Chapter 1 OOPS JAVA

B.tech (Dr. A.P.J. Abdul Kalam Technical University)

Object Oriented Programming Using ' JAVA'

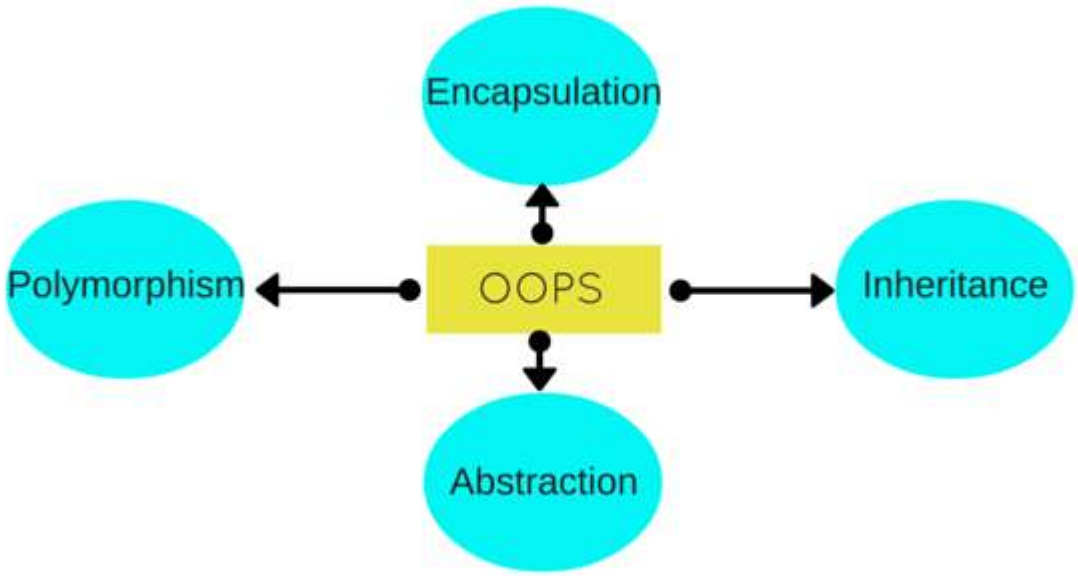
Chapter-1

Introduction and Features

OOP:-

ऑब्जेक्ट ओरिएन्टेड प्रोग्रामिग** implementation **की एक विधि है जिसमें** program objects **के** cooperative **(सहकारी) संग्रह के रूप में संगठित रहते हैं जिनमें से प्रत्येक ऑब्जेक्ट किसी** Class **के** Instance **का** Representation **करता है एवं जिनकी** Classes Inheritance relationship **के द्वारा क्लासों की** Hierarchy **की** Member **होती है

ऑब्जेक्ट ओरिएन्टेड प्रोग्रामिंग एक ऐसी अवधारणा है जो डेटा तथा फंक्शन के लिए इस तरह अलग - अलग Memory Area का निर्माण कर Program का मोड्यूलराइजेशन करते हैं जिससे उन्हें मोड्यूल की कॉपी करने के लिए टेम्पलेट की तरह प्रयोग किया जा सकता है



Need Of OOPS :-

High Level Language , जैसे - कोबॉल . फोरट्रॉन या C में लिखे प्रोग्राम **प्रोसीजर ओरिएन्टेड प्रोग्राम** कहलाते हैं ।

इस प्रकार की प्रोग्रामिंग में प्रोग्राम को किये जाने वाले कार्यों के रूप में देखा जाता है ।

इन कार्यों को करने के लिए एक से अधिक फंक्शन लिखे जाते हैं ।

फंक्शन के विकास पर ध्यान केन्द्रित करते समय उस डेटा को भूल जाते हैं जिसे इन फंक्शन द्वारा प्रयोग किया जाता है ।

इसमें global dala का प्रयोग करते हैं अर्थात एक ही Data बहुत से प्रोग्राम द्वारा प्रयोग किया जाता है ।

डेटा को आसानी से एक से दूसरे फंक्शन में भेजा जा सकता है ।

इन सारी समस्याओं से निदान पाने के लिए **ऑब्जेक्ट ओरिएन्टेड प्रोग्रामिंग** का विकास किया गया है ।

