

Lab Report
of
OBJECT ORIENTED PROGRAMMING
(By Java)

5TH Semester of
Bachelor of Technology
(Computer Science & Engineering)



Session 2022-23

Submitted To:

Ms. Manisha Mudgal

Submitted By:

Avijit Dutta

CSE-20/020

Department of Computer Science
Satyug Darshan Institute of Engineering & Technology

INDEX

S.No	PROGRAMS	DATE	SIGN
1.	Introduction to Java.		
2.	Write a Program to Print HelloWorld.		
3.	Write a Program to add two numbers with out Scanner.		
4.	Write a program to perform all mathematical calculations without Scanner.		
5.	Write a program two sum two numbers by taking user input.		
6	Write a program to make menu using Switch case.		
7.	Write a Program to perform single inheritance that can solve a real life reusability problem.		
8.	Write a program showing multi level inheritance where each class at new features and inherit some features from the base class		
9	Write a program to perform multiple inheritance using interfaces		
10.	Write a program of abstract class		
11.	Write a program to perform exception handling using try and catch		
12.	Write a program to create an application using swing		

PROGRAM NO: 1

AIM: Introduction to Java

THEORY: JAVA was developed by James Gosling at Sun Microsystems Inc in the year 1995, later acquired by Oracle Corporation. It is a simple programming language. Java makes writing, compiling, and debugging programming easy. It helps to create reusable code and modular programs. **Java** is a class-based, object-oriented programming language and is designed to have as few implementation dependencies as possible. A general-purpose programming language made for developers to *write once run anywhere* that is compiled Java code can run on all platforms that support Java. Java applications are compiled to byte code that can run on any Java Virtual Machine. The syntax of Java is similar to c/c++.

Java Terminology

Before learning Java, one must be familiar with these common terms of Java.

1. Java Virtual Machine(JVM): This is generally referred to as [JVM](#). There are three execution phases of a program. They are written, compile and run the program.

- Writing a program is done by a java programmer like you and me.
- The compilation is done by the **JAVAC** compiler which is a primary Java compiler included in the Java development kit (JDK). It takes the Java program as input and generates bytecode as output.
- In the Running phase of a program, **JVM** executes the bytecode generated by the compiler.

Now, we understood that the function of Java Virtual Machine is to execute the bytecode produced by the compiler. Every Operating System has a different JVM but the output they produce after the execution of bytecode is the same across all the operating systems. This is why Java is known as a **platform-independent language**.

2. Bytecode in the Development process: As discussed, the Javac compiler of JDK compiles the java source code into bytecode so that it can be executed by JVM. It is saved as **.class** file by the compiler. To view the bytecode, a disassembler like [javap](#) can be used.

3. Java Development Kit(JDK): While we were using the term JDK when we learn about bytecode and JVM. So, as the name suggests, it is a complete Java development kit that includes everything including compiler, Java Runtime Environment (JRE), java debuggers, java docs, etc. For the program to execute in java, we need to install JDK on our computer in order to create, compile and run the java program.

4. Java Runtime Environment (JRE): JDK includes JRE. JRE installation on our computers allows the java program to run, however, we cannot compile it. JRE includes a browser, JVM, applet supports, and plugins. For running the java program, a computer needs JRE.

Primary/Main Features of Java

1. Platform Independent: Compiler converts source code to bytecode and then the JVM executes the bytecode generated by the compiler. This bytecode can run on any platform be it Windows, Linux, or macOS which means if we compile a program on Windows, then we can run it on Linux and vice versa. Each operating system has a different JVM, but the

output produced by all the OS is the same after the execution of bytecode. That is why we call java a platform-independent language.

2. Object-Oriented Programming Language: Organizing the program in the terms of collection of objects is a way of object-oriented programming, each of which represents an instance of the class.

The four main concepts of Object-Oriented programming are:

- Abstraction
- Encapsulation
- Inheritance
- Polymorphism

3. Simple: Java is one of the simple languages as it does not have complex features like pointers, operator overloading, multiple inheritances, and Explicit memory allocation.

4. Robust: Java language is robust which means reliable. It is developed in such a way that it puts a lot of effort into checking errors as early as possible, that is why the java compiler is able to detect even those errors that are not easy to detect by another programming language. The main features of java that make it robust are garbage collection, Exception Handling, and memory allocation.

5. Secure: In java, we don't have pointers, so we cannot access out-of-bound arrays i.e it shows **ArrayIndexOutOfBoundsException** if we try to do so. That's why several security flaws like stack corruption or buffer overflow are impossible to exploit in Java. Also java programs run in an environment that is independent of the os(operating system) environment which makes java programs more secure .

6. Distributed: We can create distributed applications using the java programming language. Remote Method Invocation and Enterprise Java Beans are used for creating distributed applications in java. The java programs can be easily distributed on one or more systems that are connected to each other through an internet connection.

7. Multithreading: Java supports multithreading. It is a Java feature that allows concurrent execution of two or more parts of a program for maximum utilization of the CPU.

8. Portable: As we know, java code written on one machine can be run on another machine. The platform-independent feature of java in which its platform-independent bytecode can be taken to any platform for execution makes java portable.

9. High Performance: Java architecture is defined in such a way that it reduces overhead during the runtime and at some time java uses Just In Time (JIT) compiler where the compiler compiles code on-demand basics where it only compiles those methods that are called making applications to execute faster.

