



Vidyavardhini's

College of Engineering & Technology

Vasai Road (W)

Department of Artificial Intelligence & Data Science

Laboratory Manual Student Copy

Semester	III	Class	S.E.
Course Code	CSL304		
Course Name	Skill based Lab Course: Object Oriented Programming with Java		



Vidyavardhini's College of Engineering & Technology

Vision

To be a premier institution of technical education; always aiming at becoming a valuable resource for industry and society.

Mission

- To provide technologically inspiring environment for learning.
- To promote creativity, innovation and professional activities.
- To inculcate ethical and moral values.
- To cater personal, professional and societal needs through quality education.



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Department Vision:

To foster proficient artificial intelligence and data science professionals, making remarkable contributions to industry and society.

Department Mission:

- To encourage innovation and creativity with rational thinking for solving the challenges in emerging areas.
- To inculcate standard industrial practices and security norms while dealing with Data.
- To develop sustainable Artificial Intelligence systems for the benefit of various sectors.

Program Specific Outcomes (PSOs):

PSO1: Analyze the current trends in the field of Artificial Intelligence & Data Science and convey their findings by presenting / publishing at a national / international forum.

PSO2: Design and develop Artificial Intelligence & Data Science based solutions and applications for the problems in the different domains catering to industry and society.



Program Outcomes (POs):

Engineering Graduates will be able to:

- **PO1. Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
- **PO2. Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
- **PO3. Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
- **PO4. Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
- **PO5. Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
- **PO6. The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
- **PO7. Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
- **PO8. Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
- **PO9. Individual and teamwork:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
- **PO10. Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
- **PO11. Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
- **PO12. Life-long learning:** Recognize the need for and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Course Objective

1	To learn the basic concept of object-oriented programming
2	To study JAVA Programming language
3	To study various concepts of JAVA programming like multithreading, exception handling, packages etc.
4	To explain components of GUI based application.

Course Outcomes

CO	At the end of course students will be able to:	Action verbs	Bloom's Level
CSL304.1	Apply the Object Oriented Programming and basic programming constructs for solving problems using JAVA.	Apply	Apply (level 3)
CSL304.2	Apply the concept of packages, classes , objects and accept the input using Scanner and Buffered Reader Class.	Apply	Apply (level 3)
CSL304.3	Apply the concept of strings, arrays, and vectors to perform various operations on sequential data.	Apply	Apply (level 3)
CSL304.4	Apply the concept of inheritance as method overriding and interfaces for multiple inheritance.	Apply	Apply (level 3)
CSL304.5	Apply the concept of exception handling using try, catch, finally, throw and throws and multithreading for thread management.	Apply	Apply (level 3)
CSL304.6	Develop GUI based application using applets and AWT Controls.	Develop	Create (level 6)



Mapping of Experiments with Course Outcomes

List of Experiments	Course Outcomes					
	CSL304.1	CSL304.2	CSL304.3	CSL304.4	CSL304.5	CSL304.6
Implement a program using Basic programming constructs like branching and looping	3	-	-	-	-	-
Implement a program to accept the input from user using Scanner and Buffered Reader.	3	-	-	-	-	-
Implement a program that demonstrates the concepts of class and objects	-	3	-	-	-	-
Implement a program on method and constructor overloading.	-	3	-	-	-	-
Implement a program on Packages.	-	-	3	-	-	-
Implement a program on 2D array & strings functions.	-	-	3	-	-	-
Implement a program on single inheritance.	-	-	-	3	-	-
Implement a program on Multiple Inheritance with Interface.	-	-	-	3	-	-
Implement a program on Exception handling.	-	-	-	-	3	-



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Implement a program on Multithreading.	-	-	-	-	3	-
Implement a program on Applet or AWT Controls.	-	-	-	-	-	3
Mini Project based on the content of the syllabus (Group of 2-3 students)	-	-	-	-	-	3



INDEX

Sr. No.	Name of Experiment	D.O.P.	D.O.C.	Page No.	Remark
1	Implement a program using Basic programming constructs like branching and looping				
2	Implement a program to accept the input from user using Scanner and Buffered Reader.				
3	Implement a program that demonstrates the concepts of class and objects				
4	Implement a program on method and constructor overloading.				
5	Implement a program on Packages.				
6	Implement a program on 2D array & strings functions.				
7	Implement a program on single inheritance.				
8	Implement a program on Multiple Inheritance with Interface.				
9	Implement a program on Exception Handling.				
10	Implement a program on Multithreading.				
11	Implement a program on Applet or AWT Controls				
12	Mini Project based on the content of the syllabus (Group of 2-3 students)				

D.O.P: Date of performance

D.O.C : Date of correction



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Experiment No.1
Basic programming constructs like branching and looping
Date of Performance:
Date of Submission:



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Aim :- To apply programming constructs of decision making and looping.

Objective :- To apply basic programming constructs like Branching and Looping for solving arithmetic problems like calculating factorial of a no entered by user at command prompt .

Theory :-

Programming constructs are basic building blocks that can be used to control computer programs. Most programs are built out of a fairly standard set of programming constructs. For example, to write a useful program, we need to be able to store values in variables, test these values against a condition, or loop through a set of instructions a certain number of times. Some of the basic program constructs include decision making and looping.

Decision Making in programming is similar to decision making in real life. In programming also we face some situations where we want a certain block of code to be executed when some condition is fulfilled. A programming language uses control statements to control the flow of execution of program based on certain conditions. These are used to cause the flow of execution to advance and branch based on changes to the state of a program.

- if
- if-else
- nested-if
- if-else-if
- switch-case
- break, continue

These statements allow you to control the flow of your program's execution based upon conditions known only during run time.

A loop is a programming structure that repeats a sequence of instructions until a specific condition is met. Programmers use loops to cycle through values, add sums of numbers, repeat functions, and many other things. ... Two of the most common types of loops are the while loop and the for loop. The different ways of looping in programming languages are

- while



- do-while
- for loop
- Some languages have modified for loops for more convenience eg :- Modified for loop in java.

For and while loop is entry-controlled loops. Do-while is an exit-controlled loop.

Code: -

1} while loop

```
class Whileloop
```

```
{
```

```
    public static void main(String args[])
```

```
    {
```

```
        int a=4;
```

```
        while(a%2==0)
```

```
        {
```

```
            System.out.println("\n Number is even");
```

```
            break;
```

```
        }
```

```
    } }
```



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

A screenshot of a Windows Command Prompt window. The title bar reads 'Command Prompt'. The window shows the following text:

```
Microsoft Windows [Version 10.0.22821.3428]
(c) Microsoft Corporation. All rights reserved.

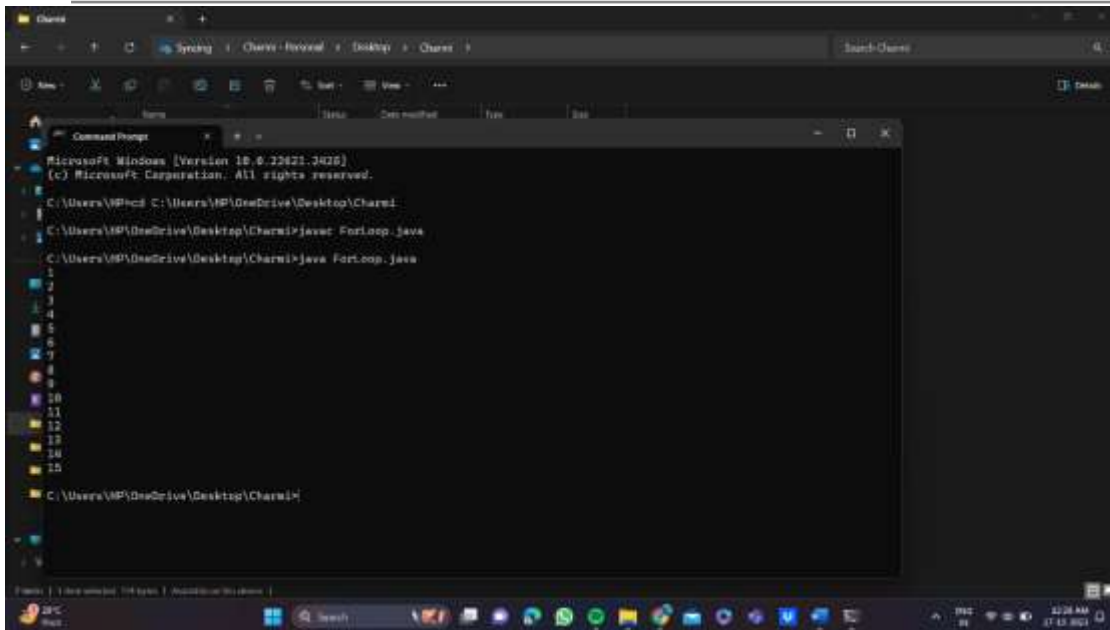
C:\Users\MP>cd C:\Users\MP\OneDrive\Desktop\Charvi
C:\Users\MP\OneDrive\Desktop\Charvi>javac WhileLoop.java
C:\Users\MP\OneDrive\Desktop\Charvi>java WhileLoop.java
Number is even
C:\Users\MP\OneDrive\Desktop\Charvi>
```

The taskbar at the bottom shows the Windows Start button, a search bar, and several application icons. The system clock in the bottom right corner indicates '11:07 AM' on '17/10/2023'.

2} for loop

class Forloop

```
{
    public static void main(String args[])
    {
        int x;
        for(x=1;x<=10;x++)
        {
            System.out.println(x);
        }
    }
}
```



3} dowhile loop

```
class Dowhileloop
```

```
{
```

```
    public static void main(String arg[])
```

```
    {
```

```
int a=0;
```

```
do
```

```
{
```

```
    if(a%20==0)
```

```
    {
```

```
        System.out.println(a);
```

```
    } a++;
```

```
} while(a<=100);
```

```
}
```

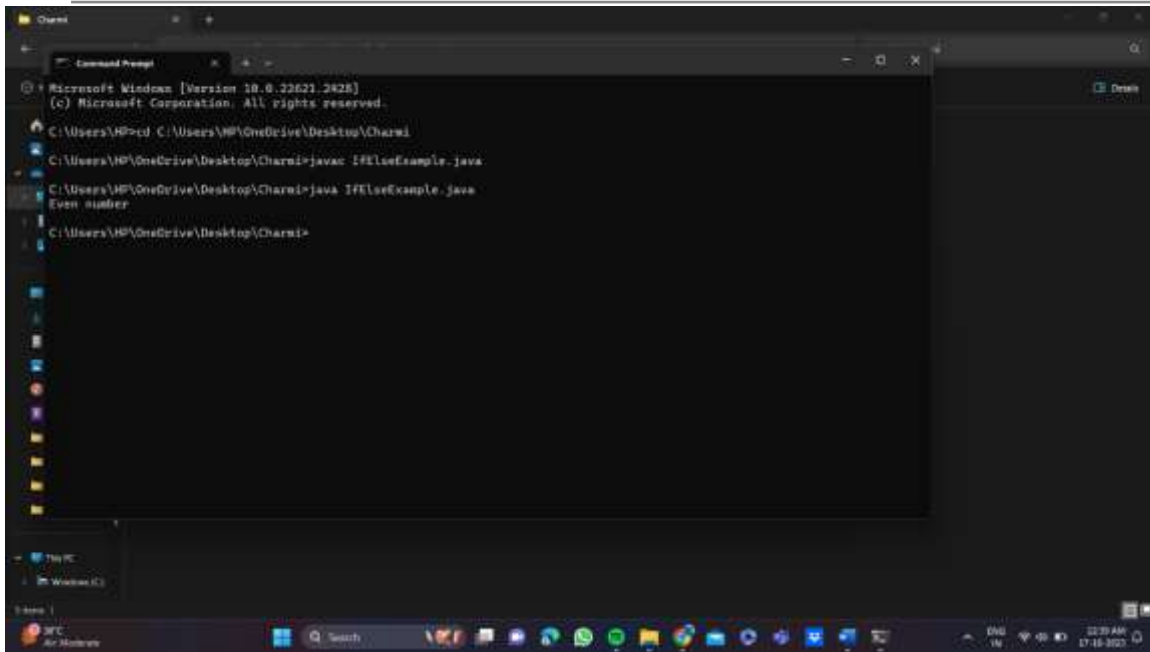
```
}
```

A screenshot of a Windows Command Prompt window. The title bar reads 'Command Prompt'. The window shows the following commands and output:

```
Microsoft Windows [Version 10.0.22021.1428]  
(c) Microsoft Corporation. All rights reserved.  
C:\Users\HP>cd C:\Users\HP\OneDrive\Desktop\Charmi  
C:\Users\HP\OneDrive\Desktop\Charmi>java DoubleLoop.java  
C:\Users\HP\OneDrive\Desktop\Charmi>java DoubleLoop.java  
0  
20  
40  
60  
80  
100  
C:\Users\HP\OneDrive\Desktop\Charmi>
```

4)if else

```
public class IfElseExample {  
    public static void main(String[] args) {  
        int number=10;  
        if(number%2==0){  
            System.out.println("Even number");  
        }else{  
            System.out.println("Odd number");  
        }  
    }  
}
```



5) Ladder if else

```
class SecJavaProgram
```

```
{
```

```
    public static void main(String args[])
```

```
    {
```

```
        int a=90;
```

```
        if(a>=90)
```

```
        {
```

```
            System.out.println("grade A");
```

```
        }
```

```
        else if(a>=80)
```

```
        {
```

```
            System.out.println("grade B");
```

```
        }
```



```
else if(a>=70)

{

System.out.println("grade c");

}

else if(a<70)

{

System.out.println("grade F");

}

}}
```

A screenshot of a Windows Command Prompt window. The window title is 'Command Prompt'. The text inside shows the execution of a Java program. The prompt is 'C:\Users\BP>cd C:\Users\BP\Desktop\Charmi'. The next line is 'C:\Users\BP\Desktop\Charmi>java Sec.SchoolProgram.java'. The output is 'grade A'. The prompt then changes to 'C:\Users\BP\Desktop\Charmi>'. The taskbar at the bottom shows the Windows logo, search bar, and several application icons. The system tray on the right shows the date and time as '11:38 AM 11/30/2021'.