इसमें फंक्शन की अपेक्षा डेटा को प्रोग्राम के विकास का महत्वपूर्ण अवयव माना जाता है ।

#GPM

इसमें डेटा , उन फंक्शन के साथ ही जुड़े होते हैं जिसमें इनका प्रयोग किया जाता है ।
सिस्टम में Data को External Function द्वारा Independent रूप से बदले जाने की अनुमति नहीं होती है ।
इसमें डेटा एवं इस पर ऑपरेट करने वाले फंक्शन एक - दूसरे से निकटता (Nearest)
से बँधे होते हैं एवं यह बाहरी फंक्शन द्वारा डेटा के आकस्मिक बदलाव से सुरक्षा करती है ।
ऑब्जेक्ट ओरिएन्टेड प्रोग्रामिंग में समस्या को Entities में विभाजित किया जा सकता है जिन्हें ऑब्जेक्ट कहते हैं और फिर इन ऑब्जेक्ट्स के लिए डेटा एवं फंक्शन बनते हैं । एक ऑब्जेक्ट का डेटा केवल उस ऑब्जेक्ट से जुड़े हुए फंक्शन द्वारा एक्सेस किया जा सकता है जबकि एक ऑब्जेक्ट के फंक्शन दूसरे ऑब्जेक्टस के फंक्शन्स को एक्सेस कर सकते हैं ।

Features of Object Oriented Programming Language :-

Object Oriented Programming Language के निम्न Features हैं, और इसके Basic Concept निम्न हैं-

1. **Objects**
2. **Class**
3. **Data Abstraction**
4. **Polymorphism**
5. **Data Encapsulation or Data Hiding**
6. **Inheritance**
7. **Message Passing**
8. **Reusability**
9. **Dynamic Binding**

1. Objects

यह OOPs की basic run-time entity है। जो कि किसी object (person, place, a bank account etc.) को represent करता है।Object User define डेटा टाइप्स जैसे - वेक्टर , समय एवं सूची (list) आदि को भी प्रदर्शित कर सकते हैं ।object, class का variable है जो कि class को execute करता है और उसमे उपलब्ध methods को use कर डाटा को process करता है। object के create होने पर यह memory मे अन्य variables की तरह ही space लेता है।
प्रोग्राम के ऑब्जेक्ट्स का चयन इस प्रकार किया जाना चाहिए कि वे वास्तविक विश्व के समान ही हों या उससे मिलता जुलता हो

2. Class

C + + में क्लास का अत्यधिक महत्व है । इसी महत्ता के कारण शुरू में इस भाषा का नाम ही **विथ क्लास** ' था ।क्लास की Concept के प्रयोग के कारण ही C + + प्रोसीजर भाषा के साथ - साथ **ऑब्जेक्ट ओरिएन्टेड प्रोग्रामिंग** की एक Advance भाषा बन सकी है । Class यूजर के द्वारा बनाये जाना वाला डेटा टाइप है । क्लास उन ऑब्जेक्ट्स का समूह होती है , जिनके Property Same होते हैं एवं जिनका व्यवहार व संबंध (relationships) साधारण (Common) होता हैं । ऑब्जेक्ट्स में डेटा एवं उस डेटा को मैनिपुलेट करने के लिए Code Contain रहता है ।एक ऑब्जेक्ट के डेटा एवं कोड का Complete set क्लास की सहायता से यूजर डिफाइन्ड डेटा टाइप बनाया जा सकता है ।वास्तव में , ***ऑब्जेक्ट्स क्लास टाइप के वेरियेबल होते हैं । एक बार क्लास के परिभाषित होने के बाद , उस क्लास से संबंधित (belonging) ऑब्जेक्ट्स के कितने भी मेम्बर बनाए जा सकते हैं ।*** प्रत्येक ऑब्जेक्ट क्लास टाइप के डेटा से associate रहता है , जिससे वो बनता है ।अतः एक क्लास समान टाइप के ऑब्जेक्ट्स का संग्रह होती है ।
उदाहरण - mango , apple एवं orange **क्लास fruit** के मेम्बर हैं |
चूँकि क्लास यूजर - डिफाइन्ड है और क्लास एक प्रोग्रामिंग भाषा के डेटा टाइप की तरह व्यवहार करता है