10. Dynamic flexibility: Java being completely object-oriented gives us the flexibility to add classes, new methods to existing classes and even create new classes through sub-classes. Java even supports functions written in other languages such as C, C++ which are referred to as native methods.

11. Sandbox Execution: Java programs run in a separate space that allows user to execute their applications without affecting the underlying system with help of a bytecode verifier. Bytecode verifier also provides additional security as its role is to check the code for any violation of access.

12. Write Once Run Anywhere: As discussed above java application generates a '.class' file which corresponds to our applications(program) but contains code in binary format. It provides ease t architecture-neutral ease as bytecode is not dependent on any machine architecture. It is the primary reason java is used in the enterprising IT industry globally worldwide.

PROGRAM NO: 2

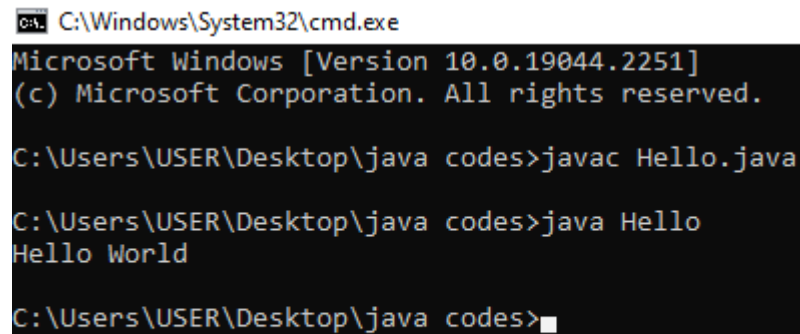
AIM: Write a Program to Print HelloWorld.

THEORY: A "Hello, World!" is a simple program that outputs Hello, World! on the screen. Since it's a very simple program, it's often used to introduce a new programming language to a newbie. Let's explore how Java "Hello, World!" program works.

PROGRAM CODE:

```
class Hello{  
    public static void main(String[] args){  
        System.out.println("Hello World");  
    }  
}
```

OUTPUT:



```
C:\Windows\System32\cmd.exe  
Microsoft Windows [Version 10.0.19044.2251]  
(c) Microsoft Corporation. All rights reserved.  
  
C:\Users\USER\Desktop\java codes>javac Hello.java  
  
C:\Users\USER\Desktop\java codes>java Hello  
Hello World  
  
C:\Users\USER\Desktop\java codes>
```

PROGRAM NO: 3

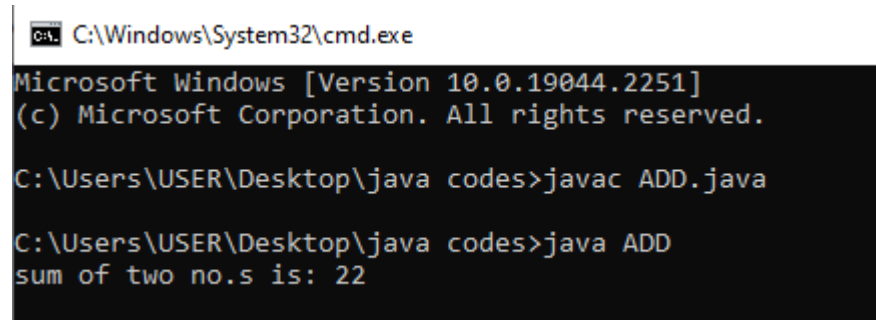
AIM: Write a Program to add two numbers without Scanner.

THEORY: In Java, finding the sum of two or more numbers is very easy. First, declare and initialize two variables to be added. Another variable to store the sum of numbers. Apply mathematical operator (+) between the declared variable and store the result. The following program calculates and prints the sum of two numbers.

PROGRAM CODE:

```
class ADD {  
    public static void main(String[] args) {  
        int a=10,b=12,sum;  
        sum=a+b;  
        System.out.println("sum of two no.s is: "+sum);  
    }  
}
```

OUTPUT:



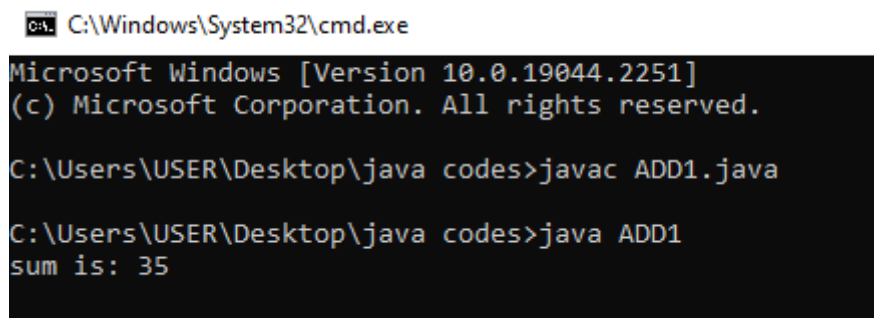
```
C:\Windows\System32\cmd.exe  
Microsoft Windows [Version 10.0.19044.2251]  
(c) Microsoft Corporation. All rights reserved.  
C:\Users\USER\Desktop\java codes>javac ADD.java  
C:\Users\USER\Desktop\java codes>java ADD  
sum of two no.s is: 22
```

