**Inheritance**

**Example:**

class Parent {

String name = "Ranjan Kumar Singh";

void getName() {

System.out.println(this.name);

}

}

class Child extends Parent {

}

public class Main {

// main Method

public static void main(String[] args) {

Child child = new Child();

child.getName();

}

}

**Output:**Ranjan Kumar Singh

**Types of Inheritance**

1. Single Inheritance
2. Multilevel Inheritance
3. Hierarchical Inheritance
4. Multiple Inheritance (Through Interfaces)
5. Hybrid Inheritance (Through Interfaces)
6. **Single Inheritance**

In this Inheritance, there will be only one super/base/parent class & child/derived/parent class.

**#1\_Example:**

class Engine {

void startEngine() {

System.out.println("Engine Started");

}

}

class Car extends Engine {

}

public class Main {

// main Method

public static void main(String[] args) {

Car car = new Car();

car.startEngine();

}

}

**Output:**

Engine Started

**#2\_Example:**

class Parent {

String name = "Ranjan Kumar Singh";

void getName() {

System.out.println(this.name);

}

}

class Child extends Parent {

String name = "Chandan Kumar";

void getName(String data) {

System.out.println("This is child class's Method");

}

}

public class Main {

// main Method

public static void main(String[] args) {

Child child = new Child();

**// trying to call the parameterized method existed in child class**

child.getName("hello");

System.out.println(child.name);

}

}

**Output:**

This is child class's Method

Chandan Kumar

There is a concept like **public, private & protected** property in inheritance.

**Public:** public makes the function inherited by the child class. So, public access modifier is the default property of any method inside the class.  
**Example:**

class Engine {

public void startEngine() {

System.out.println("Engine Started");

}

}

class Car extends Engine {

}

public class Main {

// main Method

public static void main(String[] args) {

Car car = new Car();

car.startEngine();

}

}

**Output:**

Engine Started

**Private:** private makes the function restricted to inherit by the child class.  
**Example:**

class Engine {

private void startEngine() {

System.out.println("Engine Started");

}

}

class Car extends Engine {

}

public class Main {

// main Method

public static void main(String[] args) {

Car car = new Car();

car.startEngine();

}

}

**Output:**

Throw an Error

**Protected:** protected property is basically use in multilevel inheritance.

**Note:**

If the function of parent & child class is same, then child method will override parent’s function.

1. **Multilevel Inheritance**

At least 3 classes are needed to form multilevel inheritance.

A 🡪 B 🡪 C

Then in multilevel inheritance, C will automatically able to inherit property of class A. A 🡪 C

**Example:**

class Company {

void getTotalEmployee() {

System.out.println("Total employee 5000");

}

}

class CountryBranch extends Company {

void getTotalCountryEmployee() {

System.out.println("Total employee 1000");

}

}

class LocalBranch extends CountryBranch{

void getTotalLocalEmployee(){

System.out.println("Total employee 500");

}

}

public class Main {

// main Method

public static void main(String[] args) {

LocalBranch lB = new LocalBranch();

***//calling method of super Class of its parent class without inheriting that.***   
 lB.getTotalEmployee();

}

}

**Output:**Total employee 5000

1. **Hierarchical Inheritance**

At least 3 classes are needed to form hierarchical inheritance.

If a parent class is inherited in multiple child classes called Hierarchical Inheritance.

A 🡪 B  
A 🡪 C  
A 🡪 D

**Example:**

class Company {

// Hierarchical Inheritance

void companyName(){

System.out.println("Name of the Company is Capgemini");

}

}

class CountryBranch extends Company {

void getTotalCountryEmployee() {

System.out.println("Total employee 1000");

}

}

class LocalBranch extends Company{

void getTotalLocalEmployee(){

System.out.println("Total employee 500");

}

}

public class Main {

// main Method

public static void main(String[] args) {

CountryBranch cB = new CountryBranch();

LocalBranch lB = new LocalBranch();

cB.companyName();

lB.companyName();

