**WRITE**

In Java programs, statements are terminated with a semicolon(;). The simplest statement of them all is the empty, or null statement. A null statement is useful in those instances where the syntax of the language requires the presence of a statement but where the logic of the program does not.

It takes the following form:

;

It is a null or empty statement



# JAVA STATEMENTS



WRITE

## Constructs of Java Program

In a Java program, the specified statements can be executed either sequentially, selectively or repeatedly.

**Sequence-** Normally, the statements of a Java program are executed sequentially, i.e. in the same order as they are specified in a program.

**Selection-** When we need to execute the program selectively, we have to use a conditional statement.

**Iteration** -When there is a need for executing certain statements repeatedly, we have to use a loop structure or iteration statement.

# JAVA STATEMENTS



## 1. Selection Statements

The selection statements are used to execute a statement or a group of statements depending upon a given condition.



**WRITE**

There are following selection statements used in Java language:

**(i) The if statement** The if statement tests a given condition, if the given condition evaluates to true, the associated statements will be executed otherwise the control of the program will be transferred to the next statement.

The syntax of if statement is as follows:

```
if (condition)
statements;
```

e.g.

```
if (i==5)
System.out.println("Five");
```





## JAVA STATEMENTS



**WRITE**

**(ii) The if-else statement-** The if-else statement is an extension of the if statement. In this format, the given test condition will be evaluated, if the condition is evaluated to true one set of statements will be executed otherwise another set of statements will be executed.

The syntax of if-else statement is as follows:

```
if (condition)
    statement 1;
else
    statement 2;
```



## JAVA STATEMENTS



**WRITE**

**(iii) Nested if statement** When an if statement is inside another if or if-else statement, then it is called Nested if statement. The general syntax of this statement is as follows:

**(a)**

```
if(condition 1)
{ if(condition 2)
    statement 1;
  else
    statement 2;
}
else
    statement 3;
```



## JAVA STATEMENTS



**WRITE** →

(b)

```
if(condition 1)
    statement 1;
else
{ if(condition 2)
    statement 2;
  else
    statement 3;
}
```



## JAVA STATEMENTS



(c)

```
if (condition 1)
{ if(condition 2)
  statement 1;
  else
    statement 2;
}
else
{ if(condition 3)
  statement 3;
  else
    statement 4;
}
```

**WRITE** →



## JAVA STATEMENTS



(c)

```
if (condition 1)
{ if(condition 2)
  statement 1;
  else
  statement 2;
}
else
{ if(condition 3)
  statement 3;
  else
  statement 4;
}
```

**WRITE** →



## JAVA STATEMENTS



**(iv) The if-else-if ladder form** -In this kind of statement, a number of logical conditions are checked for executing various statements. This is based upon the sequence of nested if and is often called the if-else-if ladder.

**WRITE**

**General form:**

```
if (condition)
    statement;
else if (condition)
    statement;
else if (condition)
    statement;
else
    statement;
```



## JAVA STATEMENTS



**WRITE**

**The switch statement-** The switch statement is a multiple branch selection statement. This statement tests a given condition. On the basis of the evaluation of test condition against a list of integers or character constants, the associated statements will be executed. The switch statements searches for a specified match, when a match is found it will execute the statements specified thereafter till the end of the structure or until the break statement is found. The fall of control to the following cases of matching case, is called Fall-Through. We can use an optional default with the switch statement. It will be executed, when no match is found.



## JAVA STATEMENTS



**WRITE**

The general syntax of switch is as follows:

```
switch (expression)
{ case value1: statement 1;
  break;
  case value2: statement 2;
  break;
  case value3: statement 3;
  break;
  default: statement;
}
```





## JAVA STATEMENTS



**WRITE**

The general syntax of switch is as follows:

```
switch (expression)
{ case value1: statement 1;
  break;
  case value2: statement 2;
  break;
  case value3: statement 3;
  break;
  default: statement;
}
```



**Q.** Design the following application to find the largest and smallest of two integers entered in the two text boxes. Depending on the selection from the radio buttons(LARGEST OR SMALLEST) ,once you click OK button the answer should be displayed in the result in the text boxes.



**Labels**

**TextField Box**

**Button**

**TextField Box**

**Radio Buttons**

LARGEST AND SMALLEST OF TWO NUMBERS

ENTER THE FIRST NUMBER 10

ENTER THE SECOND NUMBER 25

RESULT 10

LARGEST

SMALLEST

OK

## Code for OK button



```
int n1,n2 ;
n1 = Integer.parseInt( jTextField1.getText( ) );
n2 = Integer.parseInt( jTextField2.getText( ) );
if(jRadioButton1.isSelected( ) )
{
    if (n1 > n2)
        jTextField3.setText(" "+n1);
    else
        jTextField3.setText(" "+n2);
}

if(jRadioButton2.isSelected())
{
    if(n1 < n2)
        jTextField3.setText(" "+n1);
    else
        jTextField3.setText(" "+n2);
}
```

**WRITE** →





## Code for disabling the textfield3



**WRITE** →

```
public class LARGEST extends javax.swing.JFrame {  
    /** Creates new form LARGEST */  
  
    public LARGEST() {  
  
        initComponents( ) ;  
        jTextField3.setEditable(false) ;  
    }  
}
```



**Q.** Design the following application to accept marks for 5 subjects of a student and find the total marks, average marks and grade depending on the total marks scored by the student. The grade should be calculated according to the following criteria.



**WRITE** →

<u>Total Marks</u>	<u>Grade</u>
400 – 500	A
300 – 399	B
200 - 299	C
100 – 199	D
less than 100	E



Design Preview [EXAM]

### MARK ENTRY FORM

ENGLISH	<input type="text" value="90"/>		
ACCOUNTANCY	<input type="text" value="90"/>	AVERAGE	<input type="text" value="90"/>
BUSINESS STUDIES	<input type="text" value="90"/>	TOTAL	<input type="text" value="450"/>
INFORMATION TECHNOLOGY	<input type="text" value="90"/>	GRADE	<input type="text" value="A"/>
ECONOMICS	<input type="text" value="90"/>		



**WRITE** →

## code for CALCULATE button click

```
int m1, m2, m3, m4, m5 ;
float avg=0,tot=0 ;
char grade ;
m1 = Integer.parseInt(jTextField1.getText( ) ) ;
m2 = Integer.parseInt(jTextField2.getText( ) ) ;
m3 = Integer.parseInt(jTextField3.getText( ) ) ;
m4 = Integer.parseInt(jTextField4.getText( ) ) ;
m5 = Integer.parseInt(jTextField5.getText( ) ) ;
tot = (m1+ m2+m3+m4+m5) ;
avg = tot / 5;
jTextField6.setText(" "+avg);
jTextField7.setText(" "+tot);
```





**WRITE**

```
if((tot >= 400) && (tot <= 500))
    grade = 'A' ;
else if((tot >= 300) && (tot <= 399))
    grade = 'B' ;
else if((tot >= 200) && (tot <= 299))
    grade = 'C' ;
else if((tot >= 100) && (tot <= 199))
    grade = 'D' ;
else
    grade = 'E' ;
jTextField8.setText(" "+grade);
```







code for EXIT Button click

**WRITE**

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



**Q.** Design the following GUI application to accept the Principal amount, Rate of interest and Number of years from the user and find the Simple Interest using the formula  $SI = (P * R * T) / 100$ . The simple interest should be displayed in the textfield box when you click the “CALCULATE SI” button. If you click the “ CLEAR” button all the values appearing in the all the textfield boxes should be cleared.