By objects:

```
class A{  
    int a,b;  
    void input(int c, int d)  
    {  
        a=c;  
        b=d;  
    }  
    void add(){  
        System.out.println("sum is: "+a+b);  
    }  
}
```

```
    }  
}  
class ADD1{  
    public static void main(String[] args){  
  
        A obj = new A();  
        obj.input(3,5);  
        obj.add();  
    }  
}
```

OUTPUT:



```
C:\Windows\System32\cmd.exe  
Microsoft Windows [Version 10.0.19044.2251]  
(c) Microsoft Corporation. All rights reserved.  
  
C:\Users\USER\Desktop\java codes>javac ADD1.java  
  
C:\Users\USER\Desktop\java codes>java ADD1  
sum is: 35
```

PROGRAM NO: 4

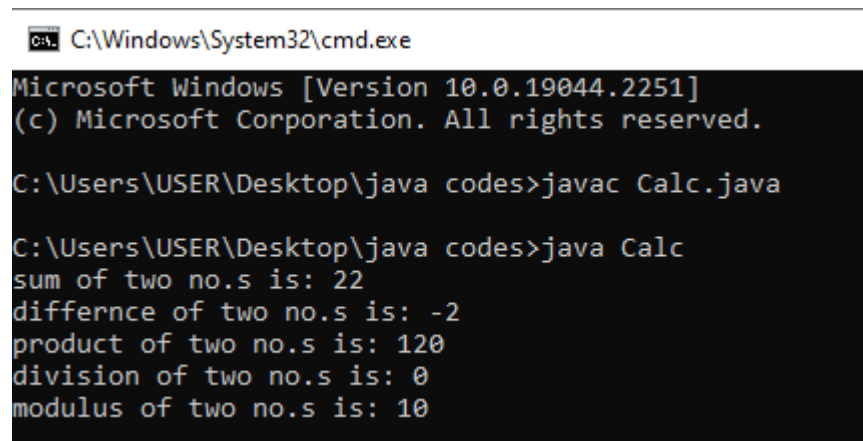
AIM: Write a program to perform all mathematical calculations without Scanner.

THEORY: This is a Java Program to Illustrate the Use of Arithmetic Operators. Initialize two integers as input. After that we perform different arithmetic operations like addition, subtraction, multiplication, division and modulus to get the desired output. It is same as above (Addition program) only difference is to use other operators also such as '-', '*', '/', '%'. Here is the source code of the Java Program to Illustrate the Use of Arithmetic Operators. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

PROGRAM CODE:

```
class Calc {  
    public static void main(String[] args) {  
        int a=10,b=12,sum,sub,product,div,mod;  
        sum=a+b;  
        sub=a-b;  
        product=a*b;  
        div=a/b;  
        mod=a%b;  
  
        System.out.println("sum of two no.s is: "+sum);  
        System.out.println("diffrence of two no.s is: "+sub);  
        System.out.println("product of two no.s is: "+product);  
        System.out.println("division of two no.s is: "+div);  
        System.out.println("modulus of two no.s is: "+mod);  
    }  
}
```

OUTPUT:



```
cmd C:\Windows\System32\cmd.exe  
Microsoft Windows [Version 10.0.19044.2251]  
(c) Microsoft Corporation. All rights reserved.  
  
C:\Users\USER\Desktop\java codes>javac Calc.java  
  
C:\Users\USER\Desktop\java codes>java Calc  
sum of two no.s is: 22  
diffrence of two no.s is: -2  
product of two no.s is: 120  
division of two no.s is: 0  
modulus of two no.s is: 10
```


PROGRAM NO: 5

AIM: Write a program to sum two numbers by taking user input.

THEORY: The first step in taking user input is to import special functions into our program. Java runs fairly lean, meaning that it doesn't include all functions in all projects. You can import only the ones you need. This reduces run-time and makes code cleaner. We'll be importing the utility Scanner, which allows input/output. Scanner is actually part of the overall Java util (utility) package. You could just import all of the util package, but since we're only taking user input from the keyboard, we will use the following statement at the top of our program:

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

Now that we've imported the utility, we can begin to take user input. But, before we can do that, we need to create a variable to store the values entered by the user. The Scanner utility is actually a Java class, so we can simply create an instance of that class. This will be used to read in the values, which you can see play out below.

PROGRAM CODE:

```
import java.util.*;

class AddScan {

public static void main(String args[]){

    int a,b,sum;

    Scanner sc = new Scanner(System.in);

    System.out.print("enter the first no.: ");

    a=sc.nextInt();

    System.out.print("enter the second no.: ");

    b=sc.nextInt();

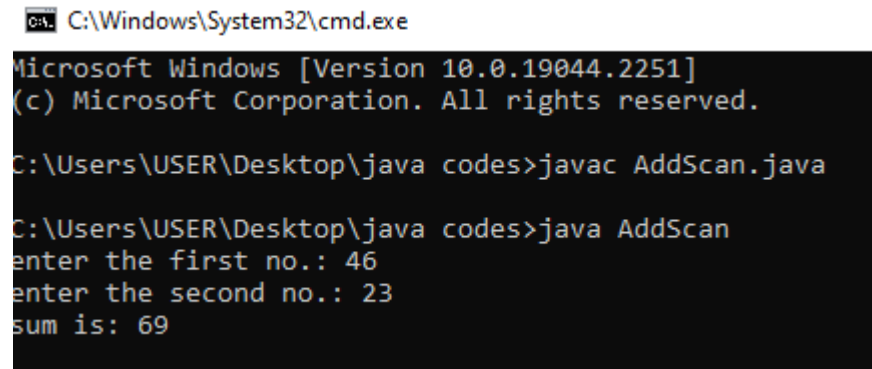
    sum=a+b;

    System.out.println("sum is: "+sum);

}

}
```

OUTPUT:



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19044.2251]
(c) Microsoft Corporation. All rights reserved.