}

}

**Output:**  
Name of the Company is Capgemini

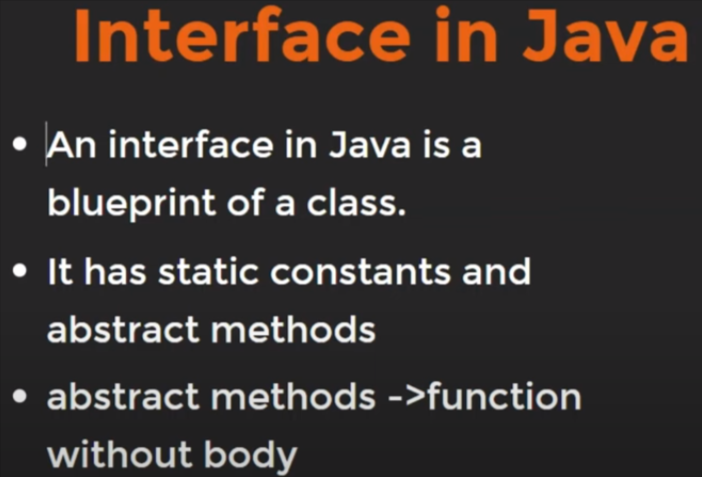
Name of the Company is Capgemini

**Note:**

Out of 5 types of Inheritance, Multiple & Hybrid inheritance go through “**Interface**”.

**Interface**

Interface basically means to create a list in a class. It ultimately act like a class.



Function without body means function would create any block with curly braces.

Note:

* To create interface, we have to use ‘interface’ keyword.
* Its naming convention would be camelCase.
* In order To take the methods of an interface inside a class, we’ve to use ‘implements’ keyword.
* Any methods or functions of an interface should be public.
* Every function that’s calling from the interface should be define inside the class.

**Example:**

interface driving{

void startEngine();

void applyBreak();

}

class Car implements driving{

public void startEngine(){

System.out.println("Engine Started");

}

public void applyBreak(){

System.out.println("Break Applied");

}

// A seperate method of this class. Doesn't need to be public

void changeGear(){

System.out.println("Gear Changed");

}

}

public class Main {

// main Method

public static void main(String[] args) {

Car cr = new Car();

cr.startEngine();

cr.changeGear();

}

}

**Output:**  
Engine Started

Gear Changed

1. **Multiple Inheritance**

Child having property/method of two different parents.

**A** **B**  
 \ /  
 **C**

**Example:**

// Parent-1 Interface

interface itServices {

public void getItServiceDetails();

}

// Parent-2 Interface

interface hardwareServices {

public void getHardwareServiceDetails();

}

// Parent-3 Interface

interface allBranch {

public void getAllBranch();

}

// Child Class

class Company implements itServices, hardwareServices {

public void getItServiceDetails() {

System.out.println("Got IT service Details");

}

public void getHardwareServiceDetails() {

System.out.println("Got Hardware service Details");

}

public void getAllBranch() {

System.out.println("Got All Branches");

}

}

public class Main {

// main Method

public static void main(String[] args) {

Company cp = new Company();

cp.getItServiceDetails();

cp.getHardwareServiceDetails();

cp.getAllBranch();

}

}

**Output:**Got IT service Details  
Got Hardware service Details  
Got All Branches

1. **Hybrid Inheritance**

Mixture of at least any two inheritance called Hybrid Inheritance.

Ex:   
Hybrid Inheritance = *Multilevel Inheritance* + *Hierarchical Inheritance*

**Code: (*Multilevel Inheritance* + *Hierarchical Inheritance****)*

class BaseCompany{

void getBaseCompanyDetails(){

System.out.println("Got Base Company Details");

}

}

class ChildCompany extends BaseCompany{

}

class LocalCompany extends ChildCompany{

}

class SharedCompany extends BaseCompany{

}

public class Main {

// main Method

public static void main(String[] args) {

LocalCompany lCp = new LocalCompany();

SharedCompany sCp = new SharedCompany();

lCp.getBaseCompanyDetails();

sCp.getBaseCompanyDetails();

}

}

**Output:**Got Base Company Details

Got Base Company Details

**Code: (*Multilevel Inheritance* + *Multiple Inheritance + Heirarchical****)*

interface itService{

void getItServiceDetails();

}

interface hardwareService{

void getHardwareServiceDetails();

}

class BaseCompany implements itService, hardwareService{

void getBaseCompanyDetails(){

System.out.println("Got Base Company Details");

}

public void getItServiceDetails(){

System.out.println("Got It Service Details");

}

public void getHardwareServiceDetails (){

System.out.println(“Got Hardware Service Details”);

}

}

class ChildCompany extends BaseCompany{

}

class LocalCompany extends ChildCompany{

}

class SharedCompany extends BaseCompany{

}

public class Main {

// main Method

public static void main(String[] args) {

LocalCompany lCp = new LocalCompany();

SharedCompany sCp = new SharedCompany();

***// Multilevel + Heirarchical***

lCp.getBaseCompanyDetails();

sCp.getBaseCompanyDetails();

***// Multilevel + Multiple***  
 lCp.getItServiceDetails();

sCp.getHardwareServiceDetails();

}

}

**Output:**

Got Base Company Details

Got Base Company Details

Got It Service Details