SIMPLE INTEREST CALCULATOR

ENTER THE PRINCIPAL AMOUNT	<input type="text" value="5000"/>	
ENTER THE RATE OF INTEREST	<input type="text" value="5"/>	<input type="button" value="CALCULATE SI"/>
ENTER THE NUMBER OF YEARS	<input type="text" value="2"/>	
SIMPLE INTEREST	<input type="text" value="500.0"/>	<input type="button" value="CLEAR"/>



## Code for CALCULATE SI BUTTON

**WRITE** →

```
float P,R,T,SI ;  
P = Integer.parseInt(jTextField1.getText( ) ) ;  
R = Integer.parseInt(jTextField2.getText( ) ) ;  
T = Integer.parseInt(jTextField3.getText( ) ) ;  
  
SI = P * R * T / 100 ;  
jTextField4.setText(" " + SI);
```



## Code for CLEAR BUTTON

**WRITE** →

```
jTextField1.setText(" ");  
jTextField2.setText(" ");  
jTextField3.setText(" ");  
jTextField4.setText(" ");
```



## Code for DISABLING THE TextField4 box

**WRITE** →

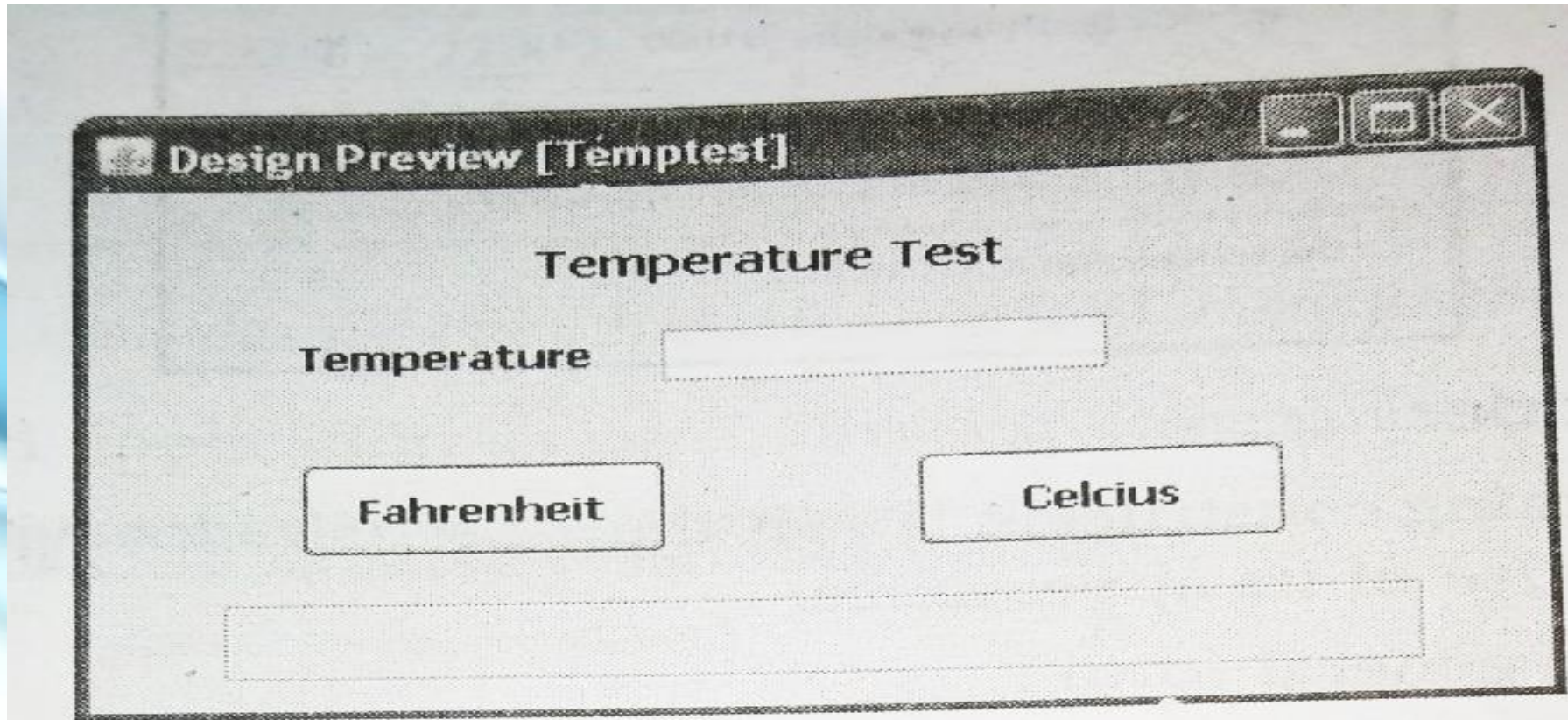
```
public SIANDCI() {  
    initComponents();  
    jTextField4.setEditable(false);  
  
}
```



**Q. Design an application that can test whether a temperature is freezing or not. It should be able to test the temperature for both Fahrenheit and Celsius scales. Use the Temperature  $32^{\circ}$  F and  $0^{\circ}$  C as the freezing temperature.**

