JAVA programming

Name : R. Laranya

Reg No : 192321022

Inheritance.

Inheritance is a fundamental concept in object-oriented programming (oop) that allows a new class to inherit fields and methods from an existing class.

Single Inheritance:

Single inheritance is a type of inheritance in which a subclass inherits from only one superclass

ex:

Class A

↓

Class B

program:

```
class A {
        int a;
        void displayA() {
                System.out.println ("a=" +a);
        }
}
class B extends A
{
```

```
    int b;
    void display B() {
        System.out.println("b="+b);
    }
}
public class Main {
    public static void main(String[] args)
    {
        B obj = new B();
        obj.a = 10;
        obj.b = 20;
        obj.displayA();
        obj.displayB();
    }
}
```

O/p: 10
20

Multiple inheritance in Java:

Multiple inheritance refers to feature in some Object-oriented programming languages where a class can inherit characteristics from more than one parent class.

ex:

```
    }
    public void display(c) {
        System.out.println(c);
    }
}
public class Main {
    public static void main(String[] Args) {
        cd = new C();
        d.displayA();
        d.displayB();
        d.displayC();
    }
}
```

O/P. 5
10
15

Multilevel :-

Multilevel inheritance is a type of inheritance in Java where a class is derived from a class that is also derived from another class.

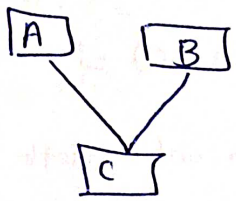
Class A

↓

Class B

↓

Class C



Program :-

```
Class A {
    int a;
  A() {
    a=5;
  }
   Void display A() {
        System.out.println(a);
    }
}
interface B {
    int b = 10;
    void display B();
}
class C extends A implements B {
        int c;
      C() {
        c = 15;
    public void display B() {
```

```
C obj = new C();
obj.display A();
obj.display B();
obj.display C();
```

o/p:

Basc class

child

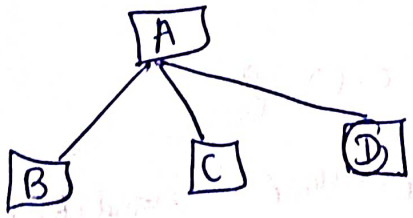
Grand child.

Hybrid:

Hierarchical Inheritance:

It occurs when multiple Subclass inherit from a single Superclass. This means that a single parent class can have multiple child classes.

ex:



program:

```
class A {
    public void display A() {
        System.out.println("Inside class A");

class B extends A {
```

ex:

Class A

↓

class B

↓

class C

program:-

```
Class A {
    public void display A() {
        System.out.println ("Baseclass");
    }
}
Class B extends A {
    public void display B() {
        System.out.println ("childclass");
    }
}
Class C extends B {
    public void display C() {
        System.out.println ("Grandchild");
    }
}
public class main {
    public static void main (String [] args)
    {
```

```
    public void display B() {
        System.out.println("Inside class B");
Class C extends A {
    public void display C() {
        System.out.println("Inside class C");
Class D extends A {
    public void display D()
    {
        System.out.println("Inside class D");
    }
}
public class Main {
    public static vad Main (String[] args)
    {
        B objB = new B();
        C objC = new C();
        D objD = new D();
    objB.display A();
    objB.display B();
    objC.display A();
    objC.display C();
    obj.display A();
    obj.display D();
```

o/p:

Inside class A

Inside class B

Inside class A

Inside class C

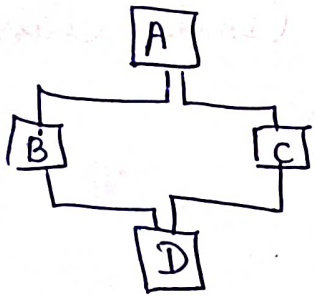
Inside class A

Inside class D

Hybrid:.

Combination of any inheritance

ex:



program:

```
class grandfather {
        public void showG() {
                System.out.println("He is grandfather");
        }
}
class father extends Grenandfather {
        public void showF() {
                System.out.println("He is father");
        }
}
```

```
Class Son extends father {
        public void shows() {
                System.out.println("He is son");
        }
}
public class Daughter extends father {
        public vid showc() {
                System.out.println("she is daughter");
        }
public static vid main(String[] args)
{
        Son obj = new son();
        obj.showS();
        obj.showF();
        obj.showGr();
    Daughter obj2 = new Daughter();
        obj2.showD();
        obj2.showF();
        obj.showGr();
    }
}
```

O/p:

He is son

He is father

He is grand father

She is daughter

She is father

He is grandfather.

Exception Handling:

An Exception is an error during the execution of a program.

Key Components of Exception Handling.

* Try
* Catch
* Throw
* Finally

Nested Catch::

```
class Main {
    public static void main (String[] args)
    {
        try {
            int a = 5/0;
```

```
        System.out.println("Rest of code in try block");
    }
    Catch(Arithmetic Exception e)
    {
      System.out.println("Arithmetic Exception => " + e.getMessage());
    }
  Catch(Exception e)
  {
    System.out.println("Exception => " + e.getMessage());
  }
}
```

O/p: Arithmetic exception => / by zero

2 try 2 catch:-

```
class Main{
    public static void main(String[] args)
    {
      try{
        int a[] = {1,2,3};
        try{
            int b = 1/0;
        }
        catch (Exception e)
        {
            System.out.println("Exception thrown" + e.
                                    getMessage());
        }
```

```
        System.out.println(a[4]);
    }
    catch(ArrayIndexOutOfBoundException e)
    {
        System.out.println("Exception thrown :" + e.getmessage());
    }
    System.out.println("out of Block");
    }
}
```

O/p: Exception thrown : / by zero

throw:- → dellare the error

```
public class Main {
    static void checkAge(int age) throws ArithmeticException
    {
        if (age < 18)
        {
            throw new ArithmeticException("Access denied - you
                must be at least 18
                year old");
        }
        else
        {
            System.out.println("Access granted - you are old
                enough");
        }
    }
```

```
    public static void main (String[] args)
    {
    try
    {
        checkAge(16);
    }
    Catch (Arithmetic Exception e)
    {
      System.out.println(e.getMessage());
    }
  }
}
```

o/p: you are not eligible

Finally:-

Try → block of code to test the error being executed

catch → block of code to be executed if an error occurs in try block

Finally → Code that always ~~error~~ executes.

The finally blak is a section of code that is executed regardless of whether an exception is thrown or not.

program:

```
public class Main {
    public static void main (String[] args)
    {
    try
```

```
{
  int a[] = {1,2,3}
  System.out.println ("Rest of the Code in the try block");
}
catch (ArrayIndexOutOfBoundException e)
{
  System.out.println ("Array index out of Exception : " +
                      e.getMessage());
}
finally
{
  System.out.println ("This is the finally blocks");
}
}.
```

o/p:

Arithmetic Exception => / by zero

This is the finally blocks.