6} nested if else

```
public class PositiveNegativeExample {

public static void main(String[] args) {

    int number=15;

    if(number>0){

        System.out.println("POSITIVE");

    }

}
```




Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

```
}else if(number<0){  
  
System.out.println("NEGATIVE");  
  
}else{  
  
System.out.println("ZERO");  
  
}  
  
}}
```

7} switch

class SwitchProgram

```
{  
  
public static void main(String args[])  
  
    {  
  
        int a = 1 ;  
  
        switch(a)  
  
        {  
  
            case 1 :
```



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

```
System.out.println("\n Monday");

break;

case 2 :

    System.out.println("\n Tuesday");

    break;

case 3 :

    System.out.println("\n Wednesday");

    break;

case 4 :

    System.out.println("\n Thursday");

    break;

case 5 :

    System.out.println("\n Friday");

    break;

case 6 :

    System.out.println("\n Saturday");

    break;

case 7 :

    System.out.println("\n Sunday");

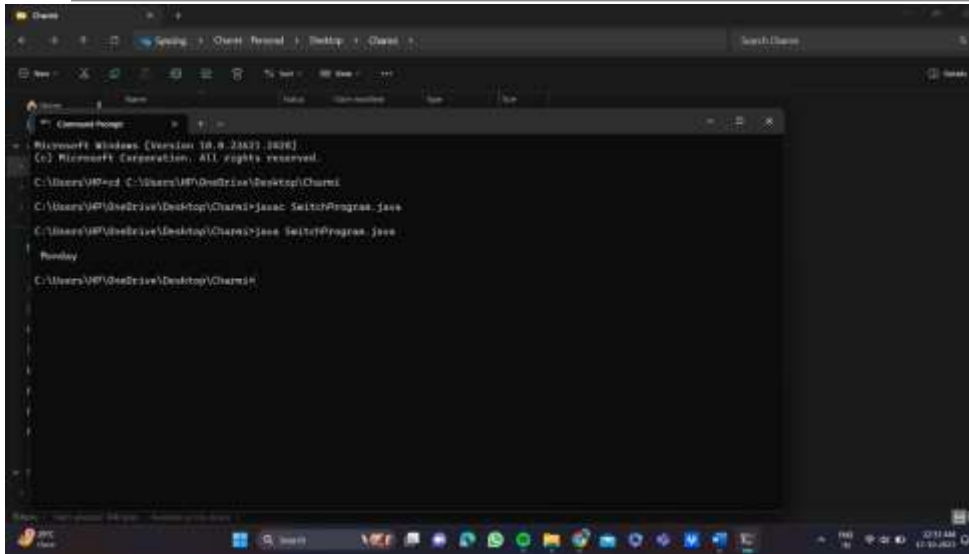
    break;

default :

    System.out.println("\n Not Valid");

}

} }
```



Conclusion:

Branching and looping are essential control structures in Java and many other programming languages that are useful for solving a wide range of problems. They provide the means to make decisions and repeat actions, making your code more dynamic and adaptable.

Branching (if statements) allows you to make decisions in your code based on conditions. You can execute different blocks of code depending on whether a condition is true or false.

Loops (for, while, and do-while) enable you to repeat a block of code multiple times, which is useful for tasks like processing arrays, lists, and performing iterative calculations. This repetition can be used to perform the same operation on each element of an array or list, or to perform a calculation a certain number of times.

In summary, branching and looping are powerful tools that allow you to create dynamic and adaptable code that can solve a wide range of problems.



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Experiment No.2
Accepting Input Through Keyboard
Date of Performance:
Date of Submission:



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Aim: To apply basic programming for accepting input through keyboard.

Objective: To use the facility of java to read data from the keyboard for any program

Theory:

Java brings various Streams with its I/O package that helps the user perform all the Java input-output operations. These streams support all types of objects, data types, characters, files, etc. to fully execute the I/O operations. Input in Java can be with certain methods mentioned below in the article.

Methods to Take Input in Java

There are two ways by which we can take Java input from the user or from a file

1. `BufferedReader` Class
2. `Scanner` Class

Using `BufferedReader` Class for String Input In Java

It is a simple class that is used to read a sequence of characters. It has a simple function that reads a character another read which reads, an array of characters, and a `readLine()` function which reads a line.

`InputStreamReader()` is a function that converts the input stream of bytes into a stream of characters so that it can be read as `BufferedReader` expects a stream of characters. `BufferedReader` can throw checked Exceptions.

Using `Scanner` Class for Taking Input in Java

It is an advanced version of `BufferedReader` which was added in later versions of Java. The scanner can read formatted input. It has different functions for different types of data types.

The scanner is much easier to read as we don't have to write throws as there is no exception thrown by it.

It was added in later versions of Java

It contains predefined functions to read an Integer, Character, and other data types as well.



Syntax of Scanner class

Scanner scn = new Scanner(System.in);

Code:

1} Scanner class

```
import java.util.Scanner;

class UserProgram
{
    public static void main(String args[])
    {
        Scanner a = new Scanner(System.in);
        System.out.println("Enter Name , Age and Salary:");
        String str = a.nextLine();
        int age = a.nextInt();
        Double salary = a.nextDouble();
        System.out.println("Name:" + str);
        System.out.println("Age:" + age);
        System.out.println("Salary:" + salary);
    }
}
```

A screenshot of a Windows Command Prompt window. The window title is 'Command Prompt'. The text inside shows the following commands and output:

```
Microsoft Windows [Version 10.0.22621.2420]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HP>cd C:\Users\HP\OneDrive\Desktop\Charvi
C:\Users\HP\OneDrive\Desktop\Charvi>javac UserProgram.java
C:\Users\HP\OneDrive\Desktop\Charvi>java UserProgram.java
Enter Name , Age and Salary:
Charvi
18
80000
Name: Charvi
Age: 18
Salary: 80000.0

C:\Users\HP\OneDrive\Desktop\Charvi>
```

2) Buffer reader class

```
package com.javatpoint;
```

```
import java.io.*;
```

```
public class BufferedReaderExample{
```

```
public static void main(String args[])throws Exception{
```

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

```
    BufferedReader br=new BufferedReader(r);
```

```
    System.out.println("Enter your name");
```

```
    String name=br.readLine();
```

```
    System.out.println("Welcome "+name);
```

```
}
```

```
}
```

A screenshot of a Windows Command Prompt window. The title bar reads 'Command Prompt'. The window shows the following text:

```
Microsoft Windows [Version 10.0.22621.2628]
(c) Microsoft Corporation. All rights reserved.

C:\Users\MP>cd C:\Users\MP\OneDrive\Desktop\Charmi

C:\Users\MP\OneDrive\Desktop\Charmi>java BufferedReaderExample.java

Enter your name
Charmi
Welcome Charmi

C:\Users\MP\OneDrive\Desktop\Charmi>
```

The background of the Command Prompt is dark, and the text is white. The Windows taskbar is visible at the bottom of the screen.

Conclusion:

In Java, both the `BufferedReader` and `Scanner` classes are commonly used for accepting user input from the command line or other input sources. Each of these classes has its own advantages and use cases.

The `BufferedReader` class is part of the `java.io` package and is primarily used for reading text from character input streams. It's efficient for reading large amounts of text efficiently. You can use `BufferedReader` to read input from a file or from the console by wrapping it around an `InputStreamReader` object.

The `Scanner` class is part of the `java.util` package and is a more high-level and user-friendly way to parse and tokenize input. It can be used for both reading from files and user input. `Scanner` is useful when you want to read input that is formatted in a particular way, such as a series of integers separated by commas. You can use `Scanner` to read input from the console or from a file.

In summary, both `BufferedReader` and `Scanner` classes are useful for accepting user input in Java, but they have different strengths and use cases.



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Experiment No. 3
Implement a program that demonstrates the concepts of class and objects
Date of Performance:
Date of Submission:



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Aim: Implement a program that demonstrates the concepts of class and objects

Objective: To develop the ability of converting real time entity into objects and create their classes.

Theory:

A class is a user defined blueprint or prototype from which objects are created. It represents the set of properties i.e., members and methods that are common to all objects of one type. In general, class declarations can include these components, in order:

1. Modifiers: A class can be public or has default access.
2. class keyword: class keyword is used to create a class.
3. Class name: The name should begin with a initial letter (capitalized by convention).
4. Superclass (if any): The name of the class's parent (superclass), if any, preceded by the keyword extends. A class can only extend (subclass) one parent.
5. Interfaces (if any): A comma-separated list of interfaces implemented by the class, if any, preceded by the keyword implements. A class can implement more than one interface.
6. Body: The class body surrounded by braces, { }.

An OBJECT is a basic unit of Object-Oriented Programming and represents the real-life entities. A typical Java program creates many objects, which interact by invoking methods.

An object consists of:

1. State: It is represented by attributes of an object. It also reflects the properties of an object.
2. Behavior: It is represented by methods of an object. It also reflects the response of an object with other objects.
3. Identity: It gives a unique name to an object and enables one object to interact with other objects.



Code:

1}

```
class Rectangle{
    int length;
    int width;
    void insert(int l, int w){
        length=l;
        width=w;
    }
    void calculateArea(){System.out.println(length*width);}
}
class TestRectangle1{
    public static void main(String args[]){
        Rectangle r1=new Rectangle();
        Rectangle r2=new Rectangle();
        r1.insert(7,9);
        r2.insert(5,12);
        r1.calculateArea();
        r2.calculateArea();
    }
}
```



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

A screenshot of a Windows Command Prompt window. The window title is "Command Prompt". The text inside shows the following commands and output:

```
Microsoft Windows [Version 10.0.22621.3428]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HP>cd C:\Users\HP\OneDrive\Desktop\Charvi
C:\Users\HP\OneDrive\Desktop\Charvi>javac Rectangle.java
C:\Users\HP\OneDrive\Desktop\Charvi>java Rectangle.java
63
68
C:\Users\HP\OneDrive\Desktop\Charvi>
```

The taskbar at the bottom shows the time as 12:41 AM on 17-10-2023.

Conclusion:

In Java, you can create a class template by using the `class` keyword followed by the class name. Inside the class, you can declare fields (attributes) to represent the state of objects, define constructors to initialize the object's state, and add methods to define the behavior and actions of the objects.

To create objects from a class, you can use the `new` keyword followed by the class constructor. The created objects can be assigned to variables.

You can access fields and call methods of an object using the dot notation.

In summary, creating a class template and its objects in Java is a straightforward process that involves defining a class with fields and methods, creating objects from the class using the `new` keyword, and accessing fields and methods using the dot notation.



Experiment No. 4
Implement a program on method and constructor overloading.
Date of Performance:
Date of Submission:

Aim: Implement a program on method and constructor overloading.

Objective: To use concept of method overloading in a java program to create a class with same function name with different number of parameters.

Theory:

CSL304: Object Oriented Programming with Java



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Method Overloading is a feature that allows a class to have more than one method having the same name, if their argument lists are different. It is similar to constructor overloading in Java, that allows a class to have more than one constructor having different argument lists.

Example: This example to show how method overloading is done by having different number of parameters for the same method name.

Class DisplayOverloading

```
{  
    public void disp(char c)  
    {  
        System.out.println(c);  
    }  
    public void disp(char c, int num)  
    {  
        System.out.println(c + " "+num);  
    }  
}
```

Class Sample

```
{  
    Public static void main(String args[])  
    {  
        DisplayOverloading obj = new DisplayOverloading();  
        Obj.disp('a');  
        Obj.disp('a',10);  
    }  
}
```

Output:

A

A 10

Java supports Constructor Overloading in addition to overloading methods. In Java, overloaded



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

constructor is called based on the parameters specified when a new is executed.

Sometimes there is a need of initializing an object in different ways. This can be done using constructor overloading.

For example, the Thread class has 8 types of constructors. If we do not want to specify anything about a thread then we can simply use the default constructor of the Thread class, however, if we need to specify the thread name, then we may call the parameterized constructor of the Thread class with a String args like this:

```
Thread t= new Thread (" MyThread ");
```

Code:

```
class Overload2
{
    public static void main(String args[])
    {
        System.out.println(Add.add(5,4));
        System.out.println(Add.add(2.80,3.12,9.00));
    }
}
class Add{
    static int add(int a,int b) {return a+b;}
    static double add(double a,double b,double c) {return a+b+c;}
}
```



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

A screenshot of a Windows command prompt window. The window title is 'Extended Prompt'. The command history shows the following commands and their outputs:

```
Microsoft Windows [Version 10.0.22621.2428]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HP>cd C:\Users\HP\OneDrive\Desktop\Charvi
C:\Users\HP\OneDrive\Desktop\Charvi>javac Overload2.java
C:\Users\HP\OneDrive\Desktop\Charvi>java Overload2.java
9
14.92
C:\Users\HP\OneDrive\Desktop\Charvi>
```

The taskbar at the bottom shows the system clock as 10:28 AM on 17/02/2024.