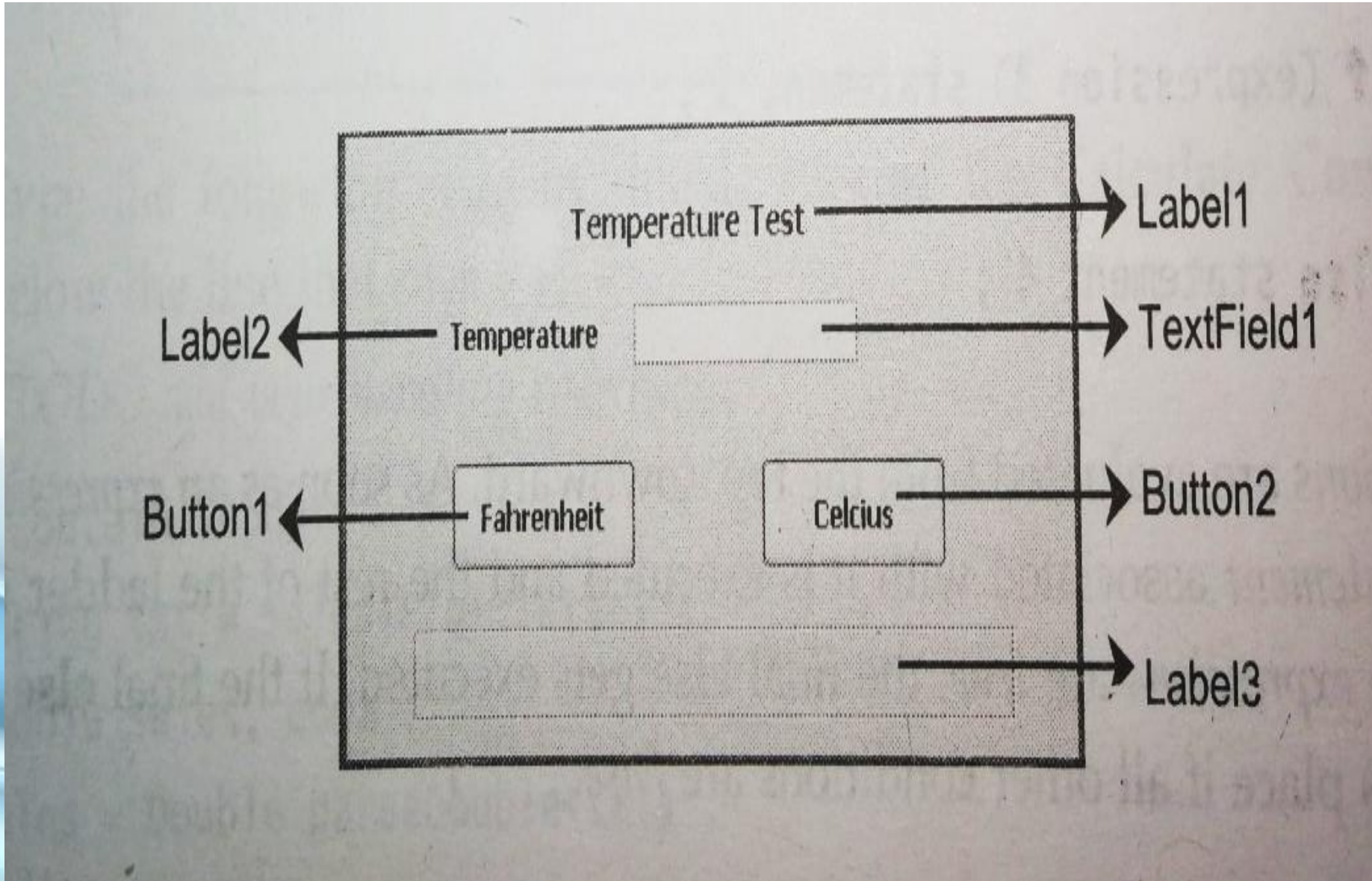


**WRITE** 





**WRITE** →



W

Temperature Test

Temperature

Temperature Test

Temperature





## CODE FOR FAHRENHEIT BUTTON



**WRITE**

```
double temp = Double.parseDouble( jTextField1.getText() );  
If(temp > 32)  
    jLabel3.setText("This temperature (in F0) is not freezing ");  
else  
    jLabel3.setText("This temperature (in F0) is freezing ");
```