C:\Users\USER\Desktop\java codes>javac AddScan.java

C:\Users\USER\Desktop\java codes>java AddScan
enter the first no.: 46
enter the second no.: 23
sum is: 69
```

PROGRAM NO: 6

AIM: Write a program to make menu using Switch case.

THEORY: Here we are going to make a program to follow the switch statement. For completion the example firstly we define a class named "A" and two integer type values from users. Java I/O Package has an input Stream and an output Stream in which input stream is used for reading the stream and memory allocating. As in this program we are going to create a buffer for the string class that can be used to store and process a string character. Here in this program we used the pursuing method for two integer values a and b.

This Program also provides you an example with complete java source code for understanding more about the **switch** statement in Java. The following java program tells you for entering numbers for involving in mathematical operations as your choice (whatever you want to do with both numbers). Four types of operations are listed through the program in sequence. Whatever you have to perform the operation firstly you have to enter your choice as an existing operation number for selecting operation what you want to do with entered two numbers.

PROGRAM CODE:

```
import java.util.Scanner;

class A{
    public static void main(String args[]){

        int a,b;
        char operation;

        Scanner input = new Scanner(System.in);

        System.out.println("enter the first no.:");
        a = input.nextInt();
        System.out.println("enter the first no.:");
        b = input.nextInt();

        System.out.println("choose the operation (+,-,*,/,%): ");
        operation = input.next().charAt(0);

        switch(operation){
            case '+':
                System.out.println("sum of two no.s is: ");
                System.out.println(a+b);
                break;

            case '-':
                System.out.println("difference of two no.s is: ");
                System.out.println(a-b);
                break;
```

```

case '*':
System.out.println("product of two no.s is: ");
System.out.println(a*b);
break;

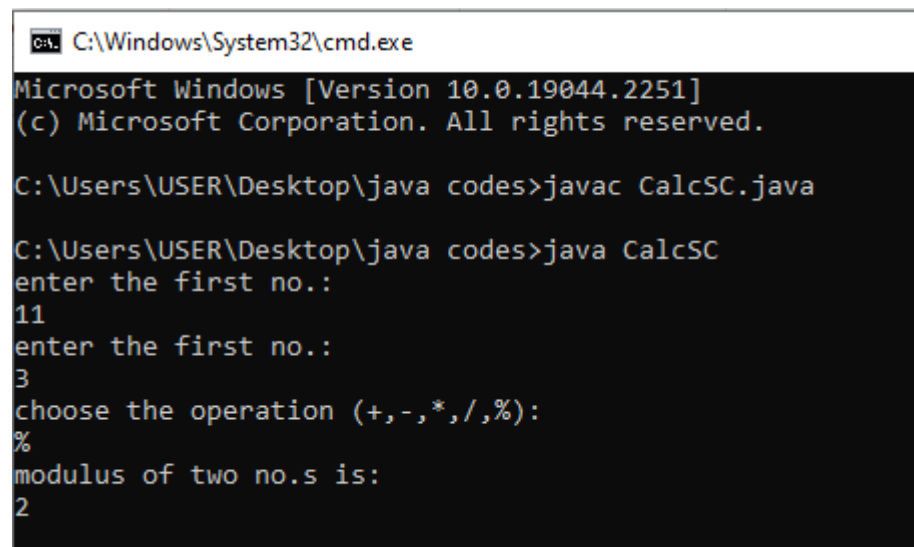
case '/':
System.out.println("division of two no.s is: ");
System.out.println(a/b);
break;

case '%':
System.out.println("modulus of two no.s is: ");
System.out.println(a%b);
break;

default:
System.out.println("Invalid input!!");
break;
}
input.close();
}
}

```

OUTPUT:



```

C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19044.2251]
(c) Microsoft Corporation. All rights reserved.