Conclusion:

In Java, function and constructor overloading involves creating multiple methods or constructors with the same name within a class but with different parameter lists.

Function overloading allows you to define multiple methods in a class with the same name but different parameter lists (number or types of parameters). This way, you can provide different behavior for the same operation depending on the inputs. The choice of which method to call is made at compile-time based on the provided arguments.

Constructor overloading is similar to function overloading but is used to define multiple constructors within a class with different parameter lists. Constructor overloading enables the creation of objects in various ways, depending on the arguments provided during object instantiation. Like function overloading, constructors can have different parameter types, but the number and types of parameters should differ to distinguish between them.

In summary, function and constructor overloading are useful techniques in Java that allow you to create multiple methods or constructors with the same name but different parameter lists. This way, you can provide different behavior for the same operation depending on the inputs or create objects in various ways.



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Experiment No. 5
Implement a program on Packages.
Date of Performance:
Date of Submission:



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Aim: To use packages in java.

Objective: To use packages in java to use readymade classes available in them using square root method in math class.

Theory:

A java package is a group of similar types of classes, interfaces and sub-packages. Packages are used in Java in order to prevent naming conflicts, to control access, to make searching/locating and usage of classes, interfaces, enumerations and annotations easier, etc.

There are two types of packages-

1. Built-in package: The already defined package like java.io.*, java.lang.* etc are known as built-in packages.
2. User defined package: The package we create for is called user-defined package.

Programmers can define their own packages to bundle group of classes/interfaces, etc. While creating a package, the user should choose a name for the package and include a package statement along with that name at the top of every source file that contains the classes, interfaces, enumerations, and annotation types that you want to include in the package. If a package statement is not used then the class, interfaces, enumerations, and annotation types will be placed in the current default package.

Code:

```
1} package mypack;  
   class Example  
   {  
       public static void main(String args[])  
       {  
           System.out.println("\n Hello I am an S.E. student");  
       }  
   }
```

A screenshot of a Windows operating system interface. In the foreground, a black command prompt window is open, displaying the following text:
Microsoft Windows [Version 10.0.22621.3438]
(c) Microsoft Corporation. All rights reserved.
C:\Users\Student\Desktop\Charvi_17>java mypack.Example
Hello I am an S.E. student
C:\Users\Student\Desktop\Charvi_17>
In the background, a file explorer window titled 'Charvi_17' is visible, showing the same directory. The Windows taskbar at the bottom includes the Start button, a search bar, and several pinned application icons. The system clock in the bottom right corner shows '12:38 PM' and '16/11/2023'.

Conclusion:

Autoencoders are a type of neural network used for data compression. They consist of an encoder that reduces the dimensions of the input data and a decoder that reconstructs the original data from the compressed representation. In Java, you can build an autoencoder for image compression. The results of image compression using autoencoders include smaller-sized images that maintain essential features, making them useful for storage and transmission. However, there is some loss of detail due to the compression process.

The architecture of an autoencoder typically consists of three main components: the encoder, the bottleneck layer, and the decoder. The encoder takes in the input data and compresses it into a lower-dimensional representation. The bottleneck layer is a compressed representation of the input data that is used as input to the decoder. The decoder takes in the compressed representation and reconstructs the original data.

In summary, autoencoders are a powerful tool for image compression that can produce smaller-sized images while maintaining essential features. However, there is some loss of detail due to the compression process.



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Experiment No. 6
Implement a program on 2D array & strings functions.
Date of Performance:
Date of Submission:



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Aim: To use 2D arrays and Strings for solving given problem.

Objective: To use 2D array concept and strings in java to solve real world problem

Theory:

- An array is used to store a fixed-size sequential collection of data of the same type.
- An array can be init in two ways:
 1. Initializing at the time of declaration:
`dataType[] myArray = {value0, value1, ..., valuek};`
 2. Dynamic declaration:
`dataType[] myArray = new dataType[arraySize];`
`myArray[index] = value;`
- Two – dimensional array is the simplest form of a multidimensional array. Data of only same data type can be stored in a 2D array. Data in a 2D Array is stored in a tabular manner which can be represented as a matrix.
- A 2D Array can be declared in 2 ways:
 1. Intializing at the time of declaration:
`dataType[][] myArray = { {valueR1C1, valueR1C2...}, {valueR2C1, valueR2C2...},...}`
 2. Dynamic declaration:
`dataType[][] myArray = new dataType[x][y];`
`myArray[row_index][column_index] = value;`

In Java, string is basically an object that represents sequence of char values. An array of characters works same as Java string. **Java String** class provides a lot of methods to perform operations on strings such as `compare()`, `concat()`, `equals()`, `split()`, `length()`, `replace()`, `compareTo()`, `intern()`, `substring()` etc.

1.String literal

To make Java more memory efficient (because no new objects are created if it exists already in the string constant pool).



Example:

```
String demoString = "GeeksforGeeks";
```

2. Using new keyword

- `String s = new String("Welcome");`
- In such a case, JVM will create a new string object in normal (non-pool) heap memory and the literal "Welcome" will be placed in the string constant pool. The variable s will refer to the object in the heap (non-pool)

Example:

```
String demoString = new String ("GeeksforGeeks");
```

Code:

```
1}
```

```
class Testarray3{  
    public static void main(String args[]){  
        int arr[][]={{ 1,2,3},{2,4,5},{4,4,5}};  
        for(int i=0;i<3;i++){  
            for(int j=0;j<3;j++){  
                System.out.print(arr[i][j]+" ");  
            }  
            System.out.println();  
        }  
    }  
}
```

A screenshot of a Windows Command Prompt window. The window title is 'Command Prompt'. The text inside shows the current directory as 'C:\Users\WP\Desktop\Charmi>' and the execution of a Java program: 'javac Testarray3.java'. The output of the program is displayed as '1 2 3', '2 3 4', and '3 4 5' on three separate lines. The Windows taskbar is visible at the bottom, showing the time as 11:23 AM on 17-11-2021.

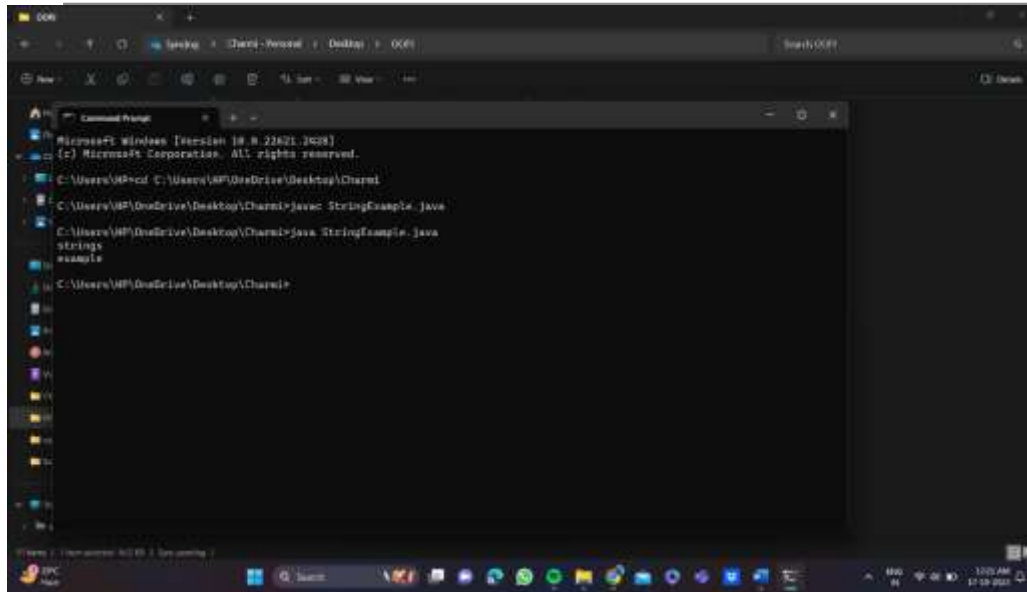
2}

```
class StringExample{  
public static void main(String args[]){  
String s1="java";  
char ch[]={'s','t','r','i','n','g','s'};  
String s2=new String(ch);  
String s3=new String("example");  
System.out.println(s2);  
System.out.println(s3);  
}}
```



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science



```
Microsoft Windows [Version 10.0.22H21.3028]
(c) Microsoft Corporation. All rights reserved.

C:\Users\WP\Desktop\Charmi> javac StringExample.java
C:\Users\WP\Desktop\Charmi> java StringExample
strings
example
```

Conclusion:

Comment on how you have used the concept of string and 2D array.

String Usage:

String s1 = "java";: Here, we've created a string s1 using a string literal.

char ch[] = {'s','t','r','i','n','g','s'};: We've defined a character array ch, and then we've created a string s2 using this character array. This demonstrates the creation of a string from an array of characters.

String s3 = new String("example");: This is another way to create a string, using the new keyword and a constructor. We have created s3 from the string literal "example".

2D Array Usage:

int arr[][] = {{1,2,3},{2,4,5},{4,4,5}};: We defined a 2D integer array arr with three rows and three columns. This represents a 3x3 grid of integer values.

The nested loops (for loops) in the Testarray3 class are used to iterate through the elements of the 2D array and print them out. This demonstrates how to access and display elements from a 2D array.



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Experiment No. 7
Implement a program on single inheritance.
Date of Performance:
Date of Submission:



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Aim: To implement the concept of single inheritance.

Objective: Ability to design a base and child class relationship to increase reusability.

Theory:

Single inheritance can be defined as a derived class to inherit the basic methods (data members and variables) and behaviour from a superclass. It's a basic is-a relationship concept exists here. Basically, java only uses a single inheritance as a subclass cannot extend more superclass.

Inheritance is the basic properties of object-oriented programming. Inheritance tends to make use of the properties of a class object into another object. Java uses inheritance for the purpose of code-reusability to reduce time by then enhancing reliability and to achieve run time polymorphism. As the codes are reused it makes less development cost and maintenance. Java has different types of inheritance namely single inheritance, multilevel, multiple, hybrid. In this article, we shall go through on basic understanding of single inheritance concept briefly in java with a programming example. Here we shall have a complete implementation in java.

Syntax:

The general syntax for this is given below. The inheritance concepts use the keyword 'extend' to inherit a specific class. Here you will learn how to make use of extending keyword to derive a class. An extend keyword is declared after the class name followed by another class name. Syntax is,

```
class base class
```

```
{.... methods
```

```
}
```

```
class derived class name extends base class
```

```
{
```

```
methods ... along with this additional feature
```

```
}
```

Java uses a keyword 'extends' to make a new class that is derived from the existing class. The inherited



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

class is termed as a base class or superclass, and the newly created class is called derived or subclass.

The class which gives data members and methods known as the base class and the class which takes the methods is known as child class.

Code:

```
1} class Animal{  
    void eat(){System.out.println("eating...");}  
}  
class Dog extends Animal{  
    void bark(){System.out.println("barking...");}  
}  
class TestInheritance{  
    public static void main(String args[]){  
        Dog d=new Dog();  
        d.bark();  
        d.eat();  
    }  
}}
```

A screenshot of a Windows command prompt window. The title bar says 'Command Prompt'. The text inside shows the following commands and output:
Microsoft Windows [Version 10.0.22621.3428]
(c) Microsoft Corporation. All rights reserved.
C:\Users\HP>cd C:\Users\HP\OneDrive\Desktop\Charvi
C:\Users\HP\OneDrive\Desktop\Charvi>javac Animal.java
C:\Users\HP\OneDrive\Desktop\Charvi>java Animal.java
barking...
eating...
C:\Users\HP\OneDrive\Desktop\Charvi>



Conclusion:

Comment on the Single inheritance.

In Java, single inheritance refers to the concept where a class can inherit the properties and behaviors of only one superclass. In other words, a Java class can have at most one direct parent class. This is a key aspect of Java's class inheritance hierarchy. In a single inheritance scenario, a Java class (subclass or derived class) can extend only one other class (superclass or base class). This means that it can inherit the fields and methods of that specific superclass.

Single inheritance is useful when you want to create a simple and straightforward class hierarchy. It allows you to reuse code from an existing class without introducing unnecessary complexity. However, it also has some limitations. For example, if you need to add functionality from multiple classes, you cannot do so directly with single inheritance. Instead, you may need to use interfaces or other techniques to achieve the desired functionality.

In summary, single inheritance is an important concept in Java's class hierarchy that allows you to reuse code from an existing class. While it has some limitations, it is useful for creating simple and straightforward class hierarchies.



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Experiment No. 8
Implement a program on multiple inheritance with interface.
Date of Performance:
Date of Submission:

Aim: Implement a program on multiple inheritance with interface.

Objective: Implement multiple inheritance in a program to perform addition, multiplication and transpose



operations on a matrix. Create an interface to hold prototype methods and create a class to read input. Inherit a new class from this interface and class. In main class create object of this child class and invoke required methods.