## CODE FOR CELCIUS BUTTON

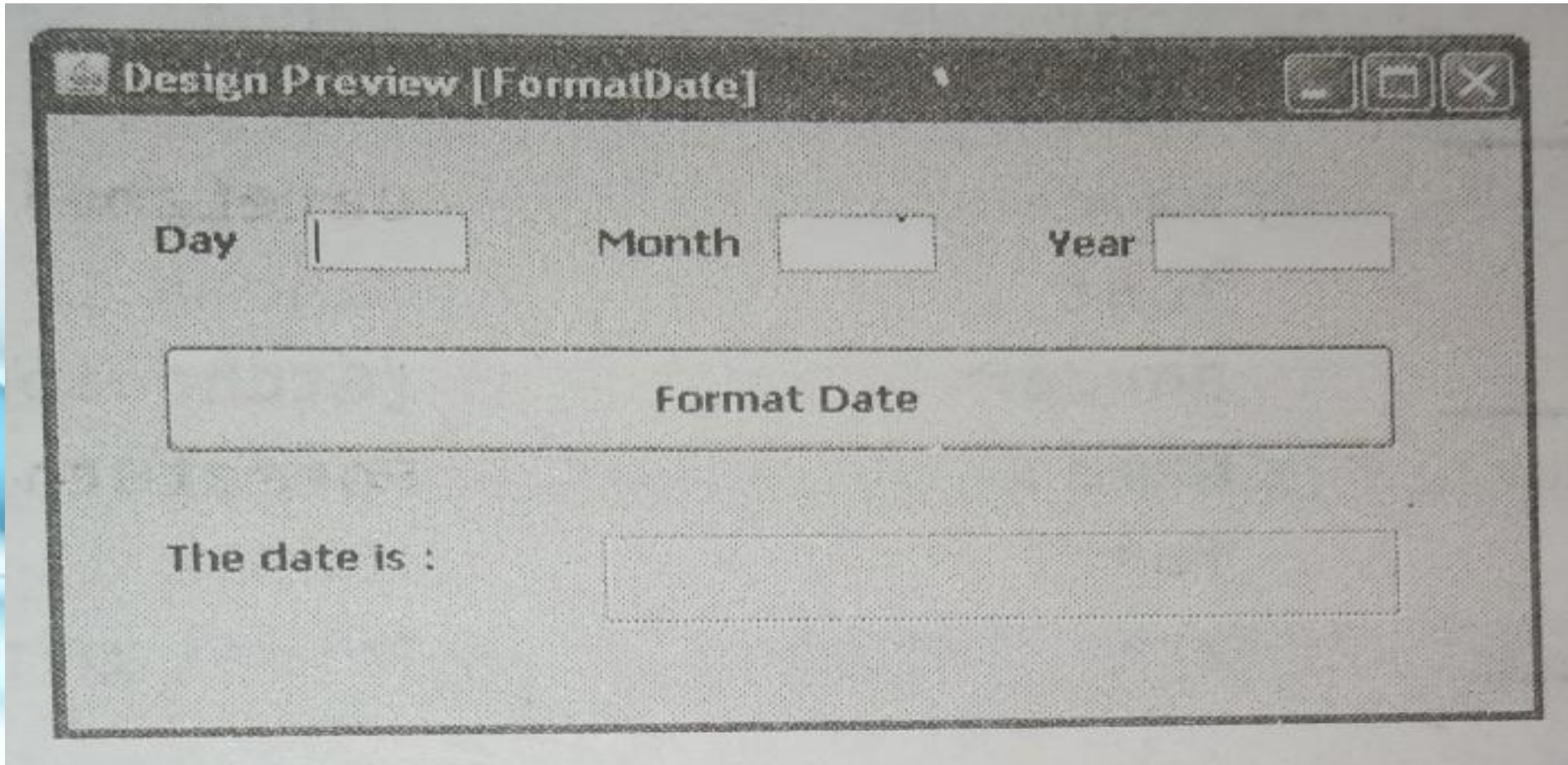


**WRITE**

```
double temp = Double.parseDouble( jTextField1.getText() );  
If(temp > 0)  
    jLabel3.setText("This temperature (in c0 ) is not freezing ");  
else  
    jLabel3.setText("This temperature (in c0 ) is freezing ");
```

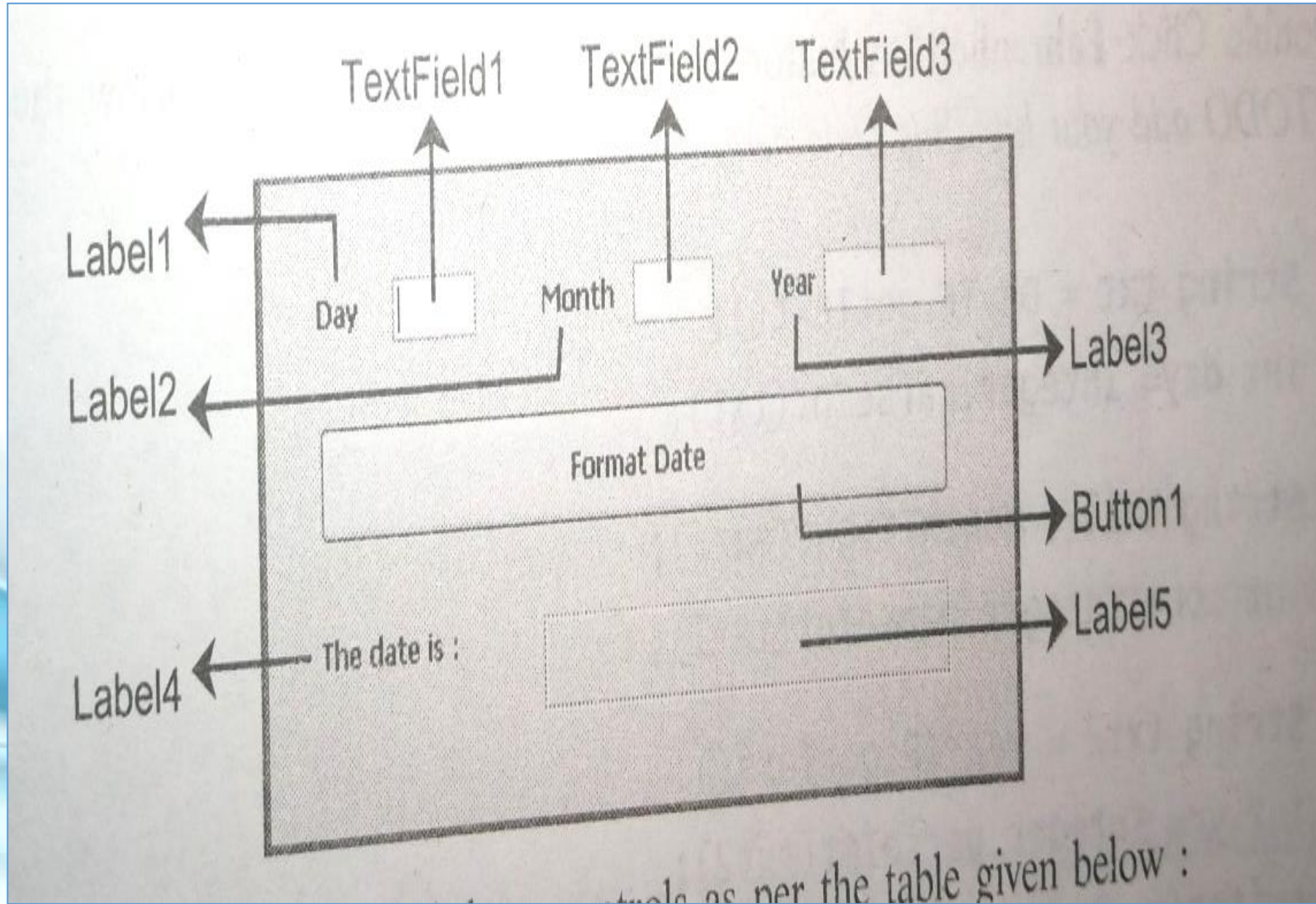


**Q. Create an application that obtains day(1-31), month(1-12) and year(>0) from the user and displays in dd-Mon-yyyy format. e.g., if user has entered as 13,04,2009 as day,month and year, then it should display date as 13-APR-2009**





**WRITE** →



... as per the table given below :



## CODE FOR FORMAT DATE BUTTON



```
int day = Integer.parseInt( jTextField1.getText() );  
int mon = Integer.parseInt( jTextField2.getText() );  
int year = Integer.parseInt( jTextField3.getText() );
```

**WRITE** →

```
switch(mon)  
{ case 1: jLabel5.setText(" " +day+ "-" + "JAN" + "-" + year ) ;  
      break ;  
  case 2: jLabel5.setText(" " +day+ "-" + "FEB" + "-" + year ) ;  
      break ;  
  case 3: jLabel5.setText(" " +day+ "-" + "MAR" + "-" + year ) ;  
      break ;
```



```
case 4: jLabel5.setText(" " + day + "-" + "APR" + "-" + year )  
        break ;  
case 5: jLabel5.setText(" " + day + "-" + "MAY" + "-" + year ) ;  
        break ;  
case 6: jLabel5.setText(" " + day + "-" + "JUN" + "-" + year ) ;  
        break ;  
case 7: jLabel5.setText(" " + day + "-" + "JUL" + "-" + year ) ;  
        break ;  
case 8: jLabel5.setText(" " + day + "-" + "AUG" + "-" + year ) ;  
        break ;  
case 9: jLabel5.setText(" " + day + "-" + "SEP" + "-" + year ) ;  
        break ;
```





**WRITE** →

```
case 10: jLabel5.setText(" " +day+ "-" + "OCT" + "-" + year );  
        break ;  
case 11: jLabel5.setText(" " +day+ "-" + "NOV" + "-" + year );  
        break ;  
case 12: jLabel5.setText(" " +day+ "-" + "DEC" + "-" + year );  
        break ;  
}
```



Q. Create an application that obtains an integer from the user in the Textbox and checks whether it is an odd number or even number and display it in the label as shown below. The result should be displayed when you click the CHECK button. The contents in the Textbox and the result displayed in the label should be cleared when you press the clear button.



