INHERITANCE

in JAVA

INHERITANCE

- One of the most effective features of Oop's paradigm.

- Establish a link/connectivity between 2 or more classes.

- Permits sharing and accessing properties from one to another class.

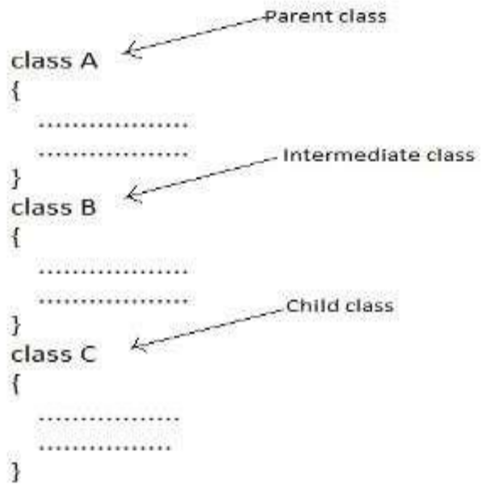
- to establish this relation Java uses '***extends***' keyword.

Category of Classes on the Basis of Inheritance

Super class
(base/parent/driver/inheritance/ancestor class).

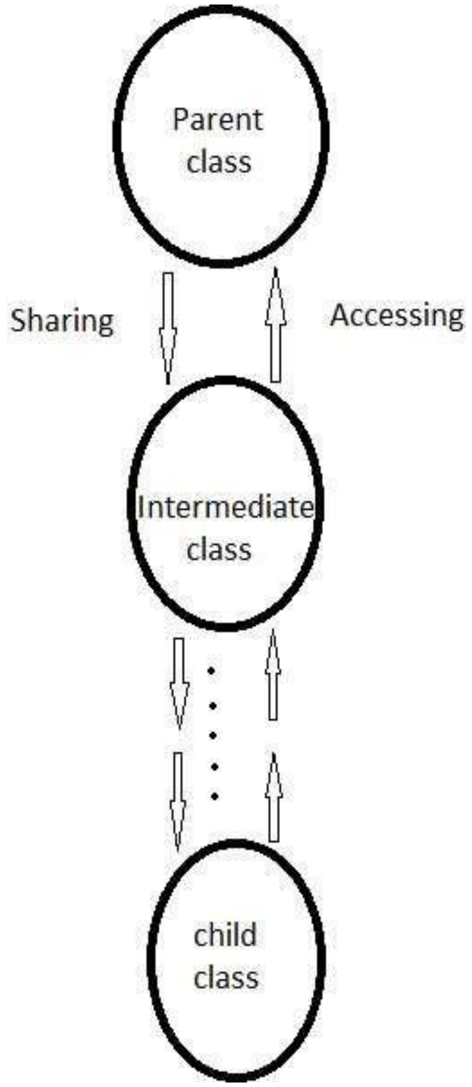
Intermediate class
(mediating/dual class).

Child class
(sub/associate/derived/inherited class).



```
class A
{
    ..................
    ..................
}
class B
{
    ..................
    ..................
}
class C
{
    ..................
    ..................
}
```

Relation between classes



Super class

- Top located class

- Service provider
  (its properties accessed by all its lower level class).

Intermediate class

- Middle located class

- Having Dual policy
  (obtain properties of upper level class and transmit properties to lower level class).

Child class

- Bottom located class
- much benefitted class
- much loaded class
- properties of child class as well as parent class can be accessed by only the object of child class.

TYPES of INHERITANCE

- Single Inheritance
- Multilevel Inheritance
- Hierarchical Inheritance

Single Inheritance

- A structure having one and only one parent as well as child class.

- Child class is authorized to access the property of Parent class.