C:\Users\USER\Desktop\java codes>javac CalcSC.java

C:\Users\USER\Desktop\java codes>java CalcSC
enter the first no.:
11
enter the first no.:
3
choose the operation (+,-,*,/,%):
%
modulus of two no.s is:
2

```

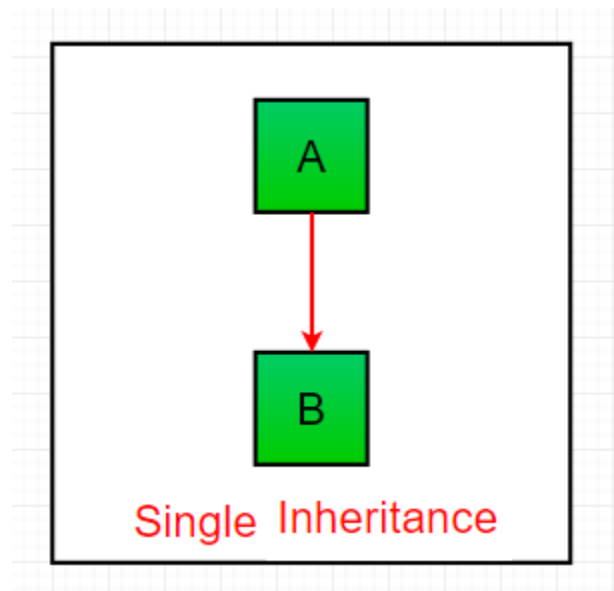
PROGRAM NO: 7

AIM: Write a Program to perform single inheritance that can solve a real life reusability problem.

THEORY: Inheritance in Java is a mechanism in which one object acquires all the properties and behaviors of a parent object. It is an important part of OOPs (Object Oriented programming system).

The idea behind inheritance in Java is that you can create new classes that are built upon existing classes. When you inherit from an existing class, you can reuse methods and fields of the parent class. Moreover, you can add new methods and fields in your current class also.

Inheritance represents the **IS-A relationship** which is also known as a *parent-child* relationship.



The **extends keyword** indicates that you are making a new class that derives from an existing class. The meaning of "extends" is to increase the functionality.

Terms used in Inheritance

- **Class:** A class is a group of objects which have common properties. It is a template or blueprint from which objects are created.
- **Sub Class/Child Class:** Subclass is a class which inherits the other class. It is also called a derived class, extended class, or child class.
- **Super Class/Parent Class:** Superclass is the class from where a subclass inherits the features. It is also called a base class or a parent class.

PROGRAM CODE:

```

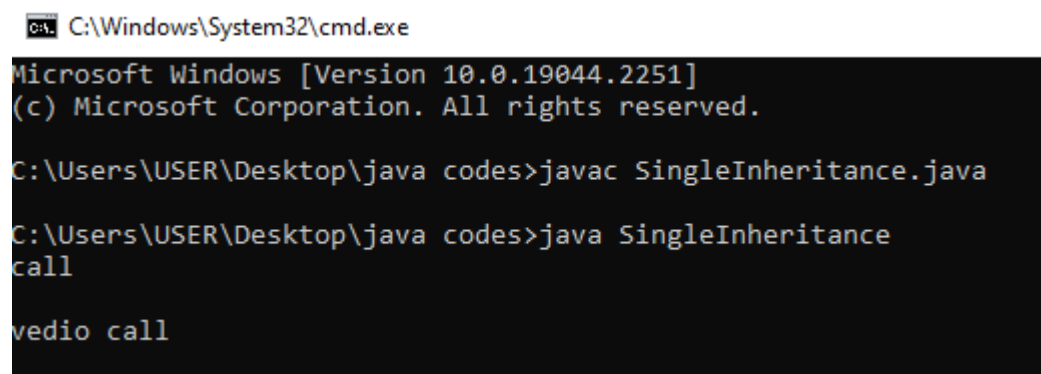
class Phone {
    void call(){
        System.out.println("call");
        System.out.println(" ");
    }
}

class Smartphone extends Phone{
    void vedio_call(){
        System.out.println("vedio call");
    }
}

class SingleInheritance{
    public static void main(String args[]){
        Smartphone obj =new Smartphone();
        obj.call();
        obj.vedio_call();
    }
}

```

OUTPUT:



```

C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19044.2251]
(c) Microsoft Corporation. All rights reserved.

C:\Users\USER\Desktop\java codes>javac SingleInheritance.java

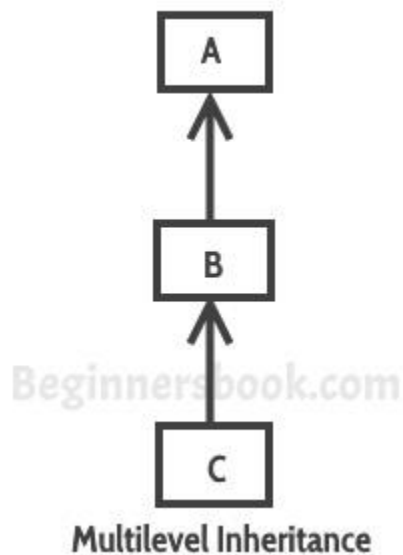
C:\Users\USER\Desktop\java codes>java SingleInheritance
call
vedio call

```

PROGRAM NO: 8

AIM: Write a program showing multi level inheritance where each class at new features and inherit some features from the base class.

THEORY: When there is a chain of inheritance, it is known as *multilevel inheritance*. As you can see in the example given below, Smartphone class inherits the Phone class which again inherits the Telephone class, so there is a multilevel inheritance.

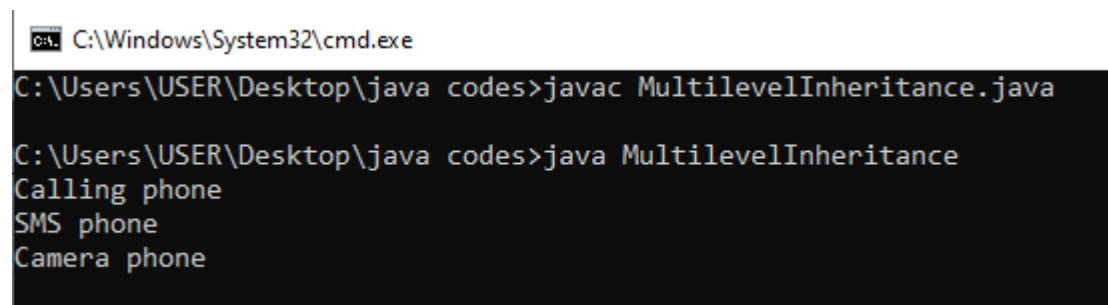


PROGRAM CODE:

```
class Telephone {  
    public void call(){  
        System.out.println("Calling phone");  
    }  
}  
  
class Phone extends Telephone{  
    public void sms(){  
        System.out.println("SMS phone");  
    }  
}  
  
class Smartphone extends Phone{  
    public void camera(){  
        System.out.println("Camera phone");  
    }  
}
```

```
}  
class MultilevelInheritance{  
    public static void main(String args[]){  
        Smartphone obj = new Smartphone();  
        obj.call();  
        obj.sms();  
        obj.camera();  
    }  
}
```

OUTPUT:



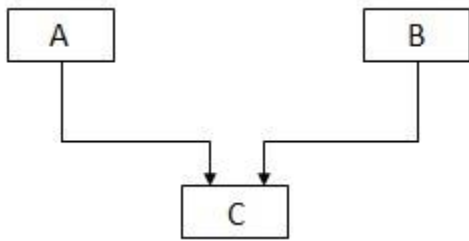
The screenshot shows a Windows command prompt window with the title bar "C:\Windows\System32\cmd.exe". The command prompt is open at the directory "C:\Users\USER\Desktop\java codes". The user has entered the command "javac MultilevelInheritance.java" to compile the Java file. The next command entered is "java MultilevelInheritance", which has been executed, resulting in the output: "Calling phone", "SMS phone", and "Camera phone" on separate lines.

```
C:\Windows\System32\cmd.exe  
C:\Users\USER\Desktop\java codes>javac MultilevelInheritance.java  
C:\Users\USER\Desktop\java codes>java MultilevelInheritance  
Calling phone  
SMS phone  
Camera phone
```


PROGRAM NO: 9

AIM: Write a program to perform multiple inheritance using interfaces.

THEORY: Multiple Inheritance is a feature of an object-oriented concept, where a class can inherit properties of more than one parent class. The problem occurs when there exist methods with the same signature in both the superclasses and subclass. On calling the method, the compiler cannot determine which class method to be called and even on calling which class method gets the priority. An interface contains variables and methods like a class but the methods in an interface are abstract by default unlike a class. Multiple inheritance by interface occurs if a class implements multiple interfaces or also if an interface itself extends multiple interfaces.



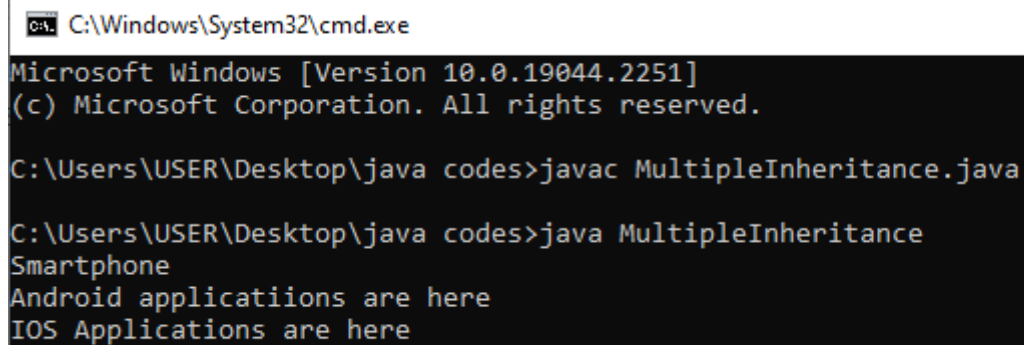
A program that demonstrates multiple inheritance by interface in Java is given as follows:

PROGRAM CODE:

```
interface Android{  
    public void Playstore();  
}  
  
interface IOS{  
    public void Applestore();  
}  
  
class Smartphone implements Android,IOS{  
    public void Playstore(){  
        System.out.println("Android applicatiions are here ");  
    }  
    public void Applestore(){  
        System.out.println("IOS Applications are here ");  
    }  
}
```

```
class MultipleInheritance{  
    public static void main(String args[]){  
        System.out.println("Smartphone");  
        Smartphone obj = new Smartphone();  
        obj.Playstore();  
        obj.Applestore();  
    }  
}
```

OUTPUT:



```
C:\Windows\System32\cmd.exe  
Microsoft Windows [Version 10.0.19044.2251]  
(c) Microsoft Corporation. All rights reserved.  
  