**Textbox**

**LABEL CONTROLS**

**BUTTON CONTROLS**





## CODE FOR CHECK BUTTON

**WRITE**

```
int num =Integer.parseInt(jTextField1.getText());
if(num%2==0)
    jLabel4.setText("EVEN NUMBER");
else
    jLabel4.setText("ODD NUMBER");
```



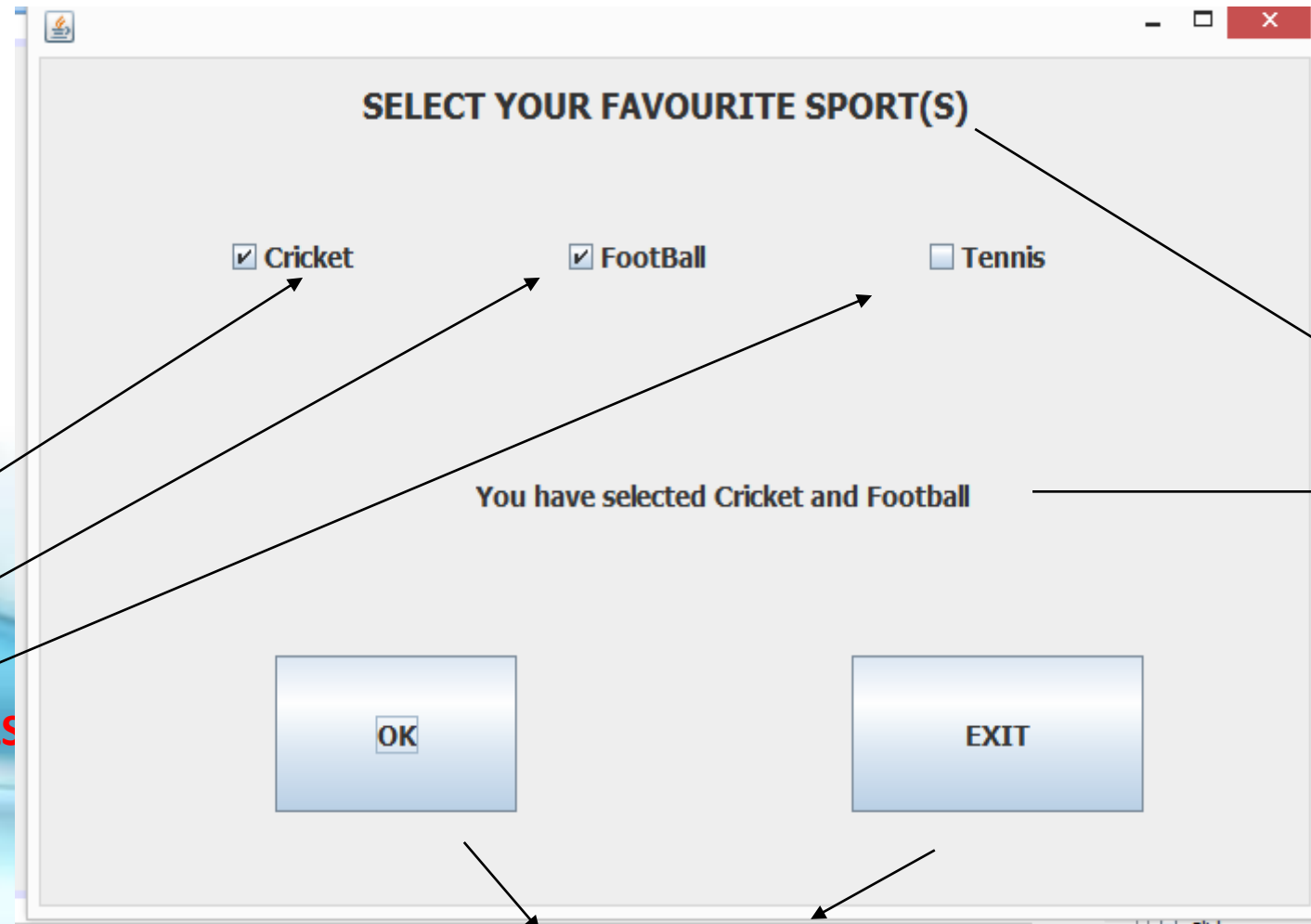
**WRITE**

CODE FOR CLEAR BUTTON

```
jLabel4.setText(" ");  
jTextField1.setText("");
```



Q. Create an application as shown below. The user can select any one, two or all three sports from the check box, depending on the selection when you click the OK button. The selected sports names will be displayed in a label as shown. When you click the EXIT button the program should stop.



**CHECKBOX CONTROLS**

**LABEL CONTROL**

**BUTTON CONTROLS**



## CODE FOR OK BUTTON



**WRITE**

```
if(jCheckBox1.isSelected() && jCheckBox2.isSelected() && jCheckBox3.isSelected( ))  
    jLabel1.setText("You have selected Cricket, Football and Tennis");  
  
else if(jCheckBox1.isSelected() && jCheckBox2.isSelected( ))  
    jLabel1.setText("You have selected Cricket and Football");  
  
else if(jCheckBox1.isSelected() && jCheckBox3.isSelected( ))  
    jLabel1.setText("You have selected Cricket and Tennis");  
  
else if(jCheckBox2.isSelected() && jCheckBox3.isSelected( ))  
    jLabel1.setText("You have selected Football and Tennis");
```



## CODE FOR OK BUTTON(Continuation)



**WRITE** →

```
else if(jCheckBox1.isSelected( ))
    jLabel1.setText("You have selected Cricket");

else if(jCheckBox2.isSelected( ))
    jLabel1.setText("You have selected Football");

else if(jCheckBox3.isSelected( ))
    jLabel1.setText("You have selected Tennis");
```



**WRITE**

CODE FOR EXIT BUTTON

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



Q.Create an application as shown below. It accepts Login id and password from the user. If correct password, it displays a message in a Label as “Password Valid, Entry granted” and if the password is incorrect then it will display in the Label “Password Invalid, Entry not granted”. **welcome** is the correct password.