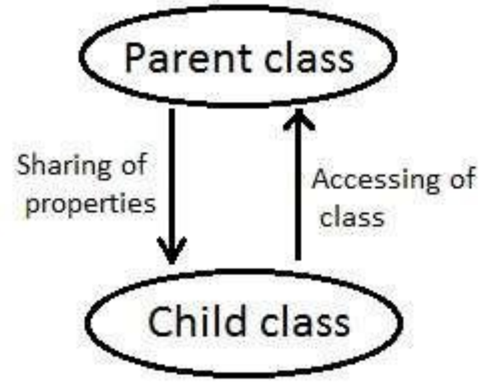
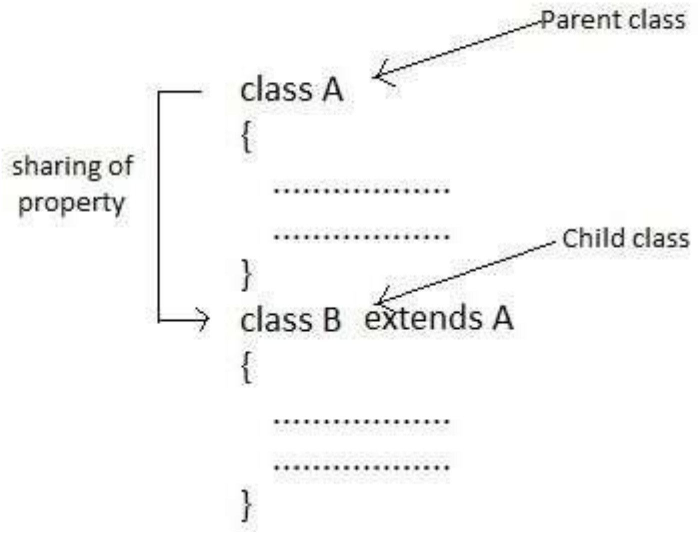


Fig : Single Inheritance

Syntax :



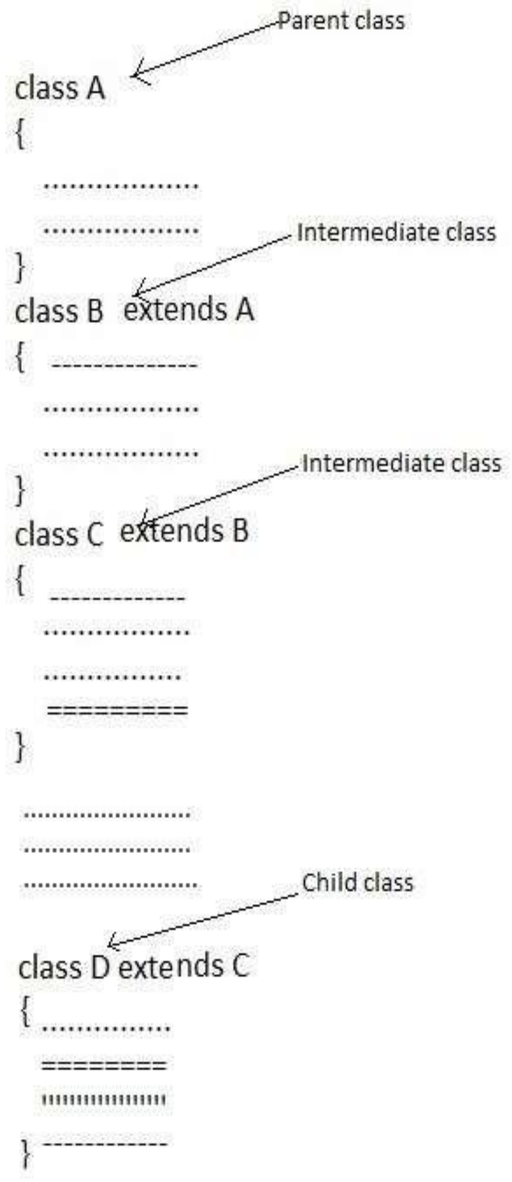
```
class A
{
   ..................
   ..................
}
class B  extends A
{
   ..................
   ..................
}
```

Multilevel Inheritance

- Standard structure of Single Inheritance having one Parent, one or more intermediate and one child classes.

- Child class as well as intermediate class may access the properties of upper level classes.