C:\Users\USER\Desktop\java codes>javac MultipleInheritance.java  
  
C:\Users\USER\Desktop\java codes>java MultipleInheritance  
Smartphone  
Android applications are here  
IOS Applications are here
```

PROGRAM NO: 10

AIM: Write a program of abstract class.

THOERY: A class which is declared with the abstract keyword is known as an abstract class in [Java](#). It can have abstract and non-abstract methods (method with the body).

Abstraction is a process of hiding the implementation details and showing only functionality to the user. Another way, it shows only essential things to the user and hides the internal details, for example, sending SMS where you type the text and send the message. You don't know the internal processing about the message delivery.

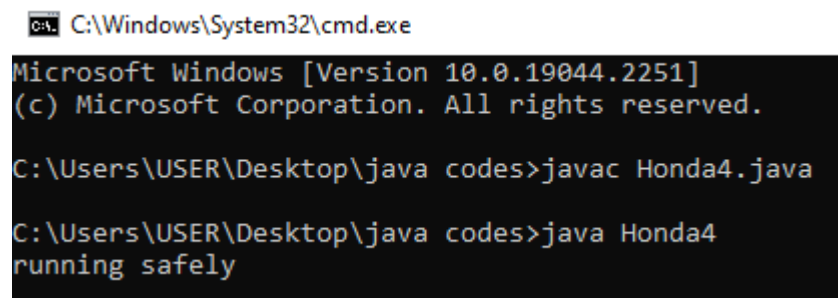
A method which is declared as abstract and does not have implementation is known as an abstract method.

In this example, Bike is an abstract class that contains only one abstract method run. Its implementation is provided by the Honda class.

PROGRAM CODE:

```
abstract class Bike{
    abstract void run();
}
class Honda4 extends Bike{
    void run(){
        System.out.println("running safely");
    }
    public static void main(String args[]){
        Bike obj = new Honda4();
        obj.run();
    }
}
```

OUTPUT:



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19044.2251]
(c) Microsoft Corporation. All rights reserved.

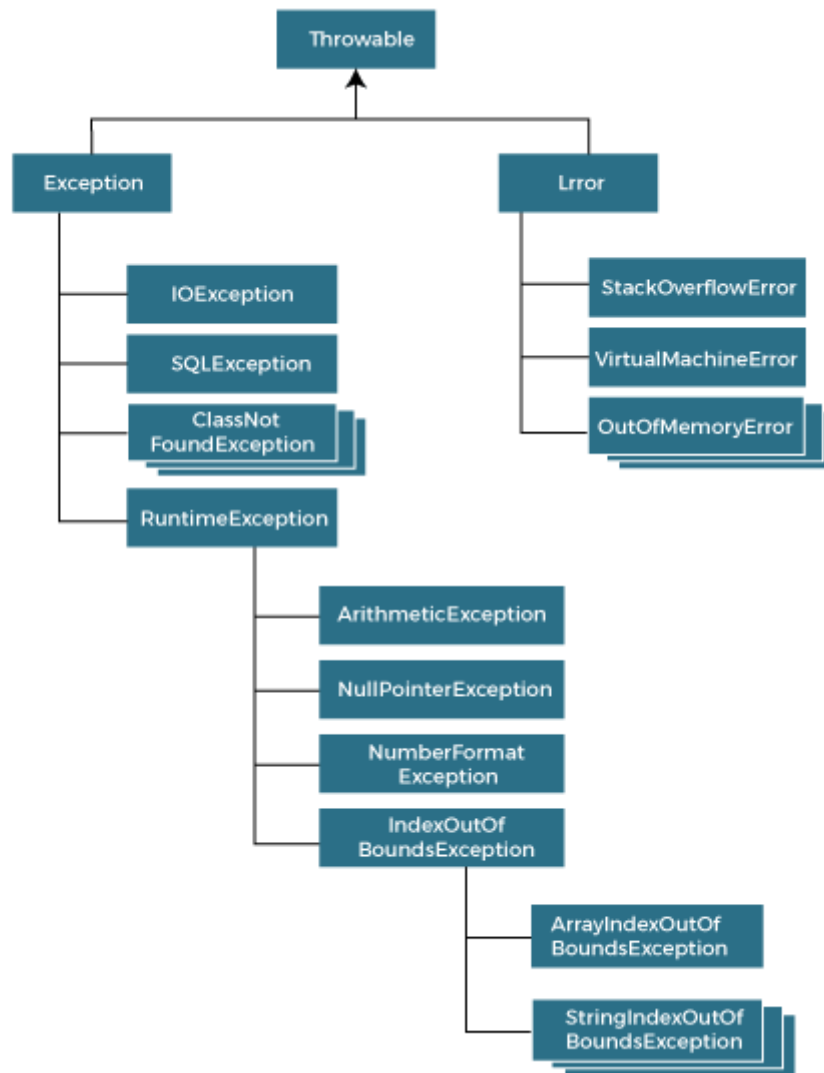
C:\Users\USER\Desktop\java codes>javac Honda4.java

C:\Users\USER\Desktop\java codes>java Honda4
running safely
```

PROGRAM NO: 11

AIM: Write a program to perform exception handling using try and catch

THEORY: The Exception Handling in Java is one of the powerful *mechanism to handle the runtime errors* so that the normal flow of the application can be maintained. The core advantage of exception handling is **to maintain the normal flow of the application**. An exception normally disrupts the normal flow of the application; that is why we need to handle exceptions



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.

PROGRAM CODE:

```

public class ExceptionHandle {

    public static void main(String[] args) {

        try
        {
            int data=50/0;
        }

        catch(Exception e)
        {
            System.out.println(e);
        }

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

```

OUTPUT:

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19044.2251]
(c) Microsoft Corporation. All rights reserved.

