classmate
Date
Page

* **Java Static Keyword:**

It belongs to class Than instance of The class.

(1) get memory only once in class and at The time of class loading.
(2) So memory efficient.
(3) Belongs to class rather Than object of class.
(5) Can access static data member & can change The value of it.

Restriction : (1) Static methods cannot use non-static data-member or call non-static method directly. int i; static void update() { i=101;} X as i is non static
(2) This and Super keyword cannot be used in static context.

• Static Block : (1) Used to initilize The static data member.
(2) Executed before main method at time of class loading.

* We can execute a program without main method ( ) , but before [JDK version 1.7 before]

* **Static & Dynamic Binding:**

An object of a class is created, and using That object we call a method inside a class. and This method call binds The method body inside class.
So binding means mapping ~~The~~ between method class & with method body.

void m1()
{ }
obj.m1();

```
class Parent
{
  Void method()
   { sop("Hello"); }
}
class Child extends Parent
{
  Void method()
   { sop("Hello"); }
}
```

Parent P = new Parent(); — thus at time of compilation compiler knows ~~that object created is of Parent~~ (Static Binding)

P.method();

Parent P1 = new Child(); — But in run time it knows that this object is of child. (Dynamic Binding)

P1.method();

- Static Binding takes place during compile time.
- Dynamic Binding takes place during Runtime.

* Abstract Class :

```
Abstract Class Parent
   {
        Abstract public void m1();

        public void m2()
          { sop("m2"); }
   }

   class Child extends Parent
{
        public void m1()
          { sop(m1); }
```

① An abstract ~~cas~~ class can have abstract as well as non abstract methods.

② If a method is ~~not~~ defined as abstract then it is compulsory to create the class as abstract.

③ An abstract methods in an abstract class has only declaration and no defination is present.

④ If a child class extends a abstract it is compulsory to override that abstract method present in the abstract class.

*⑤ We cannot create an object of an abstract class.

```
Parent obj; ✓
Parent obj = new Parent(); X
```

⑥ → inherit using 'extends' keyword.

* Interface

- Interface Parent

```
Interface Parent
        { void m1(); }
Class Child implements Parent.
        { void m1()
            { sop("done"); }
        }
```

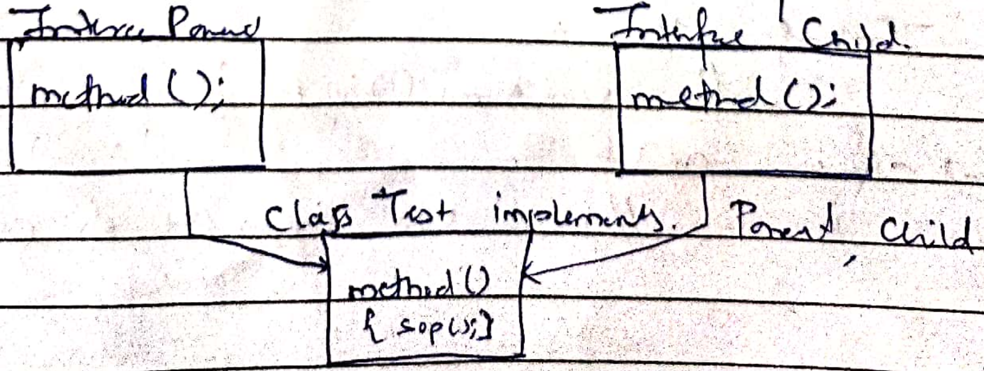
- Interface Parent

```
Interface Parent
        { void m1(); }
Abstract class Child
        { // no need to override as class is abstract }
```

1. Interface is a collection of abstract methods.
2. The methods in the interface will ~~not~~ have only the declarations but not the definitions
3. When the interface parent implements (extends) the child class then all the methods must be overrided inside the child class.
4. Here in-heritance is done using implement keyword.
5. If the child class which is implemented by the Interface parent is not abstract then we need to override all methods present in Interface Parent to the child class. ~~and~~ ~~if~~ ~~the~~
   if the child class Implemented by the parent class is an Abstract Class then there is no need of overriding the methods of the ~~parent~~ Interface Parent.
6. Methods inside the interface are always abstract, it is done by JVM.
7. No constructor of Interface ~~class~~ can be created
8. An interface can extend multiple interface
9. It cannot contain instance fields.
10. All the fields appearing are static & Final (~~methods, and~~ variables)
11. Can be used to ~~[illegible]~~ achieve multiple Inheritance.