LOGIN FORM

LOGIN ID ANIL KUMAR

PASSWORD .....

SUBMIT

Password Valid, Entry granted

**LABEL CONTROL**

**TEXTFIELD CONTROL**

**PASSWORDFIELD CONTROL**

**BUTTON CONTROL**



## CODE FOR SUBMIT BUTTON



String str;

**WRITE** →

```
str = new String(jPasswordField1.getPassword());
```

```
if( str.equals("welcome" ) )
```

```
    jLabel3.setText("Password Valid, Entry granted");
```

```
else
```

```
    jLabel3.setText("Password Invalid, Entry denied")
```





Q. Create an application as shown below. It accepts Bill amount from the user. The user has to select the customer type (NEW/SILVER/ GOLD / PLATINUM) from the COMBO BOX, if NEW - No discount, Silver- 10% Discount , Gold- 20% discount, Platinum- 30% Discount. The final amount should be displayed when the CALCULATE button is clicked. The program should end when STOP button is clicked.



**BILL DETAILS**

BILL AMOUNT	<input type="text" value="5000"/>
CUSTOMER TYPE	<input type="text" value="SILVER"/>
FINAL AMOUNT	<input type="text" value="4500.0"/>

**LABEL CONTROL**

**TEXTFIELD CONTROL**

**COMBOBOX CONTROL**

**TEXTFIELD CONTROL**

**BUTTON CONTROL**

## CODE FOR CALCULATE BUTTON



```
double Famt = 0;  
double BAmt = Double.parseDouble(jTextField1.getText());  
switch(jTextField1.getSelectedIndex( ))
```

```
{  
    case 0: Famt = BAmt;  
            break;  
    case 1: Famt = BAmt * 0.9 ;  
            break;  
    case 2: Famt = BAmt * 0.8 ;  
            break;  
    case 3: Famt = BAmt * 0.7 ;  
            break;  
    default : Famt = BAmt;  
}  
jTextField2.setText (" " +Famt ) ;
```

**WRITE**



**WRITE**

CODE FOR STOP BUTTON

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



Q. Create an application as shown below. It accepts Bill amount from the user. The user has to select the customer type (NEW/SILVER/ GOLD / PLATINUM) from the COMBO BOX, if NEW - No discount, Silver- 10% Discount , Gold- 20% discount, Platinum- 30% Discount. The final amount should be displayed when the CALCULATE button is clicked. The program should end when STOP button is clicked.



SELECT STREAM

SCIENCE     COMMERCE     HUMANITIES

OK    EXIT

You have selected COMMERCE stream

LABEL CONTROL

RADIOBUTTON CONTROL

BUTTON CONTROL

BUTTON CONTROL



## CODE FOR OK BUTTON



```
if(jRadioButton1.isSelected( ))
    jLabel2.setText("You have selected SCIENCE stream");

else if(jRadioButton2.isSelected( ))
    jLabel2.setText("You have selected COMMERCE stream");

else if(jRadioButton3.isSelected( ))
    jLabel2.setText("You have selected HUMANITIES stream");
```



CODE FOR EXIT BUTTON

**WRITE**

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



Q. Create an application as shown below. It accepts an alphabet in a TextField box and checks whether it is a vowel or a consonant and displays the message on another TextField box when the CHECK button is clicked. The program should end when STOP button is clicked.



**LABEL CONTROL**

VOWEL OR CONSONANT

ENTER AN ALPHABET

E

RESULT

vowel

CHECK

STOP

**TEXTFIELD CONTROL**

**BUTTON CONTROL**



## CODE FOR CHECK BUTTON

**WRITE**

```
String s ;  
s = jTextField1.getText();  
s = s.toLowerCase();  
char c = s.charAt(0);  
if(c== 'a' || c== 'e' || c== 'i' || c== 'o' || c== 'u' )  
    jTextField2.setText("vowel");  
else  
    jTextField2.setText("consonant");
```





**WRITE**

CODE FOR STOP BUTTON

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



**WRITE**

CODE FOR STOP BUTTON

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



## Dialog Box



A dialog box is a small separate sub-window that appears on the screen for providing/requesting information to/from the users. According to the Java documentation, “A dialog window is an independent sub-window meant to carry temporary notice apart from the main Swing Application Window”.

In Java language, the dialog box is supported by the Java Swing control. It supports the following types of controls:



**WRITE**



**WRITE**



## 1. JDialog

The JDialog is a Swing window dialog which provides the normal behaviour of a window. It displays a dialog box which has minimize, maximize and close icon in the title bar of the dialog box.

## 2. JOptionPane

In Java language, the JOptionPane is used to create a pop-up window with varied contents. It can display an alert message box or an input box.



## Structure of JOptionPane

The basic elements of JOptionPane are as follows:



WRITE

- 1. Icon-** The icon determines the type of dialog box or message box. There are four default options available. These are error, information, warning and question.
- 2. Message-** The message option is used to set the text information that we want to convey through the dialog box.
- 3. Input area-** The input area allows the user to provide a response in the form of an input. For user response, we can use JTextField, JComboBox or JList.
- 4. Button-** The button area display a set of buttons such as OK/Cancel, Yes/No, etc. The buttons can be customised according to the user's requirements.



## Dialog Box Type

We can create four types of predefined dialog boxes. These are as follows:

**1. Input dialog-** This type of dialog box is used to enter the data. We can use JTextField, JComboBox or JList. There are two buttons OK and Cancel.

To create an input dialog box, the **showInputDialog()** method is used.

**2. Confirm dialog-** This type of dialog box is used to ask from the user about the confirmation of any information. It includes buttons Yes / No, OK and Cancel, etc.

To create a confirm dialog box, the **showConfirmDialog( )** method is used.

WRITE





**WRITE**

**3. Message dialog-** The message dialog box is used to simply display some information to the user. It includes only a single OK button.

To create a message dialog box, the **showMessageDialog()** method is used.

**4. Option dialog** -The option dialog is a flexible type. It can be used to create a dialog box according to the user's need.

To create an option dialog box, the **showOptionDialog( )** method is used.