Theory:

- In Multiple inheritance, one class can have more than one superclass and inherit features from all parent classes. Java does not support multiple inheritance with classes. In java, we can achieve multiple inheritance only through Interfaces.
- An interface contains variables and methods like a class but the methods in an interface are abstract by default unlike a class. If a class implements multiple interfaces, or an interface extends multiple interfaces, it is known as multiple inheritance.
- However, Java supports multiple interface inheritance where an interface extends more than one super interfaces.
- A class implements an interface, but one interface extends another interface. Multiple inheritance by interface occurs if a class implements multiple interfaces or also if an interface itself extends multiple interfaces.
- The following is the syntax used to extend multiple interfaces in Java:

```
access_specifier interface subinterfaceName extends superinterface1, superinterface2, ..... {  
  
// Body  
  
}
```

Code:

```
class MultInherit{  
public static void main(String args[])  
{  
Pig a=new Pig();  
a.animalsound();  
a.sleep();  
}  
}  
interface Animal{  
public void animalsound();  
public void sleep();  
}  
class Pig implements Animal{  
public void animalsound(){
```



Vidya Vardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

```
}  
System.out.println("The Pig says:");  
public void sleep(){  
System.out.println("zzzzzzzz");  
}  
}
```

```
Microsoft Windows [Version 10.0.22621.2420]  
(c) Microsoft Corporation. All rights reserved.  
  
C:\Users\HP>cd C:\Users\HP\OneDrive\Desktop\Charmi  
C:\Users\HP\OneDrive\Desktop\Charmi>javac Multinherit.java  
C:\Users\HP\OneDrive\Desktop\Charmi>java Multinherit.java  
The Pig says: woo-woo  
zzzzzzzz  
C:\Users\HP\OneDrive\Desktop\Charmi>
```

Conclusion:

Comment on how interface are useful and implemented using java.

Interfaces in Java are a fundamental concept that allows you to define a contract specifying a set of methods that implementing classes must adhere to.

Abstraction: Interfaces allow you to define a contract or a set of methods without specifying the implementation. This promotes abstraction, enabling you to focus on what a class should do rather than how it should do it.



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Experiment No. 9
Implement a program on Exception handling.
Date of Performance:
Date of Submission:

Aim: Implement a program on Exception handling.

Objective: To able handle exceptions occurred and handle them using appropriate keyword

Theory:

The Exception Handling in Java is one of the powerful mechanisms to handle the runtime errors so that the normal flow of the application can be maintained.

Exception Handling is a mechanism to handle runtime errors such as `ClassNotFoundException`, `IOException`, `SQLException`, `RemoteException`, etc.

Java Exception Keywords



Java provides five keywords that are used to handle the exception. The following table describes each.

Keyword	Description
try	The "try" keyword is used to specify a block where we should place an exception code. It means we can't use try block alone. The try block must be followed by either catch or finally.
catch	The "catch" block is used to handle the exception. It must be preceded by try block which means we can't use catch block alone. It can be followed by finally block later.
finally	The "finally" block is used to execute the necessary code of the program. It is executed whether an exception is handled or not.
throw	The "throw" keyword is used to throw an exception.
throws	The "throws" keyword is used to declare exceptions. It specifies that there may occur an exception in the method. It doesn't throw an exception. It is always used with method signature.

```
public class JavaExceptionExample{  
  
    public static void main(String args[]){  
  
        try{  
  
            //code that may raise exception  
  
            int data=100/0;  
  
        }catch(ArithmeticException e){System.out.println(e);}  
  
        //rest code of the program  
  
        System.out.println("rest of the code...");  
  
    }  
  
}
```

Output:

```
Exception in thread main java.lang.ArithmeticException:/ by zero  
rest of the code...
```

Code:



```
1} Try-catch
class Main2
{
public static void main(String args[])
{
try{
int divideByZero = 8/0;
System.out.println("Rest of code in try block");
}

catch (ArithmeticException e) {
System.out.println("ArithmeticException => " + e.getMessage());
}
}
}
```

```
Microsoft Windows [Version 10.0.22621.2428]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HP>cd C:\Users\HP\OneDrive\Desktop\Charmi
C:\Users\HP\OneDrive\Desktop\Charmi>javac Main2.java
C:\Users\HP\OneDrive\Desktop\Charmi>java Main2.java
ArithmeticException: / by zero

C:\Users\HP\OneDrive\Desktop\Charmi>
```

```
2} finally
class TestFinallyBlock {
public static void main(String args[]){
try{
int data=25/5;
System.out.println(data);
}
catch(NullPointerException e){
System.out.println(e);
}
finally {
System.out.println("finally block is always executed");
}

System.out.println("rest of the code...");
}
```

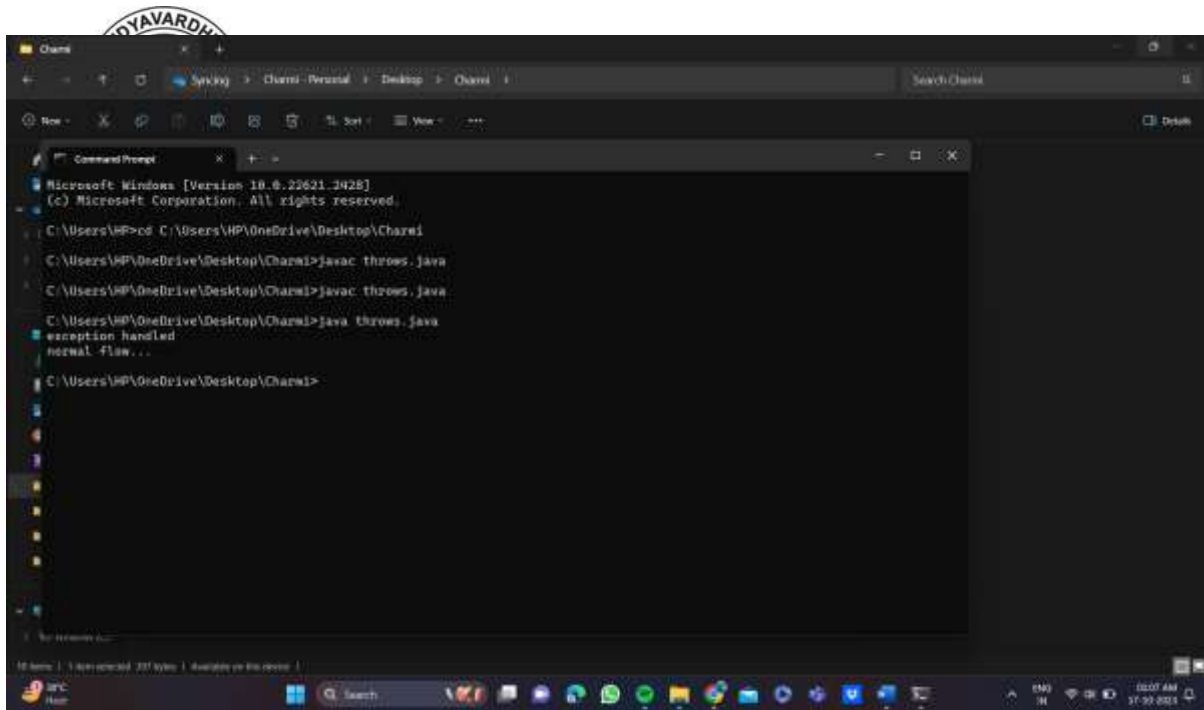


```
Microsoft Windows [Version 10.0.22621.3428]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HP>cd C:\Users\HP\OneDrive\Desktop\Charvi
C:\Users\HP\OneDrive\Desktop\Charvi>javac TestFinallyBlock.java
C:\Users\HP\OneDrive\Desktop\Charvi>java TestFinallyBlock.java
5
finally block is always executed
rest of the code...
C:\Users\HP\OneDrive\Desktop\Charvi>
```

```
}throws
import java.io.IOException;
class Testthrows2{
    public static void main(String args[]){
        try{
            M m=new M();
            m.method();
        }catch(Exception e){System.out.println("exception handled");}

        System.out.println("normal flow...");
    }
}
class M {
    void method() throws IOException {
        throw new IOException("device error");
    }
}
```

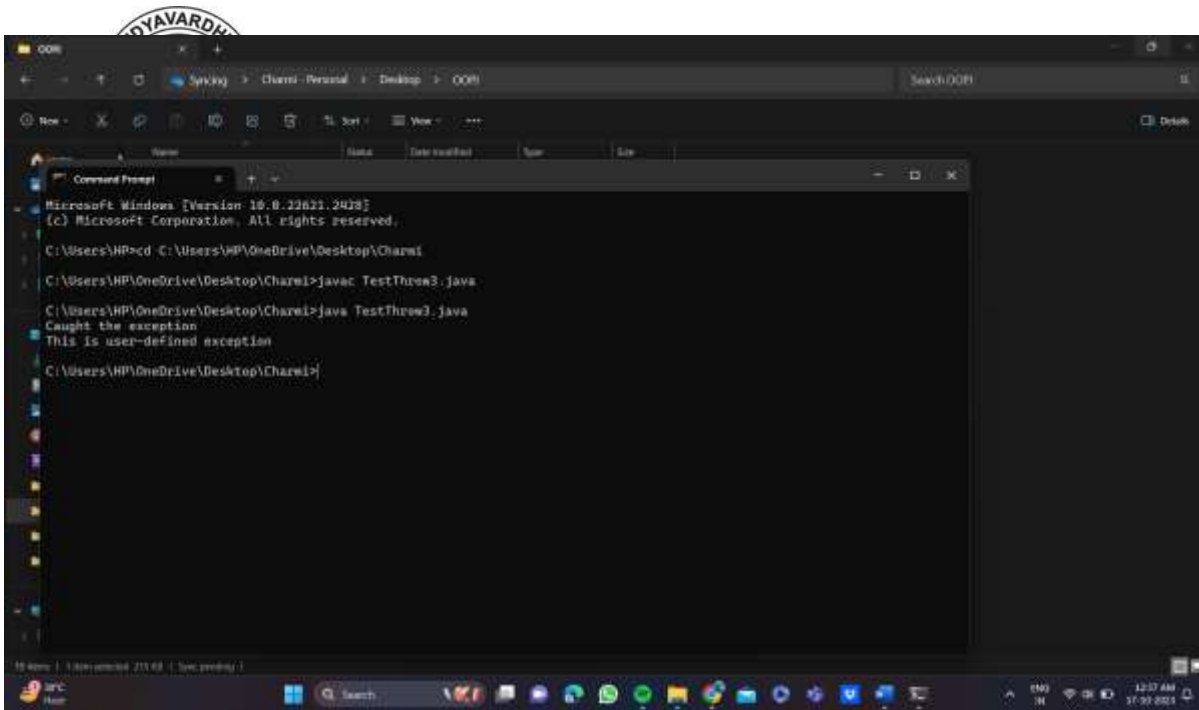


ogy

4} throw

```
class TestThrow3
{
    public static void main(String args[])
    {
        try
        {
            throw new UserDefinedException("This is user-defined exception");
        }
        catch (UserDefinedException ude)
        {
            System.out.println("Caught the exception");
            System.out.println(ude.getMessage());
        }
    }
}

class UserDefinedException extends Exception
{
    public UserDefinedException(String str)
    {
        super(str);
    }
}
```



```
Microsoft Windows [Version 10.0.22621.2428]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HP>cd C:\Users\HP\OneDrive\Desktop\Charvi
C:\Users\HP\OneDrive\Desktop\Charvi>javac TestThrow3.java
C:\Users\HP\OneDrive\Desktop\Charvi>java TestThrow3.java
Caught the exception
This is user-defined exception
C:\Users\HP\OneDrive\Desktop\Charvi>
```

ology

Conclusion:

In Java, exceptions are handled using a combination of the try, catch, finally, and throw keywords. The primary mechanism for handling exceptions is the try-catch block. Code that may potentially throw an exception is placed within a try block, and you provide one or more catch blocks to handle specific types of exceptions. You can also use a finally block after the try-catch blocks. Code within the finally block is executed regardless of whether an exception was thrown or not. It's typically used for cleanup actions (e.g., closing resources). You can use the throw keyword to explicitly throw an exception within your code. This is often done when you encounter an exceptional situation that your code can't handle, and you want to pass the control to an exception handler.



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Experiment No. 10
Implement program on Multithreading
Date of Performance:
Date of Submission:



Vidyavardhini's College of Engineering and Technology

Aim: Implement program on Multithreading

Department of Artificial Intelligence & Data Science

Objective:

Theory:

Multithreading in Java is a process of executing multiple threads simultaneously.

A thread is a lightweight sub-process, the smallest unit of processing. Multiprocessing and multithreading, both are used to achieve multitasking.

However, we use multithreading than multiprocessing because threads use a shared memory area. They don't allocate separate memory area so saves memory, and context-switching between the threads takes less time than process.

Java Multithreading is mostly used in games, animation, etc.

Java provides **Thread class** to achieve thread programming. Thread class provides constructors and methods to create and perform operations on a thread. Thread class extends Object class and implements Runnable interface.

There are two ways to create a thread:

1. By extending Thread class
2. By implementing Runnable interface.

Thread class:

Thread class provide constructors and methods to create and perform operations on a thread. Thread class extends Object class and implements Runnable interface.