Syntax :



```
class A
{
   ..................
   ..................
}
class B  extends A
{  -------------
   ..................
   ..................
}
class C  extends B
{  ------------
   ..................
   ..................
   =========
}

 ........................
 ........................
 ........................

class D extends C
{ ...............
   ========
   ''''''''''''''''''
} -----------
```

Parent class
Intermediate class
Intermediate class
Child class

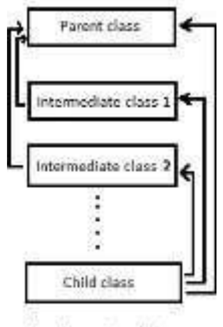


Fig : Accessing of class

Hierarchical Inheritance

- A structure having one parent and more child class.
- Child classes must be connected with only Parent class.

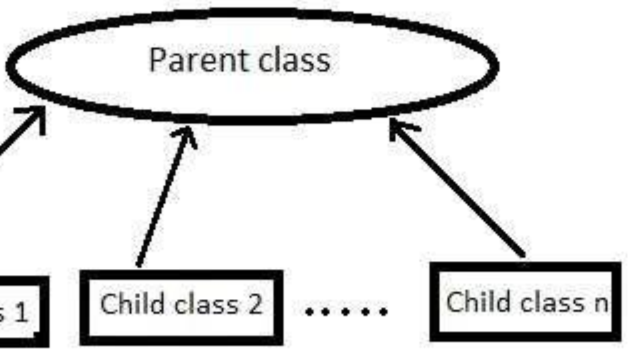
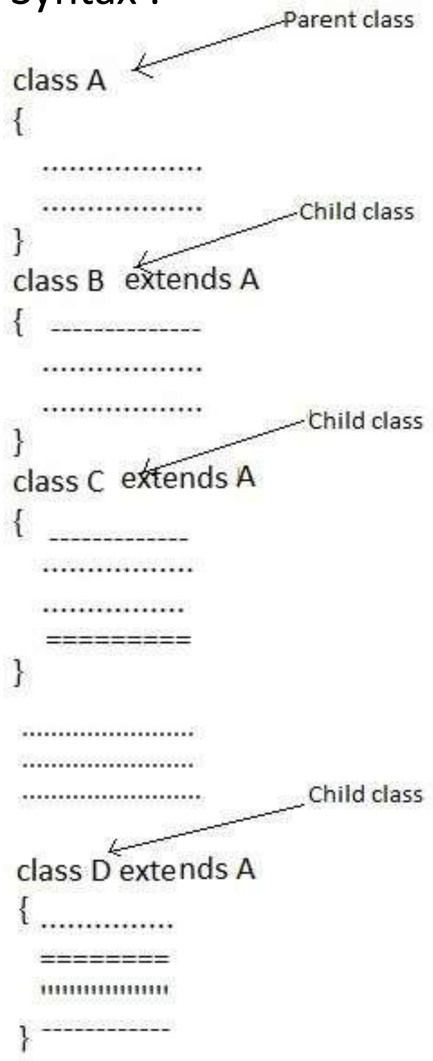


Fig : accessing of parent class by child classes

Syntax :

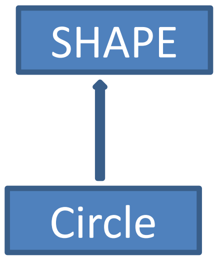

Parent class
Child class
Child class
Child class

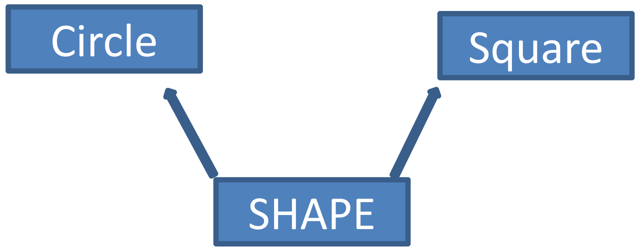
```
class A
{
  ..................
  ..................
}
class B  extends A
{  -------------
  ..................
  ..................
}
class C  extends A
{  -----------
  ................
  ...............
  =========
}

........................
........................
........................

class D extends A
{ ...............
  ========
  ''''''''''''''''
} -----------
```