। अतः यदि fruit एक क्लास की तरह डिफाइन होता है , तब कथन , fruit mango ; एक ऑब्जेक्ट Generate करेगा जो fruit क्लास से संबंधित है ।

3. Data Abstraction

Data Abstraction का आशय Total Complexity के Simplification से हैं

डेटा एब्सट्रैक्शन में किसी कार्य को Complete करने के लिए केवल आवश्यक ज्ञान ही पर्याप्त है , न कि उस कार्य की Internal Process एवं उससे सम्बन्धित Indirect (अप्नत्यक्ष) रूप से होने वाले कार्य के ज्ञान की ।

उदाहरण - लाइट ऑन करने के लिए स्विच बोर्ड पर केवल एक स्विच को दबाने का कार्य किया जाता है । इस कार्य के लिए स्विच को दबाने से स्विच के अन्दर क्या हुआ स्विच के दबाने से लाइट कैसे ऑन होती है , यह सब जानने की आवश्यकता नहीं होती है | यह गुण एब्सट्रैक्शन कहलाता है ।

4. Polymorphism

Polymorphism ' पॉली ' (poly) शब्द की उत्पत्ति **ग्रीक** शब्द से हुई है जिसका अर्थ है ' अनेक ' (many) एवं मॉर्फिज्म (morphism) का अर्थ है । रूप ' (form) । अतः पॉलीमॉर्फिज्म का अर्थ है " अनेक रूप ' (many forms) |

ऑब्जेक्ट ओरिएन्टेड प्रोग्रामिंग में पॉलीमॉर्फिज्म का अर्थ है , समान नाम के फंक्शन्स (भेम्बर फंक्शन्स) इन फंक्शन्स का व्यवहार जिन ऑब्जेक्ट्स को वे Specify करते हैं , उनके आधार पर अलग - अलग होता है ।

5. Data Encapsulation Or data Hiding

Data Encapsulation ऑब्जेक्ट ओरिएन्टेड प्रोग्रामिंग शैली का एक अत्यन्त महत्वपूर्ण तथ्य है । डेटा और फंक्शन को एकल इकाई (class) में संगठित करना , एनकैप्सुलेशन कहलाता है ।एनकैप्सुलेशन ऑब्जेक्ट के Internal Form को यूजर से छुपाता है और उपयोग हो सकने वाले ऑब्जेक्ट को सूचित भी करता है । **डेटा एनकैप्सुलेशन** अर्थात् Functions व Data का Integration करना ।डेटा एनकैप्सुलेशन के द्वारा प्रोग्राम एवं डेटा , दूसरे प्रोग्राम के द्वारा प्रभावित नहीं होते हैं । इस प्रकार डेटा एनकैप्सुलेशन एक रक्षक की तरह कार्य करता है

6. Inheritance

इनहेरिटेन्स भी ऑब्जेक्ट ओरिएन्टेड प्रोग्राम शैली की एक महत्वपूर्ण एवं उपयोगी विशेषता है । ***इसके द्वारा एक Class की Properties किसी दूसरे Class से प्राप्त की जा सकती हैं या Inherit की जा सकती हैं***

इस प्रकार यह पूर्व में उपस्थित किसी क्लास का नए क्लास के रूप में परिभाषित करने की क्षमता रखता है । पहले से बने क्लास को Base Class क्लास कहते हैं और नए क्लास को Derived Class क्लास कहते हैं ।

उदाहरण - मोटरसाइकिल अपने आप में एक क्लास है एवं यह दुपहिया क्लास का मेम्बर है । दुपहिया क्लास ऑटोमेटिव क्लास का सदस्य है एवं ऑटोमेटिव बेस क्लास है और दुपहिया डिराइब्ड क्लास है । इस प्रकार मोटरसाइकिल एक दुपहिया ऑटोमेटिव है ।