1) Java Thread Example by extending Thread class

FileName: Multi.java

```
class Multi extends Thread{
    public void run(){
        System.out.println("thread is running...");
    }
    public static void main(String args[]){
        Multi t1=new Multi();
        t1.start();
    }
}
```

Output:

```
thread is running...
```



2) Java Thread Example by implementing Runnable interface

FileName: Multi3.java

Department of Artificial Intelligence & Data Science

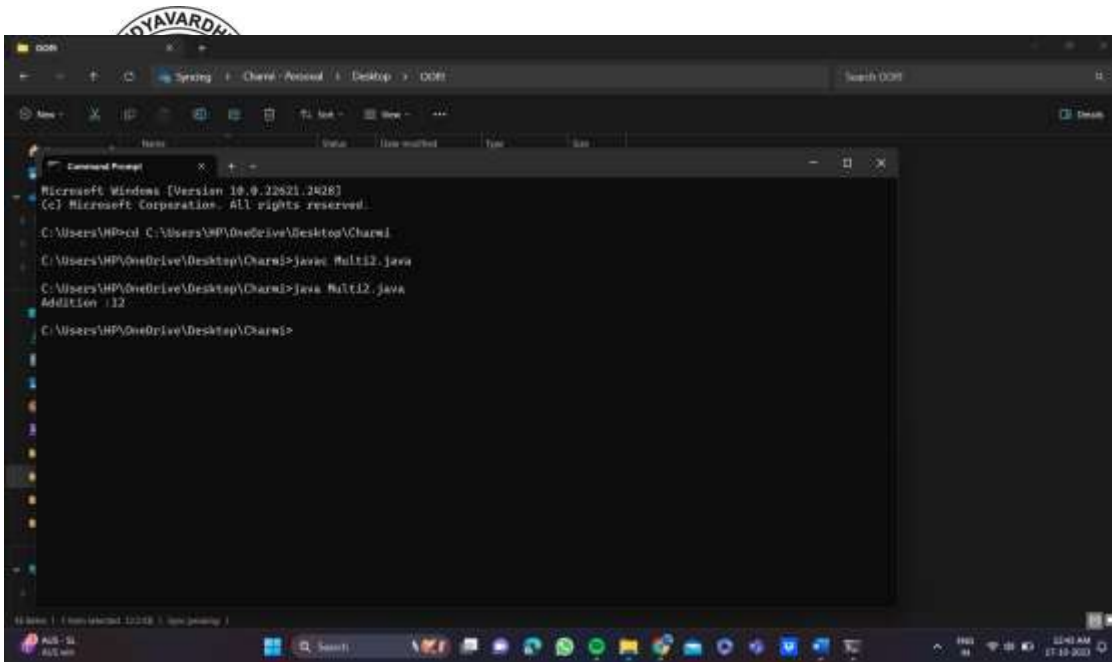
```
class Multi3 implements Runnable{  
  
    public void run(){  
        System.out.println("thread is running...");  
    }  
  
    public static void main(String args[]){  
        Multi3 m1=new Multi3();  
        Thread t1 =new Thread(m1); // Using the constructor Thread(Runnable r)  
        t1.start();  
    }  
}
```

Output:

```
thread is running...
```

Code:

```
class Multi2 implements Runnable{  
    public void run()  
    {  
        int a=5;  
        int b=7;  
        int c=a+b;  
        System.out.println("Addition :"+c);  
    }  
  
    public static void main(String args[]){  
        Multi2 m1=new Multi2();  
        Thread t1=new Thread(m1);  
        t1.start();  
    }  
}
```

Conclusion:

Multithreading in Java is supported through the `Thread` class and the `Runnable` interface. You can create and manage threads by extending the `Thread` class or implementing the `Runnable` interface. Java provides thread synchronization, management, and various thread states to enable concurrent execution of tasks. It's a fundamental feature for efficient resource utilization, improved application responsiveness, and better performance in multi-tasking environments.

In Java, you can create a new thread by extending the `Thread` class and overriding its `run()` method. Alternatively, you can implement the `Runnable` interface and pass an instance of the implementing class to a new `Thread` object. The `Thread` class provides several methods for managing threads, such as starting, stopping, pausing, and resuming threads.

Java also provides thread synchronization mechanisms such as locks and semaphores to ensure that multiple threads can access shared resources safely. This is important to avoid race conditions and other concurrency issues.

In summary, multithreading is a fundamental feature of Java that enables concurrent execution of tasks. It's supported through the `Thread` class and the `Runnable` interface, and Java provides several mechanisms for managing threads and ensuring thread safety.



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Experiment No. 11
Implement a program on Applet or AWT Controls
Date of Performance:
Date of Submission:

Aim: Implement a program on Applet or AWT Controls

CSL304: Object Oriented Programming with Java



Objective: Vidyavardhini's College of Engineering and Technology

To develop application like Calculator, Games, Animation using AWT Controls.

Department of Artificial Intelligence & Data Science

Theory:

Java AWT (Abstract Window Toolkit) is an API to develop Graphical User Interface (GUI) or windows-based applications in Java.

Java AWT components are platform-dependent i.e. components are displayed according to the view of operating system.

AWT is heavy weight i.e. its components are using the resources of underlying operating system (OS).

The `java.awt` package provides classes for AWT API such as `TextField`, `Label`, `TextArea`, `RadioButton`, `CheckBox`, `Choice`, `List` etc.

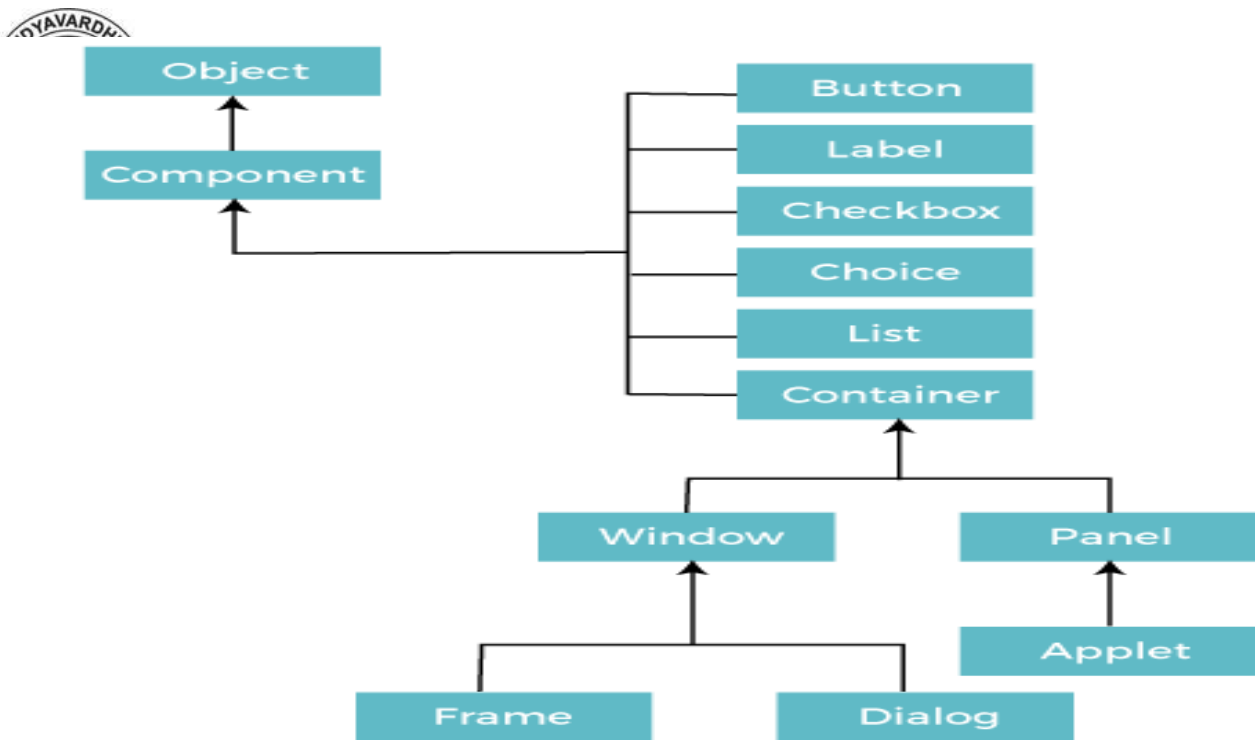
1. A general interface between Java and the native system, used for windowing, events and layout managers. This API is at the core of Java GUI programming and is also used by Swing and Java 2D. It contains the interface between the native windowing system and the Java application¹.
2. A basic set of GUI widgets such as buttons, text boxes, and menus¹. AWT also provides Graphics and imaging tools, such as shape, color, and font classes². AWT also avails layout managers which helps in increasing the flexibility of the window layouts².

Java AWT calls the native platform calls the native platform (operating systems) subroutine for creating API components like `TextField`, `ChechBox`, button, etc.

For example, an AWT GUI with components like `TextField`, label and button will have different look and feel for the different platforms like Windows, MAC OS, and Unix. The reason for this is the platforms have different view for their native components and AWT directly calls the native subroutine that creates those components.

In simple words, an AWT application will look like a windows application in Windows OS whereas it will look like a Mac application in the MAC OS.

Java AWT Hierarchy



Code:

```

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class GFG {
    public static void converter()
    {
        JFrame f = new JFrame("CONVERTER");
        JLabel l1, l2;
        JTextField t1, t2;
        JButton b1, b2, b3;
        l1 = new JLabel("Rupees:");
        l1.setBounds(20, 40, 60, 30);
        l2 = new JLabel("Dollars:");
        l2.setBounds(170, 40, 60, 30);
        t1 = new JTextField("0");
        t1.setBounds(80, 40, 50, 30);
        t2 = new JTextField("0");
        t2.setBounds(240, 40, 50, 30);
        b1 = new JButton("INR");
        b1.setBounds(50, 80, 60, 15);
        b2 = new JButton("Dollar");
        b2.setBounds(190, 80, 60, 15);
        b3 = new JButton("close");
        b3.setBounds(150, 150, 60, 30);
    }
}
  
```



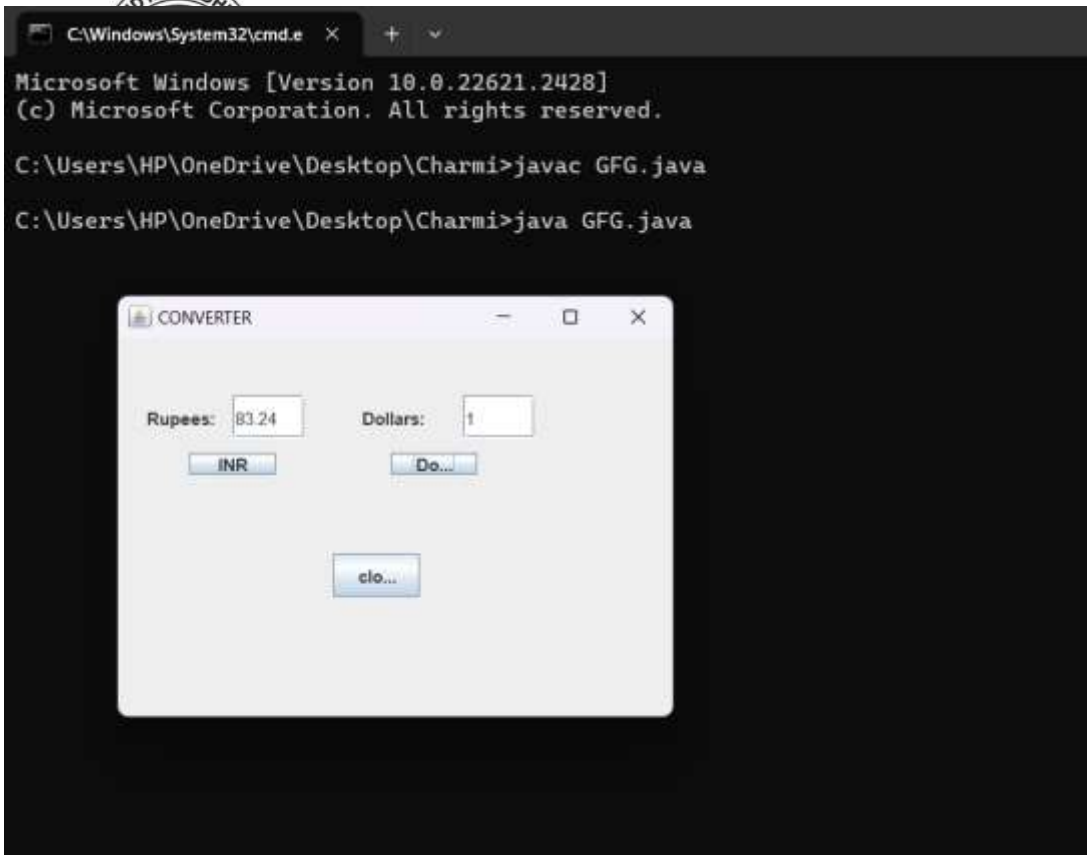
Vidya Vardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Department of Artificial Intelligence & Data Science

```
        double d
            = Double.parseDouble(t1.getText());
        double d1 = (d / 83.24);
        String str1 = String.valueOf(d1);
        t2.setText(str1);
    }
});
b2.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e)
    {
        double d2
            = Double.parseDouble(t2.getText());
        double d3 = (d2 * 83.24);
        String str2 = String.valueOf(d3);
        t1.setText(str2);
    }
});
b3.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e)
    {
        f.dispose();
    }
});
f.addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent e)
    {
        System.exit(0);
    }
});
f.add(l1);
f.add(t1);
f.add(l2);
f.add(t2);
f.add(b1);
f.add(b2);
f.add(b3);

f.setLayout(null);
f.setSize(400, 300);
f.setVisible(true);
}
public static void main(String args[])
{
    converter();
} }
```



Conclusion:

In Java, AWT (Abstract Window Toolkit) controls are used to create graphical user interfaces (GUIs) for desktop applications. AWT provides a set of basic GUI components, such as buttons, labels, text fields, and more. AWT also provides layout managers to arrange and position controls within your GUI. You can customize the appearance and behavior of AWT controls. AWT is platform-independent but may not provide the most modern look and feel. AWT allows you to create top-level containers (e.g., `Frame`) as the main windows for your application.