C:\Users\USER\Desktop\java codes>javac ExceptionHandle.java

C:\Users\USER\Desktop\java codes>java ExceptionHandle
java.lang.ArithmeticException: / by zero
rest of the code
```

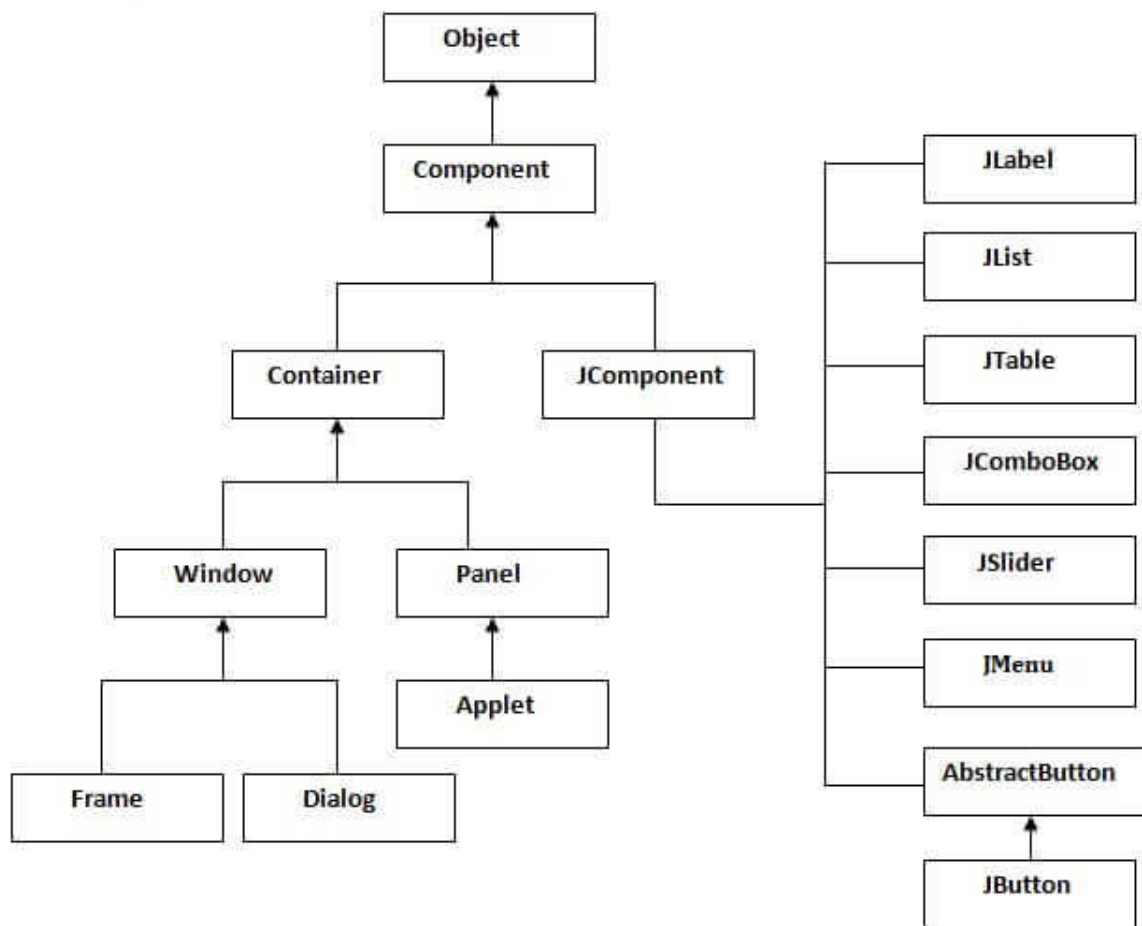
PROGRAM NO: 12

AIM: Write a program to create an application using swing

THEORY: Java Swing tutorial is a part of Java Foundation Classes (JFC) that is *used to create window-based applications*. It is built on the top of AWT (Abstract Windowing Toolkit) API and entirely written in java.

Unlike AWT, Java Swing provides platform-independent and lightweight components.

The javax.swing package provides classes for java swing API such as JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu, JColorChooser etc.



The methods of Component class are widely used in java swing that are given below.

Method	Description
public void add(Component c)	add a component on another component.
public void setSize(int width,int height)	sets size of the component.
public void setLayout(LayoutManager m)	sets the layout manager for the component.
public void setVisible(boolean b)	sets the visibility of the component. It is by default false.

PROGRAM CODE:

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

public class SwingAddApp
{
    public static void main(String args[])
    {
        Addp obj=new Addp();
    }
}

class Addp extends JFrame implements ActionListener
{
    JLabel l1;
    JTextField t1;
    JLabel l2;
    JTextField t2;
    JButton b;
    JLabel l3;

    public Addp()
    {
        setLayout(new FlowLayout());
        l1=new JLabel("First Number:");
        t1=new JTextField(20);

        l2=new JLabel("Second Number:");
        t2=new JTextField(20);

        b=new JButton("Add");

        l3=new JLabel("Result");

        add(l1);
```



```

        add(t1);
        add(l2);
        add(t2);
        add(b);
        add(l3);

        b.addActionListener(this);

        setVisible(true);
        setSize(250,400);

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    public void actionPerformed(ActionEvent ae)
    {
        int num1=Integer.parseInt(t1.getText());
        int num2=Integer.parseInt(t2.getText());

        int value=num1+num2;
        l3.setText(""+value);
    }
}

```

OUTPUT:

