1. 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.

Examp:

class A
↓
class B

Program:

```
class A {
    int a;
    void display A() {
        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. display A();
         obj . display B();
     }
}
```

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.

Examp:

class A

class B

class C

Program:

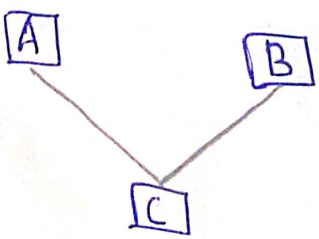
```
class A {
    Public void method A()
    {
      System. out. println ("Inside class A");
    }
}
class B extends A {
      Public void method B()
      {
        system. out. println ("Inside class B");
      }
}
```

```
class C
{
    Public void method()
    {
        system.out.println ("Inside class C");
    }
}
Public class Main
{
    Public static void main (string[] args)
    {
        C objc = new C();
        objc = method A();
        objc . method B();
        objc . method C();
    }
}
```

Multiple Inheritance:

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

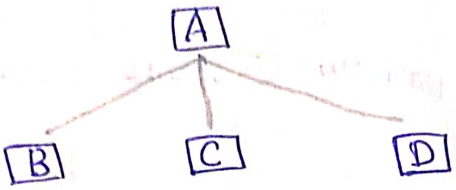
Examp:



```
class A {
    void displayA()
     {
      system.out.println("Inside class A")
     }
   }
class B {
     void displayB()
      {
       system.out.println("Inside class B")
      }
    }
class C extends A(){
       void display A()
       {
        super.display A();
        system.out.println("Inside class C");
       }
       void display()
        {
         system.out.println("Method of class C");
        }
     }
Public class Main
 {
   Public static void main (string[] args)
     {
        C obj C = new C();
          obj C . display A();
```

```
        B obj B = new B();
        obj B . display B();
        obj C . display C(),
    }
}
```

Hierarchial Management



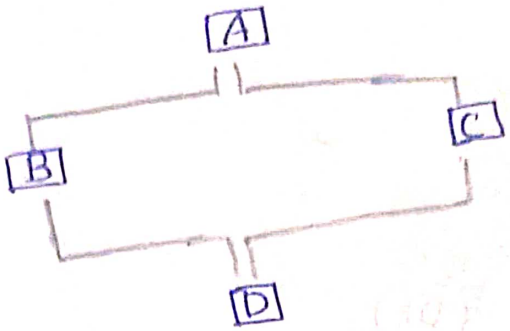
It occurs when multiple subclass inherit from a single superclass. This means that a single parent class can have multiple classes.

```
class A {
    Public void display A()
      {
        system . out. println ("Inside class A");
      }
    }
class B extends A{
    Public void display B()
    {
      system . out . println ("Inside class B");
    }
    }
```

```
class C extends A {
    Public void displayCC()
    {
        system.out.println("Inside class C");
    }
}

class D extends B{
    Public void displayDC()
    {
        system.out.println("Inside class D");
    }
}

Public class Main
{
    Public obj void system main (string[] . args
    {
        B obj B = newB();
        C obj C = newC();
        D obj D = new D();

        obj B. displayAC();
        obj B. display B:
        system. out. println();

        obj D. display AC();
        obj D. display BC();
        obj D. display DC();
    }
}
```

Hybrid :

Combination of any inheritance.




```
class A {
    Public void display A()
    { system . out println (" Inside class A")
    }
}
class B {
    Public void display B()
    { system . out . println (" Inside class B"); } }
class D exteds class C{
  Public . void println (" Insid class D") display
    {
      system . out . println (" Inside class D")
    }
}
Public class main
    {
      Public static void main (string [] args)
       {
          C objc = newc();
          C objD = newD;
```

```
obj c . display();
obj c . displayc();
System.out.println();

obj D . displayA();
obj D . displayc();
obj D . displayD();
    }
}
```

Exception Handling:

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

Key components of exception Handling:

Try.
catch.
Throw.
Finally.
Throws.

Nested catch:

```
class Main {
    Public static void main (string [] args)
        {
          try
          {
            int a = 5/10;
        system.out.println ("Rest or code in key blok");
        }
      catch (Arithmetic exception e)
        { system . out. println ("Arithmetic expression" + e. get mes
        }
    catch (Exception e)
      { system . out. println ("exception =") + e. getMessage());
      }
    }
}
```

Arithmetic exception by zero.

Try & catch:

```
class Main {
Public static void main (string[] / args)
 {
   try {
      int b= 1/0;
       }
   catch exception e
   {
    system. out. println ("Exception known" + e. getMessage());
    }
   system. out. println (a[4]);
  }
  catch (Array Index order Bound Exception)
    {
     system. out. println ("Exception thrown" + e. getMessage());
    }
   system. out . println (" out. of Block")
  }
 }
```

O/p: Exception thrown (o)

Throw :

```
Public class Main {
  static void checkage (int known age) throws Arithmeticexception
    {
      if (age <19)
    {
    throw new arithmetic exception ("Access defined - You must
                                    be at-least 18 Your world")
    }
  else
    {
      System .out . println ("Access granted - You are old
                                    enough");
    }
}
Public static void main (string[] args)
{
  try
  {
    check age (16)
  }
  catch (Arithmetic exception e)
    { system . out . println (e. get message ());
    }
  }
  }
```

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 executes.

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

Program:

```
Public          Main{
    Public static void main (string[] args)
    {
      try {
       int a = {1,2,3}
       system. out. println ("array index out of exception:"
                                + e. getMessage ());
      }
      catch (Array index out of Bound exception)
      {
       system . out. println (" Rest of code in the
                                try block");

      }
```

```
Finally
  {
    system.out.println("this is the finally blocks");
  }
}
```

Arithmetic => / zero.

∴ this is the finally blocks.