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Experiment No. 12
Course Project based on the content of the syllabus.
Date of Performance:
Date of Submission:



Vidyavardhini's College of Engineering and Technology
Department of Artificial Intelligence & Data Science

Report On

Scientific Calculator

Submitted in partial fulfillment of the requirements of the Course project in
Semester III of Second Year Artificial Intelligence and Data Science

by

Aryan Kantilal Gaikwad (Roll No. 09)

Aaryan Chandrakant Gole (Roll No. 12)

Sunit Sunil Khaire (Roll No. 21)

Supervisor

Prof. Sneha M. Yadav



University of Mumbai

Vidyavardhini's College of Engineering & Technology

Department of Artificial Intelligence and Data Science





Vidyavardhini's College of Engineering and Technology
Department of Artificial Intelligence & Data Science

(2023-24)

Vidyavardhini's College of Engineering & Technology
Department of Artificial Intelligence and Data Science



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

CERTIFICATE

This is to certify that the project entitled "Scientific Calculator" is a bonafide work of " Aaryan Gole (Roll No. 12), Aryan Gaikwad (Roll No. 09), Sunit Khair (Roll No. 21) " submitted to the University of Mumbai in partial fulfillment of the requirement for the Course project in semester III of Second Year Artificial Intelligence and Data Science engineering.

Supervisor

Prof. Sneha Yadav

Dr. Tatwadarshi P. N.
Head of Department



Table of Contents

Chapter No	Title	Page No.
1	Abstract	4
2	Problem Statement	4
3	Module Description	5
4	Brief Description of Software and Hardware Used	5
5	Code	6
6	Results and conclusion	40



Abstract

This abstract provides an overview of a course project aimed at developing a scientific calculator using Java programming. The project involves the creation of a comprehensive and user-friendly scientific calculator that incorporates a wide range of mathematical functions and features, suitable for students, scientists, and professionals alike.

The project begins with the design and architecture of the calculator, which includes the selection of appropriate data structures and algorithms for efficient computation. The user interface is also carefully designed to ensure an intuitive and visually appealing experience. The calculator features a user-friendly graphical interface, which allows users to input mathematical expressions using familiar notations and symbols.

The calculator supports a variety of mathematical operations, including basic arithmetic, advanced functions like trigonometric, logarithmic, and exponential calculations, as well as the ability to work with complex numbers. Error handling is implemented to ensure that invalid inputs or operations are appropriately managed, providing a robust and reliable tool for users.

In addition to its mathematical capabilities, the calculator also includes features such as history tracking, allowing users to review and reuse previous calculations. Unit conversion functionalities are integrated, enabling easy conversions between different measurement units. Furthermore, the project includes the incorporation of a graphical representation of functions, such as graphs and plots, enhancing the calculator's versatility for scientific and engineering applications.

The project leverages object-oriented programming principles to create a modular and extensible codebase, making it easy to add new functions and features in the future. Proper documentation is provided to ensure that the codebase is well-understood and can be maintained and extended by other developers.

This scientific calculator project in Java demonstrates the application of programming and software engineering concepts learned in the course, offering a practical and real-world coding experience. It serves as a valuable tool for students to deepen their understanding of Java, object-oriented design, and graphical user interface development while producing a sophisticated scientific calculator as a tangible outcome.

Problem Statement

Develop a Java-based Scientific Calculator using the AWT library including various modules like Swing, AWT, etc.



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Module Description:

1. User Interface Module:

- This module is a crucial component of a scientific calculator application, as it directly interfaces with the user and enables them to effectively utilize the calculator's mathematical capabilities.
- It provides the buttons and controls necessary for the user to interact with the calculator. These may include numerical keys (0-9), mathematical operators (+, -, *, /, etc.), functions (sin, cos, log, sqrt, etc.), and special keys (clear, delete, equals, etc.).
- For a scientific calculator, this module should include a comprehensive set of mathematical functions and operations, including but not limited to trigonometric functions, logarithmic functions, exponential functions, and more.

2. Input Handling Module:

- This module manages the user's input, which can include numerical values, mathematical operators, functions, and special keys (e.g., clear, delete, equals).

Brief Description of Software and Hardware Used

Software Used:

1. Java Development Kit (JDK) The calculator is developed using Java, and the JDK provides the necessary tools for Java programming, including the Java compiler and runtime environment.
2. Integrated Development Environment (IDE): An IDE like Eclipse, IntelliJ IDEA, or NetBeans can be used to write, debug, and run the Java code. IDEs provide features like code editing, debugging tools, and project management.



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

3. AWT (Abstract Window Toolkit): AWT is a Java library used to create graphical user interfaces. It provides classes for display screen, handling events, and computing functions while calculating them on the screen.



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Hardware:

1. Computer: A standard computer system with sufficient processing power and memory to run the Java development environment and execute the game. The hardware requirements are typically minimal for Java-based applications.
2. Mouse: Used for clicking buttons of the calculator for various functions calculations/computational operations.

CODE:

```
/*  
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license  
 * Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JFrame.java to edit this template  
 */  
package javaapplication1;  
  
public class Scialc2 extends javax.swing.JFrame {  
  
    /**  
     * Creates new form Scialc2  
     */  
    public Scialc2() {  
        initComponents();  
    }  
  
    double num1,num2,result;
```



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

String opr;

public void enable()



```
jRadioButton1.setEnabled(false);
```

```
jRadioButton2.setEnabled(true);
```

```
jButton1.setEnabled(true);
```

```
jButton2.setEnabled(true);
```

```
jButton3.setEnabled(true);
```

```
jButton4.setEnabled(true);
```

```
jButton5.setEnabled(true);
```

```
jButton6.setEnabled(true);
```

```
jButton7.setEnabled(true);
```

```
jButton8.setEnabled(true);
```

```
jButton9.setEnabled(true);
```

```
jButton10.setEnabled(true);
```

```
jButton11.setEnabled(true);
```

```
jButton12.setEnabled(true);
```

```
jButton13.setEnabled(true);
```

```
jButton14.setEnabled(true);
```

```
jButton15.setEnabled(true);
```

```
jButton16.setEnabled(true);
```

```
jButton17.setEnabled(true);
```

```
jButton18.setEnabled(true);
```

```
jButton20.setEnabled(true);
```

```
jButton22.setEnabled(true);
```

```
jButton23.setEnabled(true);
```

```
jButton24.setEnabled(true);
```

```
jButton25.setEnabled(true);
```

```
jButton26.setEnabled(true);
```

```
jButton27.setEnabled(true);
```

```
jButton28.setEnabled(true);
```

```
jButton29.setEnabled(true);
```



jButton30.setEnabled(true);

jButton31.setEnabled(true);

jButton32.setEnabled(true);

jButton33.setEnabled(true);

jButton34.setEnabled(true);

jButton36.setEnabled(true);

jButton37.setEnabled(true);

}

public void disable()

{

jTextField1.setEnabled(false);

jRadioButton1.setEnabled(true);

jRadioButton2.setEnabled(false);

jButton1.setEnabled(false);

jButton2.setEnabled(false);

jButton3.setEnabled(false);

jButton4.setEnabled(false);

jButton5.setEnabled(false);

jButton6.setEnabled(false);

jButton7.setEnabled(false);

jButton8.setEnabled(false);

jButton9.setEnabled(false);

jButton10.setEnabled(false);

jButton11.setEnabled(false);

jButton12.setEnabled(false);

jButton13.setEnabled(false);

jButton14.setEnabled(false);

jButton15.setEnabled(false);

jButton16.setEnabled(false);



```
jButton17.setEnabled(false);
jButton18.setEnabled(false);
jButton20.setEnabled(false);
jButton22.setEnabled(false);
jButton23.setEnabled(false);
jButton24.setEnabled(false);
jButton25.setEnabled(false);
jButton26.setEnabled(false);
jButton27.setEnabled(false);
jButton28.setEnabled(false);
jButton29.setEnabled(false);
jButton30.setEnabled(false);
jButton31.setEnabled(false);
jButton32.setEnabled(false);
jButton33.setEnabled(false);
jButton34.setEnabled(false);
jButton36.setEnabled(false);
jButton37.setEnabled(false);

}

/**
 * This method is called from within the constructor to initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is always
 * regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    buttonGroup1 = new javax.swing.ButtonGroup();
    jTextField1 = new javax.swing.JTextField();
    jRadioButton1 = new javax.swing.JRadioButton();
```



```
jRadioButton2 = new javax.swing.JRadioButton();
```

```
jButton1 = new javax.swing.JButton();
```

```
jButton2 = new javax.swing.JButton();
```

```
jButton3 = new javax.swing.JButton();
```

```
jButton4 = new javax.swing.JButton();
```

```
jButton5 = new javax.swing.JButton();
```

```
jButton6 = new javax.swing.JButton();
```

```
jButton7 = new javax.swing.JButton();
```

```
jButton8 = new javax.swing.JButton();
```

```
jButton9 = new javax.swing.JButton();
```

```
jButton10 = new javax.swing.JButton();
```

```
jButton11 = new javax.swing.JButton();
```

```
jButton12 = new javax.swing.JButton();
```

```
jButton13 = new javax.swing.JButton();
```

```
jButton14 = new javax.swing.JButton();
```

```
jButton15 = new javax.swing.JButton();
```

```
jButton16 = new javax.swing.JButton();
```

```
jButton17 = new javax.swing.JButton();
```

```
jButton18 = new javax.swing.JButton();
```

```
jButton20 = new javax.swing.JButton();
```

```
jButton22 = new javax.swing.JButton();
```

```
jButton23 = new javax.swing.JButton();
```

```
jButton24 = new javax.swing.JButton();
```

```
jButton25 = new javax.swing.JButton();
```

```
jButton26 = new javax.swing.JButton();
```

```
jButton27 = new javax.swing.JButton();
```

```
jButton28 = new javax.swing.JButton();
```

```
jButton29 = new javax.swing.JButton();
```

```
jButton30 = new javax.swing.JButton();
```

```
jButton31 = new javax.swing.JButton();
```

```
jButton32 = new javax.swing.JButton();
```

```
jButton33 = new javax.swing.JButton();
```

```
jButton34 = new javax.swing.JButton();
```



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

```
setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);  
setTitle("Scientific Calculator");  
setResizable(false);  
setType(java.awt.Window.Type.UTILITY);
```

```
jTextField1.setFont(new java.awt.Font("Segoe UI", 1, 24)); // NOI18N  
jTextField1.setHorizontalAlignment(javax.swing.JTextField.RIGHT);  
jTextField1.addActionListener(new java.awt.event.ActionListener() {  
    public void actionPerformed(java.awt.event.ActionEvent evt) {  
        jTextField1ActionPerformed(evt);  
    }  
});
```

```
jRadioButton1.setBackground(new java.awt.Color(255, 255, 255));  
buttonGroup1.add(jRadioButton1);  
jRadioButton1.setText("on");  
jRadioButton1.addActionListener(new java.awt.event.ActionListener() {  
    public void actionPerformed(java.awt.event.ActionEvent evt) {  
        jRadioButton1ActionPerformed(evt);  
    }  
});
```

```
jRadioButton2.setBackground(new java.awt.Color(255, 255, 255));  
buttonGroup1.add(jRadioButton2);  
jRadioButton2.setText("off");  
jRadioButton2.addActionListener(new java.awt.event.ActionListener() {  
    public void actionPerformed(java.awt.event.ActionEvent evt) {  
        jRadioButton2ActionPerformed(evt);  
    }  
});
```



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

```
jButton1.setBackground(new java.awt.Color(102, 102, 102));  
jButton1.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N
```

```
jButton1.setText("x^y");  
jButton1.addActionListener(new java.awt.event.ActionListener() {  
    public void actionPerformed(java.awt.event.ActionEvent evt) {  
        jButton1ActionPerformed(evt);  
    }  
});
```

```
jButton2.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N  
jButton2.setText("7");  
jButton2.addActionListener(new java.awt.event.ActionListener() {  
    public void actionPerformed(java.awt.event.ActionEvent evt) {  
        jButton2ActionPerformed(evt);  
    }  
});
```

```
jButton3.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N  
jButton3.setText("9");  
jButton3.addActionListener(new java.awt.event.ActionListener() {  
    public void actionPerformed(java.awt.event.ActionEvent evt) {  
        jButton3ActionPerformed(evt);  
    }  
});
```

```
jButton4.setBackground(new java.awt.Color(102, 102, 102));  
jButton4.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N  
jButton4.setText("1/x");  
jButton4.addActionListener(new java.awt.event.ActionListener() {  
    public void actionPerformed(java.awt.event.ActionEvent evt) {  
        jButton4ActionPerformed(evt);  
    }  
});
```



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

```
jButton5.setBackground(new java.awt.Color(102, 102, 102));
jButton5.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N
jButton5.setForeground(new java.awt.Color(255, 255, 255));
jButton5.setText("%");
jButton5.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton5ActionPerformed(evt);
    }
});

jButton6.setBackground(new java.awt.Color(102, 102, 102));
jButton6.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N
jButton6.setForeground(new java.awt.Color(255, 255, 255));
jButton6.setText("sinh");
jButton6.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton6ActionPerformed(evt);
    }
});

jButton7.setBackground(new java.awt.Color(102, 102, 102));
jButton7.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N
jButton7.setText("\n");
jButton7.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton7ActionPerformed(evt);
    }
});

jButton8.setBackground(new java.awt.Color(102, 102, 102));
jButton8.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N
```



```
jButton8.setForeground(new java.awt.Color(255, 255, 255));
jButton8.setText("log");
jButton8.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton8ActionPerformed(evt);
    }
});
```

```
jButton9.setBackground(new java.awt.Color(255, 0, 51));
jButton9.setFont(new java.awt.Font("Segoe UI", 1, 18)); // NOI18N
jButton9.setForeground(new java.awt.Color(255, 255, 255));
jButton9.setText("C");
jButton9.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton9ActionPerformed(evt);
    }
});
```

```
jButton10.setBackground(new java.awt.Color(102, 102, 102));
jButton10.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N
jButton10.setText("x3");
jButton10.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton10ActionPerformed(evt);
    }
});
```

```
jButton11.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N
jButton11.setText("8");
jButton11.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton11ActionPerformed(evt);
    }
});
```




Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

```
jButton12.setFont(new java.awt.Font("Segoe UI", 1, 24)); // NOI18N
```

```
jButton12.setForeground(new java.awt.Color(0, 204, 102));
```

```
jButton12.setText("-");
```

```
jButton12.setMaximumSize(new java.awt.Dimension(26, 26));
```

```
jButton12.addActionListener(new java.awt.event.ActionListener() {
```

```
    public void actionPerformed(java.awt.event.ActionEvent evt) {
```

```
        jButton12ActionPerformed(evt);
```

```
    }
```

```
});
```

```
jButton13.setFont(new java.awt.Font("Segoe UI", 1, 24)); // NOI18N
```

```
jButton13.setForeground(new java.awt.Color(204, 51, 0));
```

```
jButton13.setText("←");
```

```
jButton13.setMaximumSize(new java.awt.Dimension(26, 26));
```

```
jButton13.addActionListener(new java.awt.event.ActionListener() {
```

```
    public void actionPerformed(java.awt.event.ActionEvent evt) {
```

```
        jButton13ActionPerformed(evt);
```

```
    }
```

```
});
```

```
jButton14.setBackground(new java.awt.Color(102, 102, 102));
```

```
jButton14.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N
```

```
jButton14.setForeground(new java.awt.Color(255, 255, 255));
```

```
jButton14.setText("cosh");
```

```
jButton14.addActionListener(new java.awt.event.ActionListener() {
```

```
    public void actionPerformed(java.awt.event.ActionEvent evt) {
```

```
        jButton14ActionPerformed(evt);
```

```
    }
```

```
});
```

```
jButton15.setBackground(new java.awt.Color(102, 102, 102));
```



```
jButton15.setFont(new java.awt.Font("Segoe UI", 1, 12));  
jButton15.setForeground(new java.awt.Color(255, 255, 255));  
jButton15.setText("cos");
```

```
jButton15.addActionListener(new java.awt.event.ActionListener() {  
    public void actionPerformed(java.awt.event.ActionEvent evt) {  
        jButton15ActionPerformed(evt);  
    }  
});
```

```
jButton16.setBackground(new java.awt.Color(102, 102, 102));  
jButton16.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N  
jButton16.setForeground(new java.awt.Color(255, 255, 255));  
jButton16.setText("exp");  
jButton16.addActionListener(new java.awt.event.ActionListener() {  
    public void actionPerformed(java.awt.event.ActionEvent evt) {  
        jButton16ActionPerformed(evt);  
    }  
});
```

```
jButton17.setBackground(new java.awt.Color(102, 102, 102));  
jButton17.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N  
jButton17.setForeground(new java.awt.Color(255, 255, 255));  
jButton17.setText("sin");  
jButton17.addActionListener(new java.awt.event.ActionListener() {  
    public void actionPerformed(java.awt.event.ActionEvent evt) {  
        jButton17ActionPerformed(evt);  
    }  
});
```

```
jButton18.setBackground(new java.awt.Color(102, 102, 102));  
jButton18.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N  
jButton18.setForeground(new java.awt.Color(255, 255, 255));  
jButton18.setText("tanh");
```



```
jButton18.addActionListener(new java.awt.event.ActionListener() {  
    public void actionPerformed(java.awt.event.ActionEvent evt) {  
        jButton18ActionPerformed(evt);  
    }  
});
```

```
jButton20.setBackground(new java.awt.Color(102, 102, 102));  
jButton20.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N  
jButton20.setForeground(new java.awt.Color(255, 255, 255));  
jButton20.setText("tan");  
jButton20.addActionListener(new java.awt.event.ActionListener() {  
    public void actionPerformed(java.awt.event.ActionEvent evt) {  
        jButton20ActionPerformed(evt);  
    }  
});
```

```
jButton22.setFont(new java.awt.Font("Segoe UI", 1, 18)); // NOI18N  
jButton22.setForeground(new java.awt.Color(0, 204, 102));  
jButton22.setText("+");  
jButton22.addActionListener(new java.awt.event.ActionListener() {  
    public void actionPerformed(java.awt.event.ActionEvent evt) {  
        jButton22ActionPerformed(evt);  
    }  
});
```

```
jButton23.setBackground(new java.awt.Color(102, 102, 102));  
jButton23.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N  
jButton23.setText("x²");  
jButton23.addActionListener(new java.awt.event.ActionListener() {  
    public void actionPerformed(java.awt.event.ActionEvent evt) {  
        jButton23ActionPerformed(evt);  
    }  
});
```



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

```
jButton24.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N  
jButton24.setText("5");
```

```
jButton24.addActionListener(new java.awt.event.ActionListener() {  
    public void actionPerformed(java.awt.event.ActionEvent evt) {  
        jButton24ActionPerformed(evt);  
    }  
});
```

```
jButton25.setFont(new java.awt.Font("Segoe UI", 1, 18)); // NOI18N  
jButton25.setForeground(new java.awt.Color(0, 204, 102));
```

```
jButton25.setText("*");  
jButton25.addActionListener(new java.awt.event.ActionListener() {  
    public void actionPerformed(java.awt.event.ActionEvent evt) {  
        jButton25ActionPerformed(evt);  
    }  
});
```

```
jButton26.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N  
jButton26.setText("4");
```

```
jButton26.addActionListener(new java.awt.event.ActionListener() {  
    public void actionPerformed(java.awt.event.ActionEvent evt) {  
        jButton26ActionPerformed(evt);  
    }  
});
```

```
jButton27.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N  
jButton27.setText("6");
```

```
jButton27.addActionListener(new java.awt.event.ActionListener() {  
    public void actionPerformed(java.awt.event.ActionEvent evt) {  
        jButton27ActionPerformed(evt);  
    }  
});
```



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

```
jButton28.setBackground(new java.awt.Color(102, 102, 102));  
jButton28.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N
```

```
jButton28.setText("n!");  
jButton28.addActionListener(new java.awt.event.ActionListener() {  
    public void actionPerformed(java.awt.event.ActionEvent evt) {  
        jButton28ActionPerformed(evt);  
    }  
});
```

```
jButton29.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N  
jButton29.setText("2");  
jButton29.addActionListener(new java.awt.event.ActionListener() {  
    public void actionPerformed(java.awt.event.ActionEvent evt) {  
        jButton29ActionPerformed(evt);  
    }  
});
```

```
jButton30.setFont(new java.awt.Font("Segoe UI", 1, 18)); // NOI18N  
jButton30.setForeground(new java.awt.Color(0, 204, 51));  
jButton30.setText("/");  
jButton30.addActionListener(new java.awt.event.ActionListener() {  
    public void actionPerformed(java.awt.event.ActionEvent evt) {  
        jButton30ActionPerformed(evt);  
    }  
});
```

```
jButton31.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N  
jButton31.setText("1");  
jButton31.addActionListener(new java.awt.event.ActionListener() {  
    public void actionPerformed(java.awt.event.ActionEvent evt) {  
        jButton31ActionPerformed(evt);  
    }  
});
```



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

```
jButton32.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N
```

```
jButton32.setText("3");
```

```
jButton32.addActionListener(new java.awt.event.ActionListener() {  
    public void actionPerformed(java.awt.event.ActionEvent evt) {  
        jButton32ActionPerformed(evt);  
    }  
});
```

```
jButton33.setBackground(new java.awt.Color(102, 102, 102));
```

```
jButton33.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N
```

```
jButton33.setText("+/-");
```

```
jButton33.addActionListener(new java.awt.event.ActionListener() {  
    public void actionPerformed(java.awt.event.ActionEvent evt) {  
        jButton33ActionPerformed(evt);  
    }  
});
```

```
jButton34.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N
```

```
jButton34.setText(".");
```

```
jButton34.addActionListener(new java.awt.event.ActionListener() {  
    public void actionPerformed(java.awt.event.ActionEvent evt) {  
        jButton34ActionPerformed(evt);  
    }  
});
```

```
jButton36.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N
```

```
jButton36.setText("0");
```

```
jButton36.addActionListener(new java.awt.event.ActionListener() {  
    public void actionPerformed(java.awt.event.ActionEvent evt) {  
        jButton36ActionPerformed(evt);  
    }  
});
```




```
.addComponent(jButton10, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.PREFERRED_SIZE, 58, javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(jButton23, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.PREFERRED_SIZE, 58, javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(jButton28, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.PREFERRED_SIZE, 58, javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(jButton33, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.PREFERRED_SIZE, 58, javax.swing.GroupLayout.PREFERRED_SIZE))
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addComponent(jButton5, javax.swing.GroupLayout.PREFERRED_SIZE, 58,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(jButton2, javax.swing.GroupLayout.PREFERRED_SIZE, 58,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(jButton26, javax.swing.GroupLayout.PREFERRED_SIZE, 58,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(jButton31, javax.swing.GroupLayout.PREFERRED_SIZE, 58,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(jButton36, javax.swing.GroupLayout.PREFERRED_SIZE, 58,
javax.swing.GroupLayout.PREFERRED_SIZE))
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
.addComponent(jButton9, javax.swing.GroupLayout.PREFERRED_SIZE, 58,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED))
```




```
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
        .addComponent(jButton22, javax.swing.GroupLayout.PREFERRED_SIZE,
58, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGroup(layout.createSequentialGroup())
        .addComponent(jButton11, javax.swing.GroupLayout.PREFERRED_SIZE,
58, javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
        .addComponent(jButton3, javax.swing.GroupLayout.PREFERRED_SIZE, 58,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
        .addComponent(jButton12, javax.swing.GroupLayout.PREFERRED_SIZE,
58, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGroup(layout.createSequentialGroup())
        .addComponent(jButton24, javax.swing.GroupLayout.PREFERRED_SIZE,
58, javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
        .addComponent(jButton27, javax.swing.GroupLayout.PREFERRED_SIZE,
58, javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
        .addComponent(jButton25, javax.swing.GroupLayout.PREFERRED_SIZE,
58, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGroup(layout.createSequentialGroup())
        .addComponent(jButton29, javax.swing.GroupLayout.PREFERRED_SIZE,
58, javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
```



```
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
    .addComponent(jButton30,    javax.swing.GroupLayout.PREFERRED_SIZE,
58, javax.swing.GroupLayout.PREFERRED_SIZE))
    .addGroup(layout.createSequentialGroup())
    .addComponent(jButton34,    javax.swing.GroupLayout.PREFERRED_SIZE,
58, javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
    .addComponent(jButton37,    javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))))
    .addGroup(layout.createSequentialGroup())
    .addComponent(jButton16,    javax.swing.GroupLayout.PREFERRED_SIZE,    58,
javax.swing.GroupLayout.PREFERRED_SIZE)
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
    .addComponent(jButton17,    javax.swing.GroupLayout.PREFERRED_SIZE,    58,
javax.swing.GroupLayout.PREFERRED_SIZE)
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
    .addComponent(jButton15,    javax.swing.GroupLayout.PREFERRED_SIZE,    58,
javax.swing.GroupLayout.PREFERRED_SIZE)
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
    .addComponent(jButton20,    javax.swing.GroupLayout.PREFERRED_SIZE,    58,
javax.swing.GroupLayout.PREFERRED_SIZE))
    .addGroup(layout.createSequentialGroup())
    .addComponent(jButton8,    javax.swing.GroupLayout.PREFERRED_SIZE,    58,
javax.swing.GroupLayout.PREFERRED_SIZE)
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
    .addComponent(jButton6,    javax.swing.GroupLayout.PREFERRED_SIZE,    58,
javax.swing.GroupLayout.PREFERRED_SIZE)
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
```



```
.addComponent(jButton18, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
.addComponent(jButton18, javax.swing.GroupLayout.PREFERRED_SIZE, 58,
javax.swing.GroupLayout.PREFERRED_SIZE))))
.addGroup(javax.swing.GroupLayout.Alignment.LEADING,
layout.createSequentialGroup()
.addComponent(jRadioButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 50,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addComponent(jRadioButton2, javax.swing.GroupLayout.PREFERRED_SIZE, 56,
javax.swing.GroupLayout.PREFERRED_SIZE))
.addComponent(jTextField1, javax.swing.GroupLayout.Alignment.LEADING))
.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
);
layout.setVerticalGroup(
layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(layout.createSequentialGroup()
.addGap(16, 16, 16)
.addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED_SIZE, 69,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
.addComponent(jRadioButton1)
.addComponent(jRadioButton2))
.addGap(12, 12, 12)
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
.addComponent(jButton7, javax.swing.GroupLayout.PREFERRED_SIZE, 48,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(jButton16, javax.swing.GroupLayout.PREFERRED_SIZE, 48,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(jButton17, javax.swing.GroupLayout.PREFERRED_SIZE, 48,
javax.swing.GroupLayout.PREFERRED_SIZE))
```



```
.addComponent(jButton10, javax.swing.GroupLayout.PREFERRED_SIZE, 48,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(jButton20, javax.swing.GroupLayout.PREFERRED_SIZE, 48,
javax.swing.GroupLayout.PREFERRED_SIZE))
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
.addComponent(jButton4, javax.swing.GroupLayout.PREFERRED_SIZE, 48,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(jButton8, javax.swing.GroupLayout.PREFERRED_SIZE, 48,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(jButton6, javax.swing.GroupLayout.PREFERRED_SIZE, 48,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(jButton14, javax.swing.GroupLayout.PREFERRED_SIZE, 48,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(jButton18, javax.swing.GroupLayout.PREFERRED_SIZE, 48,
javax.swing.GroupLayout.PREFERRED_SIZE))
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
.addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 48,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(jButton5, javax.swing.GroupLayout.PREFERRED_SIZE, 48,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(jButton9, javax.swing.GroupLayout.PREFERRED_SIZE, 48,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(jButton13, javax.swing.GroupLayout.PREFERRED_SIZE, 48,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(jButton22, javax.swing.GroupLayout.PREFERRED_SIZE, 48,
javax.swing.GroupLayout.PREFERRED_SIZE))
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
.addComponent(jButton10, javax.swing.GroupLayout.PREFERRED_SIZE, 48,
javax.swing.GroupLayout.PREFERRED_SIZE))
```



```
.addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE) 48,
.addComponent(jButton11, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(jButton3, javax.swing.GroupLayout.PREFERRED_SIZE, 48,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(jButton12, javax.swing.GroupLayout.PREFERRED_SIZE, 48,
javax.swing.GroupLayout.PREFERRED_SIZE))
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
.addComponent(jButton23, javax.swing.GroupLayout.PREFERRED_SIZE, 48,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(jButton26, javax.swing.GroupLayout.PREFERRED_SIZE, 48,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(jButton24, javax.swing.GroupLayout.PREFERRED_SIZE, 48,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(jButton27, javax.swing.GroupLayout.PREFERRED_SIZE, 48,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(jButton25, javax.swing.GroupLayout.PREFERRED_SIZE, 48,
javax.swing.GroupLayout.PREFERRED_SIZE))
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
.addComponent(jButton28, javax.swing.GroupLayout.PREFERRED_SIZE, 48,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(jButton31, javax.swing.GroupLayout.PREFERRED_SIZE, 48,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(jButton29, javax.swing.GroupLayout.PREFERRED_SIZE, 48,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(jButton32, javax.swing.GroupLayout.PREFERRED_SIZE, 48,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(jButton30, javax.swing.GroupLayout.PREFERRED_SIZE, 48,
javax.swing.GroupLayout.PREFERRED_SIZE))
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
```




Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

```
private void jButton10ActionPerformed(java.awt.event.ActionEvent evt) {
```

```
double t=Double.parseDouble(jTextField1.getText());
```

```
t=t*t*t;
```

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

```
jTextField1.setText(jTextField1.getText()+t);
```

```
}
```

```
private void jButton6ActionPerformed(java.awt.event.ActionEvent evt) {
```

```
double t=Math.sinh(Double.parseDouble(jTextField1.getText()));
```

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

```
jTextField1.setText(jTextField1.getText()+t);
```

```
}
```

```
private void jButton23ActionPerformed(java.awt.event.ActionEvent evt) {
```

```
double t=Double.parseDouble(jTextField1.getText());
```

```
t=t*t;
```

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

```
jTextField1.setText(jTextField1.getText()+t);
```

```
}
```

```
private void jButton28ActionPerformed(java.awt.event.ActionEvent evt) {
```

```
double t=Double.parseDouble(jTextField1.getText());
```

```
int a = 1;
```

```
long fact = 1;
```

```
while(a <= t)
```

```
{
```

```
fact*= t;
```

```
t--;
```



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

```
    jTextField1.setText("");
    jTextField1.setText(jTextField1.getText()+fact);
}

private void jButton33ActionPerformed(java.awt.event.ActionEvent evt) {
    double t=Double.parseDouble(String.valueOf(jTextField1.getText()));

    t=t*(-1);
    jTextField1.setText(String.valueOf(t));
}

private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
    double t=1/(Double.parseDouble(jTextField1.getText()));

    jTextField1.setText("");
    jTextField1.setText(jTextField1.getText()+t);
}

private void jButton29ActionPerformed(java.awt.event.ActionEvent evt) {
    jTextField1.setText(jTextField1.getText()+"2");
}

private void jButton24ActionPerformed(java.awt.event.ActionEvent evt) {
    jTextField1.setText(jTextField1.getText()+"5");
}

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    jTextField1.setText(jTextField1.getText()+"9");
}

private void jButton36ActionPerformed(java.awt.event.ActionEvent evt) {
```




```
jTextField1.setText(jTextField1.getText()+"1");  
}
```

```
private void jButton32ActionPerformed(java.awt.event.ActionEvent evt) {  
    jTextField1.setText(jTextField1.getText()+"3");  
}
```

```
private void jButton27ActionPerformed(java.awt.event.ActionEvent evt) {  
    jTextField1.setText(jTextField1.getText()+"6");  
}
```

```
private void jButton26ActionPerformed(java.awt.event.ActionEvent evt) {  
    jTextField1.setText(jTextField1.getText()+"4");  
}
```

```
private void jButton11ActionPerformed(java.awt.event.ActionEvent evt) {  
    jTextField1.setText(jTextField1.getText()+"8");  
}
```

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {  
    jTextField1.setText(jTextField1.getText()+"7");  
}
```

```
private void jButton34ActionPerformed(java.awt.event.ActionEvent evt) {  
    jTextField1.setText(jTextField1.getText()+".");  
}
```

```
private void jTextField1ActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:
```



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

```
private void jButton13ActionPerformed(java.awt.event.ActionEvent evt) {
```

```
String backSpace=null;
```

```
if(jTextField1.getText().length()>0)
```

```
{
```

```
    StringBuilder s=new StringBuilder(jTextField1.getText());
```

```
    s.deleteCharAt(jTextField1.getText().length()-1);
```

```
    backSpace=s.toString();
```

```
    jTextField1.setText(backSpace);
```

```
}
```

```
}
```

```
private void jButton22ActionPerformed(java.awt.event.ActionEvent evt) {
```

```
    num1=Double.parseDouble(jTextField1.getText());
```

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

```
    opr="+";
```

```
}
```

```
private void jButton12ActionPerformed(java.awt.event.ActionEvent evt) {
```

```
    num1=Double.parseDouble(jTextField1.getText());
```

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

```
    opr="-";
```

```
}
```

```
private void jButton25ActionPerformed(java.awt.event.ActionEvent evt) {
```

```
    num1=Double.parseDouble(jTextField1.getText());
```



```
opr="*";  
}  
  
private void jButton30ActionPerformed(java.awt.event.ActionEvent evt) {  
    num1=Double.parseDouble(jTextField1.getText());  
  
    jTextField1.setText("");  
  
    opr="/";  
}  
  
private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {  
    num1=Double.parseDouble(jTextField1.getText());  
  
    jTextField1.setText("");  
  
    opr="%";  
}  
  
private void jButton37ActionPerformed(java.awt.event.ActionEvent evt) {  
    num2=Double.parseDouble(jTextField1.getText());  
  
    if(opr=="+")  
    {  
        result=num1+num2;  
        jTextField1.setText(Double.toString(result));  
    }  
  
    else if(opr=="-")  
    {
```



```
}  
  
else if(opr=="*")  
{  
    result=num1*num2;  
    jTextField1.setText(Double.toString(result));  
}  
  
else if(opr=="/")  
{  
    result=num1/num2;  
    jTextField1.setText(Double.toString(result));  
}  
  
else if(opr=="%")  
{  
    result=(num1/100)*num2;  
    jTextField1.setText(Double.toString(result));  
}  
  
else if(opr=="x^y")  
{  
    result = (int)Math.pow(num1,num2);  
  
    jTextField1.setText(Double.toString(result));  
}  
}  
  
private void jButton17ActionPerformed(java.awt.event.ActionEvent evt) {  
    double t=Math.sin(Double.parseDouble(jTextField1.getText()));
```



```
jTextField1.setText("");
jTextField1.setText(jTextField1.getText()+t);
}

private void jButton20ActionPerformed(java.awt.event.ActionEvent evt) {
    double t=Math.tan(Double.parseDouble(jTextField1.getText()));

    jTextField1.setText("");
    jTextField1.setText(jTextField1.getText()+t);
}

private void jButton14ActionPerformed(java.awt.event.ActionEvent evt) {
    double t=Math.cosh(Double.parseDouble(jTextField1.getText()));

    jTextField1.setText("");
    jTextField1.setText(jTextField1.getText()+t);
}

private void jButton18ActionPerformed(java.awt.event.ActionEvent evt) {
    double t=Math.tan(Double.parseDouble(jTextField1.getText()));

    jTextField1.setText("");
    jTextField1.setText(jTextField1.getText()+t);
}

private void jButton16ActionPerformed(java.awt.event.ActionEvent evt) {
    double t=Math.exp(Double.parseDouble(jTextField1.getText()));

    jTextField1.setText("");
    jTextField1.setText(jTextField1.getText()+t);
}

private void jButton7ActionPerformed(java.awt.event.ActionEvent evt) {
```



```
double t=Math.log(Double.parseDouble(jTextField1.getText()));
jTextField1.setText("");
jTextField1.setText(jTextField1.getText()+t);
}

private void jButton8ActionPerformed(java.awt.event.ActionEvent evt) {
    double t=Math.log(Double.parseDouble(jTextField1.getText()));

    jTextField1.setText("");
    jTextField1.setText(jTextField1.getText()+t);
}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    num1=Double.parseDouble(jTextField1.getText());

    jTextField1.setText("");

    opr="x^y";
}

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
    /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
     * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
     */
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
```



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

```
break;
    }
}
} catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(Scicalc2.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
    } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(Scicalc2.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
    } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(Scicalc2.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(Scicalc2.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
    }
//</editor-fold>

/* Create and display the form */
java.awt.EventQueue.invokeLater(new Runnable() {
    public void run() {
        new Scicalc2().setVisible(true);
    }
});
}

// Variables declaration - do not modify
```



```
private javax.swing.ButtonGroup buttonGroup1;  
private javax.swing.JButton jButton1;  
private javax.swing.JButton jButton10;
```

```
private javax.swing.JButton jButton11;
```

```
private javax.swing.JButton jButton12;
```

```
private javax.swing.JButton jButton13;
```

```
private javax.swing.JButton jButton14;
```

```
private javax.swing.JButton jButton15;
```

```
private javax.swing.JButton jButton16;
```

```
private javax.swing.JButton jButton17;
```

```
private javax.swing.JButton jButton18;
```

```
private javax.swing.JButton jButton2;
```

```
private javax.swing.JButton jButton20;
```

```
private javax.swing.JButton jButton22;
```

```
private javax.swing.JButton jButton23;
```

```
private javax.swing.JButton jButton24;
```

```
private javax.swing.JButton jButton25;
```

```
private javax.swing.JButton jButton26;
```

```
private javax.swing.JButton jButton27;
```

```
private javax.swing.JButton jButton28;
```

```
private javax.swing.JButton jButton29;
```

```
private javax.swing.JButton jButton3;
```

```
private javax.swing.JButton jButton30;
```

```
private javax.swing.JButton jButton31;
```

```
private javax.swing.JButton jButton32;
```

```
private javax.swing.JButton jButton33;
```

```
private javax.swing.JButton jButton34;
```

```
private javax.swing.JButton jButton36;
```

```
private javax.swing.JButton jButton37;
```

```
private javax.swing.JButton jButton4;
```

```
private javax.swing.JButton jButton5;
```

```
private javax.swing.JButton jButton6;
```

```
private javax.swing.JButton jButton7;
```




Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

```
private javax.swing.JButton jButton1;  
private javax.swing.JButton jButton9;  
private javax.swing.JRadioButton jButton1;  
private javax.swing.JRadioButton jButton2;  
private javax.swing.JTextField jTextField1;  
// End of variables declaration
```

```
private static class jButton19 {  
  
    public jButton19() {  
    }  
}  
}
```



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Results and Conclusion:



Figure 6.1



Figure 6.2

In conclusion, the development of the Java-based Scientific Calculator using the AWT library showcases the fundamental principles of event-driven programming, graphic rendering, and user interface design as shown in Figures 6.1 and 6.2. This project provides an engaging and interactive experience, allowing users to calculate within a visually appealing display. The use of Java, Swing and AWT enables a responsive and interesting experience. Further enhancements, such as degree to radian conversions,



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Simultaneous Polynomial Equations solving, etc. can be integrated to make this project a foundation for more complex Scientific Calculator development. Overall, this project illustrates the versatility and potential of Java in creating interactive graphical applications.