## Message Type

The icon of the JOptionPane is specified through the message type value. The possible values are as follows:



**WRITE**

Value	Equivalent Code	Icon
-1	JOptionPane.PLAIN_MESSAGE	No Icon
0	JOptionPane.ERROR_MESSAGE	
1	JOptionPane.INFORMATION_MESSAGE	
2	JOptionPane.WARNING_MESSAGE	
3	JOptionPane.QUESTION_MESSAGE	





## Option Type

In Java language, the `showConfirmDialog( )` and `showOptionDialog( )` methods are used to show the confirm dialog box and option dialog box respectively and have choice to select which buttons will be displayed in the dialog box. This could be done through the 'option type' in the properties window.

**WRITE** →



IDE value	Equivalent Code	Buttons Displayed
-1	<code>JOptionPane.DEFAULT_OPTION</code>	OK button
0	<code>JOptionPane.YES_NO_OPTION</code>	YES and NO buttons
1	<code>JOptionPane.YES_NO_CANCEL_OPTION</code>	YES, NO and CANCEL buttons
2	<code>JOptionPane.OK_CANCEL_OPTION</code>	OK and CANCEL buttons

# Tables



We know that some times we need to display some information in tabular form. In Java language, the JTable component of Swing API of Java is used to create a table



## Property

**Model** : This property is used to set the model that is the source of the data for the table.

## Methods

**1.int getColumnCount( )**- This method returns the number of columns in the table.

**2.int getRowCount( )**- This method returns the number of rows in the table.

**3.TableModel getModel( )**- This method returns the table model that provides the data displayed by this JTable.

**4.Object getValueAt (int row, int column)**- This method returns the cell value at row and column.



WRITE



**Q. Create an application to accept the age of a person and check whether eligible to vote or not. If age  $\geq 18$  Display in JOptionPane Dialog box , “You are eligible to VOTE” or else “You are NOT eligible to VOTE”.**



VOTERS AGE CHECKING

ELIGIBILITY TO VOTE

ENTER YOUR AGE      19

CHECK      STOP

Message

*i* You are eligible To VOTE

OK



**WRITE**

## **CODING FOR CHECK BUTTON**

```
int x = Integer.parseInt(jTextField1.getText()) ;  
if(x>=18)  
    JOptionPane.showMessageDialog(null,"You are eligible To VOTE");  
else  
    JOptionPane.showMessageDialog(null,"You are NOT eligible To VOTE");
```



**WRITE**

**CODING STOP BUTTON**

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



**Q. Create an application to accept the price for 1 apple and calculate the amount for one dozen and two dozen apples by clicking the buttons. The Amount to be paid is displayed in the non- editable text box.**



PRICS PER APPLE	AMOUNT TO BE PAID	
<input type="text" value="10"/>	<input type="text" value="120"/>	
<input type="button" value="ONE DOZEN"/>	<input type="button" value="TWO DOZEN"/>	<input type="button" value="STOP"/>



## **CODING FOR ONE DOZEN BUTTON**

**WRITE**

```
int P = Integer.parseInt(jTextField1.getText());  
int one_dozen = 12 * P ;  
jTextField2.setText(""+one_dozen);
```



## **CODING FOR TWO DOZEN BUTTON**

**WRITE**

```
int P = Integer.parseInt(jTextField1.getText());  
int two_dozen = 24 * P ;  
jTextField2.setText(""+two_dozen);
```





## **CODING FOR STOP BUTTON**

**WRITE**

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



**Q.Design a GUI application in java to convert kilograms into grams, litres into milliliters, rupees into paise using combobox and text fields. The result is displayed in a non-editable Textbox.**



COVERTION FROM ONE UNIT TO ANOTHER

ENTER THE VALUE

RESULT



## CODING FOR CONVERT BUTTON

```
double gm = 0, ml = 0, paise = 0;
double val = Double.parseDouble(jTextField1.getText());
switch(jTextField1.getSelectedIndex())
{
    case 0: gm = val * 1000;
            jTextField2.setText(+val+ "KG=" +gm+ "Grams");
            break;
    case 1: ml = val * 1000;
            jTextField2.setText(+val+ "L=" +ml+ "ML");
            break;
    case 2: paise = val*100;
            jTextField2.setText("Rs." +val+ "=" +paise+ "paise");
            break;
    default: jTextField2.setText("INVALID CHOICE");
}
```

**WRITE** →





## **CODING FOR STOP BUTTON**

**WRITE**

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



**Q. A book publishing house decided to go in for computerization. The database will be maintained at the back end but you have to design the front end for the company. You have to accept book code, Title, Author and Quantity sold from the user. The Price will be generated depending upon the book code(use radiobuttons).**

**WRITE**

**Net price should be calculated on the basis of the discount given.**

<u>Book code</u>	<u>Discount</u>
•Book seller -	25%
• School -	20%
•Customer -	5%





**ABC PUBLISHING**

**BOOK SELLER**                       **SCHOOL**                       **CUSTOMER**

**TITLE OF THE BOOK**

**AUTHOR**

**PRICE**

**QUANTITY**

**NET PRICE**





## **CODING FOR CALCULATE BUTTON**

**WRITE**

```
double amt =0, netamt = 0, qty = 0 ;
amt = Double.parseDouble(jTextField3.getText());
qty = Double.parseDouble(jTextField4.getText());
if(jRadioButton1.isSelected())
    netamt= qty * amt * 0.75 ;
else if(jRadioButton2.isSelected())
    netamt= qty * amt * 0.80 ;
else if(jRadioButton3.isSelected())
    netamt= qty * amt * 0.95 ;

jTextField5.setText(" "+netamt);
```



## **CODING FOR STOP BUTTON**

**WRITE**

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





**Q. A networking company decided to computerize its employee salary . Develop an application to store employee's personal data which will be saved in the back end. The front end should accept Name, Gender, Address and Basic Salary, Calculate DA and HRA, gross and net salary based on the following criteria.**

**WRITE**

<u>Basic</u>	<u>DA</u>	<u>HRA</u>	<u>Deductions</u>
$\geq 40000$	35% of Basic	37% of Basic	15% of Basic
$\geq 20000$	25% of Basic	32% of Basic	10% of Basic
$\geq 10000$	20% of Basic	30% of Basic	5% of Basic
$< 10000$	10% of Basic	20 % of Basic	2% of Basic

**Gross Sal = Basic + HRA + DA**

**Net Sal = Gross Sal - Deductions**



**XYZ NETWORKING COMPANY**

NAME	<input type="text" value="Sunil Gupta"/>		
GENDER	<input type="text" value="Male"/>		
ADDRESS	<input type="text" value="Ruwi, Oman"/>	GROSS SAL	<input type="text" value="15000.0"/>
BASIC	<input type="text" value="10000"/>	NET SAL	<input type="text" value="14500.0"/>
DA	<input type="text" value="2000.0"/>		
HRA	<input type="text" value="3000.0"/>	<input type="button" value="CALCULATE"/>	<input type="button" value="STOP"/>

## CODING FOR CALCULATE BUTTON



**WRITE** →

```
double bs=0,da=0,hra=0,ded=0,gross=0,net=0 ;
    bs = Double.parseDouble(jTextField4.getText());
    if(bs >= 40000)
    { da = 0.35 * bs ;
      hra = 0.37 * bs ;
      ded = 0.15 * bs ;
    }
    else if(bs >= 20000)
    { da = 0.25 * bs ;
      hra = 0.32 * bs ;
      ded = 0.10 * bs ;
    }
    else if(bs >= 10000)
    { da = 0.20 * bs ;
      hra = 0.30 * bs ;
      ded = 0.05 * bs ;
    }
}
```





**WRITE** →

```
else
{
    da = 0.10 * bs ;
    hra = 0.20 * bs ;
    ded = 0.02 * bs ;
}

gross = bs + hra + da ;
net = gross - ded ;
jTextField5.setText(" "+da) ;
jTextField6.setText(" "+hra) ;
jTextField7.setText(" "+gross) ;
jTextField8.setText(" "+net) ;
```





## **CODING FOR STOP BUTTON**

**WRITE**

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



Q. Design a GUI application to accept the cost price and selling price from the user in two text fields then calculate the profit or loss incurred.



**PROFIT AND LOSS OF AN ITEM**

ENTER THE COST PRICE	<input type="text" value="50"/>
ENTER THE SELLING PRICE	<input type="text" value="40"/>
RESULT	<input type="text" value="Loss = 10.0"/>



## **CODING FOR CALCULATE BUTTON**

**WRITE** →

```
double sp =0,cp =0 ;
cp = Double.parseDouble(jTextField1.getText());
sp = Double.parseDouble(jTextField2.getText());

if(sp > cp)
    jTextField3.setText("Profit = "+(sp - cp));
else if( cp == sp)
    jTextField3.setText(" No profit or Loss");
else
    jTextField3.setText("Loss = "+(cp - sp));
```



## **CODING FOR STOP BUTTON**

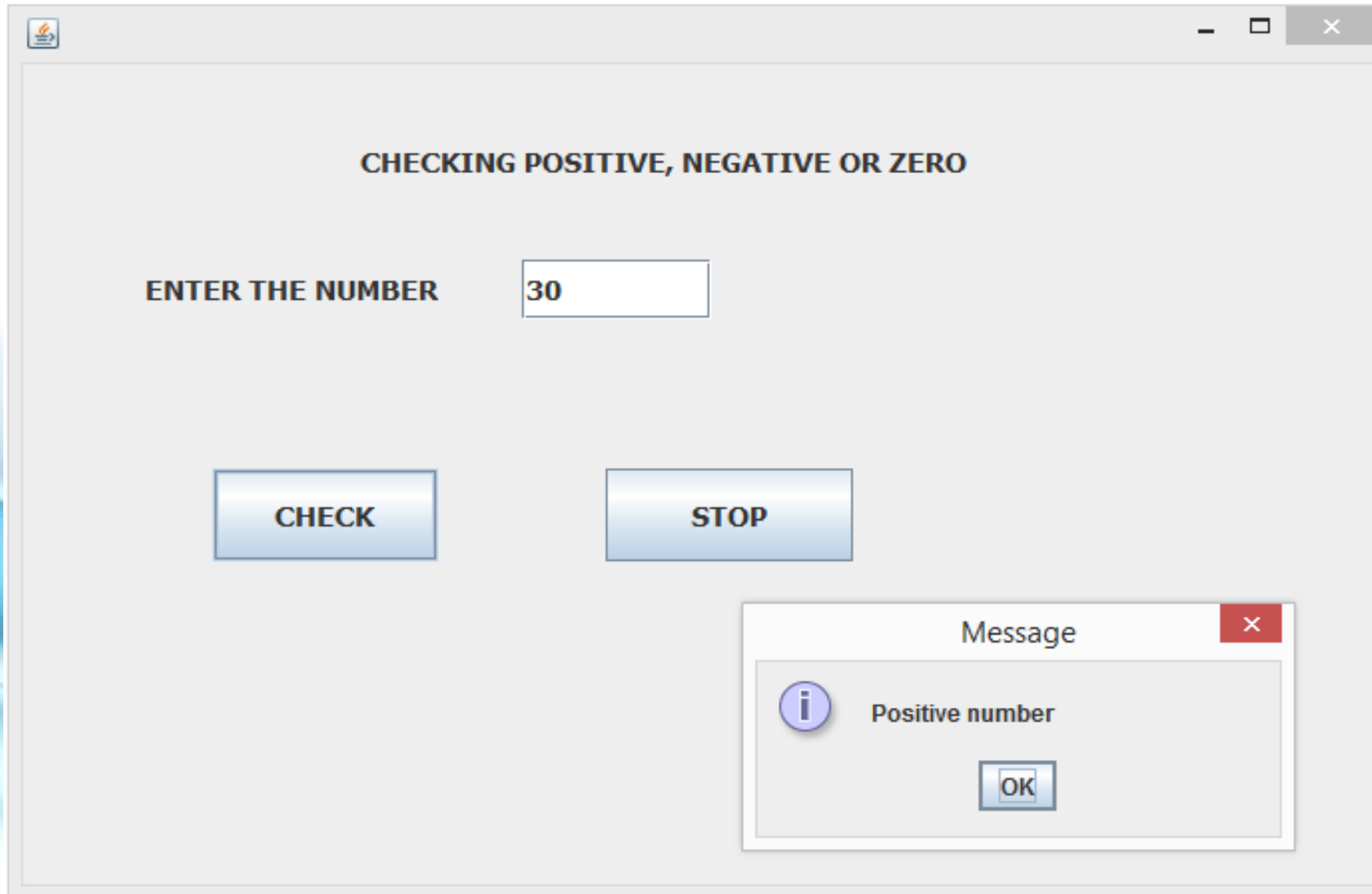
**WRITE**

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





Q. Design a GUI application to accept the a number in a TextFiled box and Check whether it is a Positive, Negative or Zero ,Display the answer in Option Pane dialog box.





## **CODING FOR CHECK BUTTON**

**WRITE** →

```
float num;  
    num = Float.parseFloat(jTextField1.getText());  
    if(num > 0)  
        JOptionPane.showMessageDialog(null, "Positive number");  
    else if(num < 0)  
        JOptionPane.showMessageDialog(null, "Negative number");  
    else  
        JOptionPane.showMessageDialog(null, "Number is Zero");
```



## **CODING FOR STOP BUTTON**

**WRITE**

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