7. Message Passing

Message Passing ऑब्जेक्ट ओरिएन्टेड प्रोग्रामिंग में , ऑब्जेक्ट ओरिएन्टेड प्रोग्राम अनेक ऑब्जेक्ट्स के समूह होते हैं , जो आपस में एक - दूसरे को आवश्यकता पड़ने पर मैसेज Send हैं और Receive भी करते हैं ।

किसी ऑब्जेक्ट्स के लिए एक मैसेज एक निश्चित प्रक्रिया अथवा फंक्शन को Implement करने के लिए होता है । अतः मैसेज प्राप्त करके ऑब्जेक्ट एक निश्चित प्रक्रिया को Implement करके Result प्रस्तुत करता है ।

उदाहरण - मैसेज पासिंग ऑब्जेक्ट का नाम , विधि (method) का नाम और सूचना जो भेजना है , को अपने में रखता है । जैसे -
Statement employee. salary (name) ; यहाँ employee Object है , salary मैसेज है । और name पैरामीटर है जो सूचना (information) को एकत्र करता है

8. Reusability

Reusability एक नई क्लास लिखने , Create करने और Debug करने के बाद इसे दूसरे प्रोग्राम्स में उपयोग के लिए Distributed किया जा सकता है , इसे रियूजेबिलिटी कहा जाता है ।

यह किसी प्रोसीजरल लैंग्वेज में फंक्शन की लाइब्रेरी को भिन्न प्रोग्राम में जोड़ने के समान है । Reusability की Concept डेटा ऑपरेशन्स के Abstraction और Encapsulation के साथ Generate होती है ।

रियूजेबिलिटी में ऑब्जेक्ट ओरिएन्टेड डेवलपमेन्ट निम्नलिखित प्रकार से होता है-

1. एक एप्लीकेशन में सूचनाओं को शेयर किया जाता है ।
2. Future Projects में कोड और डिजाइन्स का Reuse किया जाता है ।

9. Dynamic Binding

बाइंडिंग प्रोसीजर कॉल की उस डेटा से क्रिया को denote करती है , जिसे प्रोसीजर कॉल के प्रत्युत्तर में एक्जीक्यूट किया जाता है । डायनेमिक बाइंडिंग का अर्थ दी गई प्रोसीजर कॉल से संबंधित कोड की जानकारी कॉल के रनटाइम के समय तक न होना है ।

यह पॉलीमॉर्फिज्म और इनहेरिटेन्स से संबंधित होती है । पॉलीमॉर्फिक रेफरेन्स से जुड़ा एक फंक्शन कॉल उस रेरेन्स के डायनेमिक टाइप पर निर्भर करता है

Advantage Of OOPS :-

Object Oriented Programming Language के निम्नलिखित लाभ हैं:-

1. इस पद्धति में Inheritance के द्वारा redundant code को हटाया जा सकता है एवं पहले से उपस्थित Classes के उपयोग को extend किया जा सकता है ।
2. डेटा हाइडिंग का Concept प्रोग्रामर को सुरक्षित प्रोग्राम बनाने में सहायता करता है जिसे कोड द्वारा प्रोग्राम के दूसरे भागों में प्रयोग नहीं किया जा सकता है ।
3. OOP की तकनीक से एक Program को Quickly (शीघ्रतापूर्वक) Operational हेतु Objects के समूहों के आधार पर अनेक भागों में विभक्त किया जा सकता है ।
4. इस तकनीक से छोटे - छोटे Program के Combination द्वारा Large Program सरलतापूर्वक तैयार किए जा सकते हैं

5. इसके द्वारा कार्यरत स्तरीय मॉड्यूलों का संबंध सरलतापूर्वक दूसरे मॉड्यूलों से हो जाता है , जिससे कोड को बार - बार नहीं लिखना पड़ता है , जिसके कारण प्रोग्राम में समय की बचत होती ही है तथा Develop की क्षमता में वृद्धि होती है ।
6. इस तकनीक के द्वारा Object की Classes के साथ उनसे संबंधित Functions को भी integrated कर दिया जाता है , जिससे प्रोसेसिंग का कार्य सुगम और सुरक्षित हो जाता है ।
7. Objects के बीच Communication के लिए मैसेज पासिंग Method द्वारा External System के साथ इंटरफेस करना आसान होता है ।
8. इस प्रोग्रामिंग के द्वारा सॉफ्टवेयर जटिलता (complexity) को आसानी से हल किया जा सकता है ।
9. सॉफ्टवेयर विकसित (develop) करना आसान होता है ।

Java Class and Object:-

Class:-

एक class object का एक group है। यह एक Template या blueprint है जिसमें से object बनाए जाते हैं। यह एक logical entity है। यह physical entity नहीं हो सकता।

java में सब कुछ class और object के साथ-साथ इसकी विशेषताओं और method के साथ जुड़ा हुआ है। उदाहरण के लिए: वास्तविक जीवन में, एक car एक object है। car में वजन और रंग, जैसे ड्राइव और ब्रेक जैसी विशेषताएं हैं।

A class in Java can contain:

- **Fields**
- **Methods**
- **Constructors**
- **Blocks**
- **Nested class and interface**

Create a Class:-

Java में class create करना बहुत ही easy है इसके लिए आप class keyword का यूज़ करते है।

```
class Student
{
     String Name = "Ankit";
     int age = 20;

     public void display()
```

```
    {
          System.out.println("Name is "+Name+"Age is"+age);
    }

    public static void main(String args[])
    {
           this.Show();
    }
}
```

उदाहरण में Student नाम की एक class create की गयी है। इस class में name और age दो variables है। इसमें Show() नाम से method भी declare किया गया है।

Java Object:-

Java में, Class से Obejct बनाया जाता है। हमने पहले ही Student नाम की Class बना ली थी, इसलिए अब हम इसका इस्तेमाल object बनाने के लिए कर सकते हैं।

यदि हम वास्तविक दुनिया पर विचार करते हैं, तो हम अपने आस-पास कई object, car, dogs, humans आदि को पा सकते हैं। इन सभी वस्तुओं का एक State और एक Behaviour होता है।

Create an object:-

```
public class Student{
  int x = 5;
     String Name = "Ankit";
     int age = 20;

     public void display()
     {
           System.out.println("Name is "+Name+"Age is"+age);
     }

     public static void main(String args[])
     {
           this.Show();
     }
}

  public static void main(String[] args) {
    Student myobj  = new Student();
    System.out.println(myobj.x);
  }
}
```

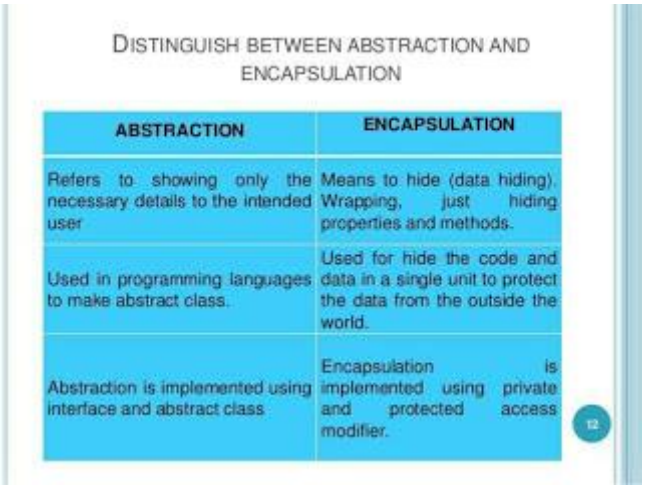
सबसे पहले हमे object बना ने के लिए जावा basic syntax लिखा उसके बाद । जब आप कोई भी object create करते है तो सबसे पहले उस class का constructor call होता है। contructor बना ने के लिए हमे brackets का यूज़ करना पड़ता है।

उसके बाद object print करने के लिए System.out.println (myobj.x); लिखा हमने।

इस तरह से Java मैं Class और Object बनाए जाते है।

Abstraction & encapsulation-

Data तथा function को एक single unit में wrap करना encapsulation कहलाता है तथा वह single unit जिससे data wrap हुआ है class कहलाती है, एक class के अंदर data outside world से access नही किया जा सकता, अगर जरुरी features को show कर background detail को hide कर दिया जाये तो यह तरीका data abstraction कहलाता है।



DISTINGUISH BETWEEN ABSTRACTION AND ENCAPSULATION

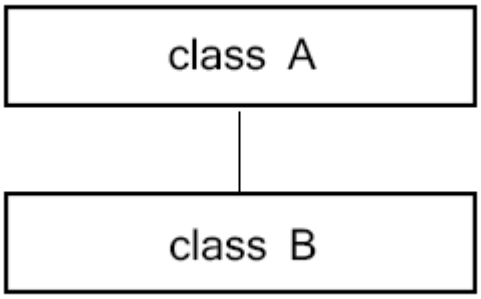
ABSTRACTION	ENCAPSULATION
Refers to showing only the necessary details to the intended user	Means to hide (data hiding). Wrapping, just hiding properties and methods.
Used in programming languages to make abstract class.	Used for hide the code and data in a single unit to protect the data from the outside the world.
Abstraction is implemented using interface and abstract class	Encapsulation is implemented using private and protected access modifier.

Inheritance:-

Java में Inheritance ये Object Oriented Programming का एक प्रकार है | जिसमे एक class की properties और methods किसी दुसरे class में inherit की जाती है |

Inheritance में मुख्यतः Parent class और child class का इस्तेमाल किया जाता है | इसमे Parent class को Base class या super class भी कहा जाता है और Child class को Derived class या sub class भी कहा जाता है | C++ ये Inheritance के प्रकार को support करता है, लेकिन Java; Multiple Inheritance को support नहीं करता मतलब Java में Parent class को कई child classes हो सकते है, लेकिन child classes को सिर्फ एक ही Parent class होता है |

Sample Inheritance



Java में Inheritance के लिए extends keywword का इस्तेमाल किया जाता है |

Syntax for Inheritance

```
class parent_class{

        //statements;
}
class child_class extends parent_class{

        //statements;
}
```

Example for Inheritance

यहाँ example में A ये एक parent class है और B ये child class है | B class; A class की सभी properties; inherit कर सकता है |

```
class A{

        //statements;

}
class B extends A{

        //statements;
```

```
}
```

Full Example for Inheritance

Source Code :

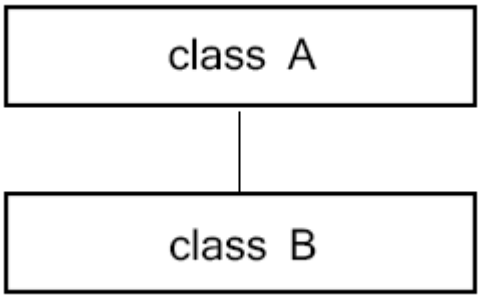
```
//B.java
class A{


        void disp(){
                System.out.println("Parent Class");
        }
}
class B extends A{
        void show(){
                System.out.println("Child Class");
        }
        public static void main(String args[]){
        B obj = new B();
        obj.disp();
        obj.show();
        }
}
```

Output :

```
Parent Class
Child Class
```

Java में तीन Inheritance होते है |

1. Single Inheritance



Source Code :

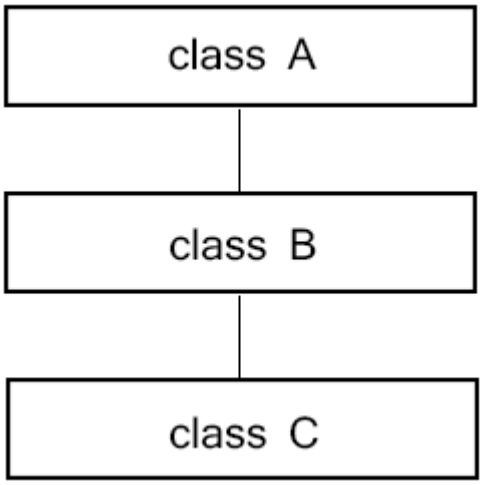
```
//B.java
class A{


		void disp(){
			System.out.println("Parent Class");
		}
}
class B extends A{
		void show(){
			System.out.println("Child Class");
		}
		public static void main(String args[]){
		B obj = new B();
		obj.disp();
		obj.show();
		}
}
```

Output :

```
Parent Class
```

Child Class

2. Multilevel Inheritance



Source Code :