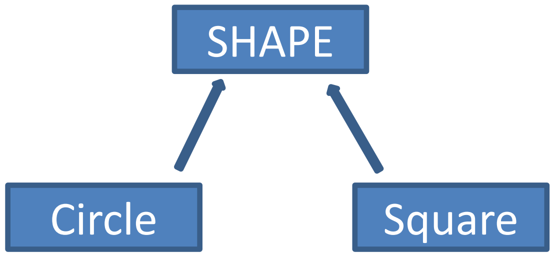
Example of single inheritance

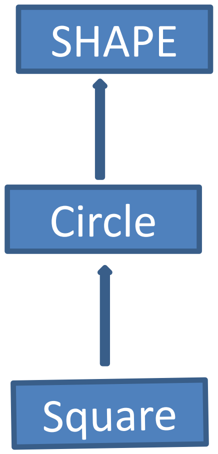
```
class Employee
{
float salary=40000;
}
class Programmer extends Employee
{
    int bonus=10000;
    public static void main(String args[])
    {
    Programmer p=new Programmer();
    System.out.println("Programmer salary is:"+p.salary);
    System.out.println("Bonus of Programmer is:"+p.bonus);
    }
}
```

Create a class shape and inherit the properties into child class for following









Indirect Mechanism of Inheritance

- Java Supports a special feature called interface.
- This feature helps to connect a class with more than one classes.
- For this type of connectivity java uses '***implements***' keyword.

```
Syntax :
interface A{
    ……..}
Interface B {
        }
class M {
        }
class N implements A,B extends M{
        }
```

Interface in Java

- An **interface in Java** is a blueprint of a class. It has static constants and abstract methods.
- The interface in Java is *a mechanism to achieve abstraction*.
- There can be only abstract methods in the Java interface, not method body. It is used to achieve abstraction and multiple inheritance in Java.
- In other words, you can say that interfaces can have abstract methods and variables. It cannot have a method body.

How to declare an interface?

An interface is declared by using the interface keyword. It provides total abstraction; means all the methods in an interface are declared with the empty body, and all the fields are public, static and final by default. A class that implements an interface must implement all the methods declared in the interface.

Syntax:

```
interface <interface_name>{
    // declare constant fields
    // declare methods that abstract


}
```

Example

```
interface printable
{
void print();
}
class A6 implements printable
{
      public void print()
      {
            System.out.println("Hello");
      }
public static void main(String args[])
{
A6 obj = new A6();
obj.print();
}
}
```

LIMITATIONS

- Link is establish into single direction(Fig).
- Java not support Multiple inheritance as well as Hybrid inheritance.
- The *extends* keyword permits to connect a class with only one class.

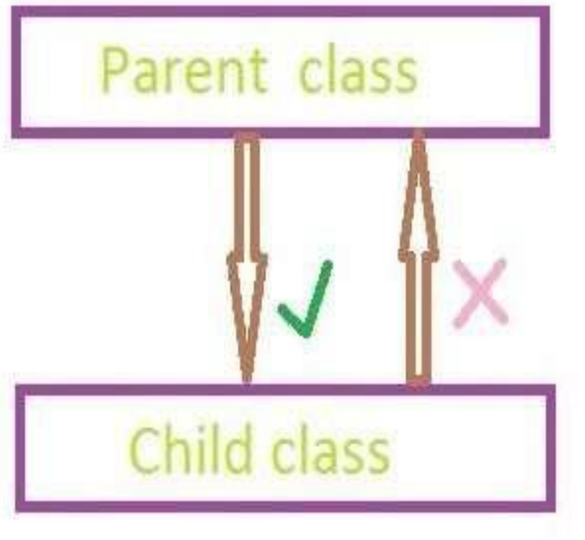


Fig : sharing of properties

- In Interface, properties are only declared and assigned, but never defined.

THANK YOU