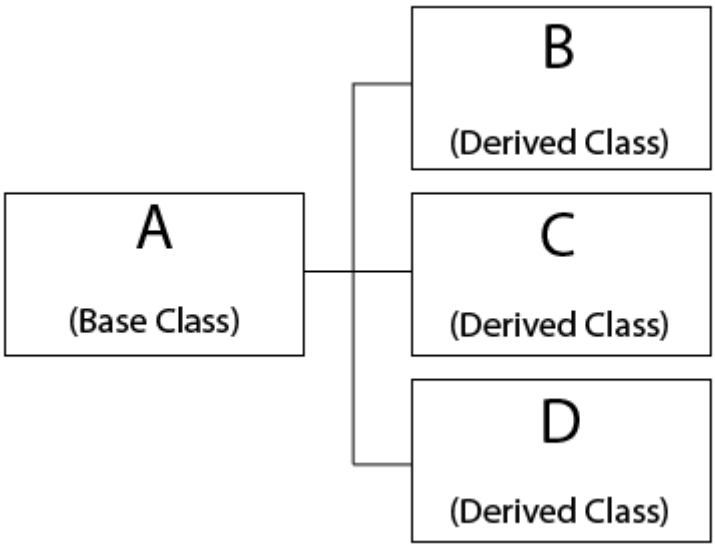
```
//C.java
class A{


        void disp(){
                System.out.println("Class A");
        }
}
class B extends A{
        void show(){
                System.out.println("Class B");
        }
}
class C extends B{
        void getdata(){
                System.out.println("Class C");
        }
        public static void main(String args[]){
```

```
        C obj = new C();
        obj.disp();
        obj.show();
        obj.getdata();
        }
}
```

Output :

```
Class A
Class B
Class C
```

2. Hierarchical Inheritance



Source Code :

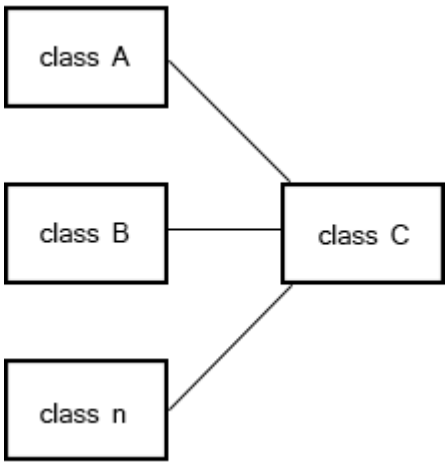
```
//C.java
class A{

        void disp(){
                System.out.println("Class A");
        }
```

```
}
class B extends A{
        void show(){
                System.out.println("Class B");
        }
}
class C extends A{
        void getdata(){
                System.out.println("Class C");
        }
        public static void main(String args[]){
        C obj = new C();
        obj.disp();
        obj.getdata();
        B b = new B();
        b.show();
        }
}
```

Output :

```
Class A
Class C
Class B
```

Multiple Inheritance



Multiple Inheritance के लिए Java में एक से ज्यादा Base classes होते है और सभी base classes को एक ही derived class inherit करता है |

Java में Multiple Inheritance; support नहीं करता है ,लेकिन interface के माध्यम से multiple Inheritance Java में support करता है |

Multiple Inheritance; Java में support क्यों नहीं करता ?

Java में जब Multiple Inheritance आता है, तब Ambiguity problem आ जाता है | Java में extends के साथ कहा पर भी Multiple Inheritance का उल्लेख नहीं किया गया है | लेकिन interface में इसे इस्तेमाल किया जा सकता है |

Java में एक से ज्यादा parent class नहीं हो सकते | Program में तीन class है | C class; A और B इन दोनों class को inherit करता है | Source Code :

```
//C.java
class A{

        void disp(){
                System.out.println("Class A");
        }
}
class B{
        void show(){
                System.out.println("Class B");
        }
}
class C extends A, B{
```

#GPM

```
        void getdata(){
                System.out.println("Class C");
        }
        public static void main(String args[]){
        C obj = new C();
        obj.disp();
        obj.show();
        obj.getdata();
}
}
```

Output :

```
Error.
```