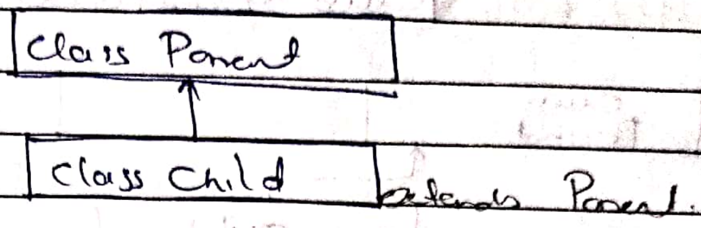
* There are same methods in both parent while deriving multiple Heritance, compiler gets confused or causes Ambiguity.

multiple Inheritance

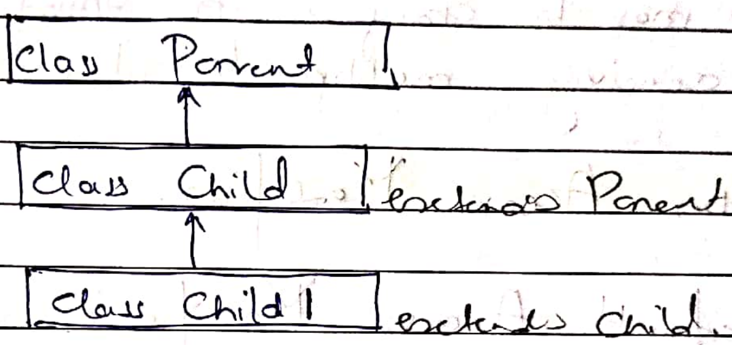


* Inheritance:

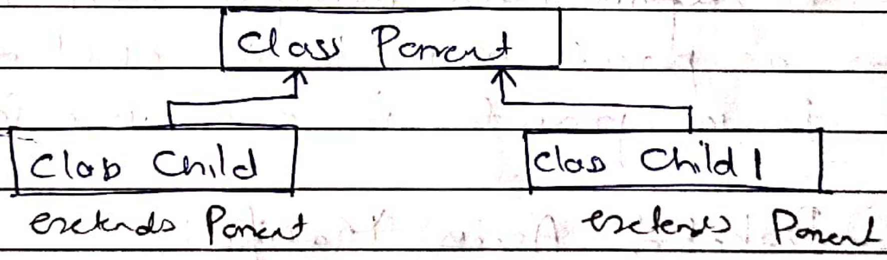
1. Single level Inheritance



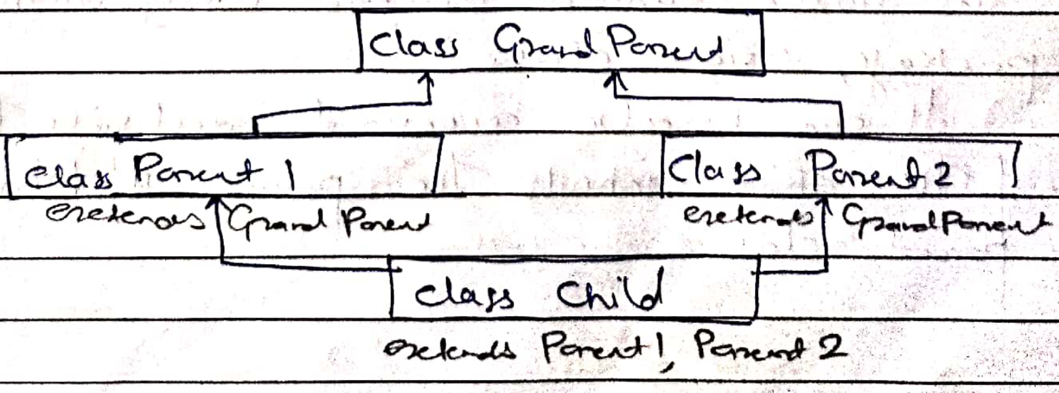
2. Multilevel Inheritance



3. Hierarchical Inheritance

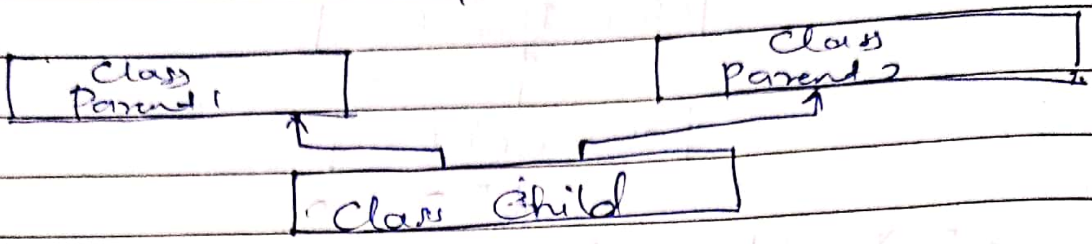


4. Hybrid Inheritance



⑤ Multiple Inheritance

(Mirazapur)



• Class must be changed to Interface instead of for achieving multiple Inheritance.

* Member Access Modifiers.

① Default Access Modifier :
e.g int i = 10; ← its default Acces

A default access modifier can be accessed from a parent class to its child class. The value of i in parent is accessible in the child.

② Protected Access Modifier :
e.g protected int i = 10; ← protected

This value can be accessed only within the package, inside class.
It can be also accessed outside the class if the ~~child inherits the parent~~ it inherits the parent class.

③ Public Access Modifier:
e.g. public int i = 10;
This value can be accessed outside the & inside class, outside the package ~~is~~

④ Private Access Modifier:
It can be accessed only within the class. & not outside the class.

* Garbage Collection in Java:

~~cla~~ Student s = Student();
Employee e = Employee();
s = null; } Thus this method we can achieve
e = null; } Garbage collection.

OR

s = e; As reference of 'e' is given to 's' 's' is having reference of e, so if we can say that s = e so 's' is of no use, hence garbage collected.

OR

By creating anonymous object : new A();
By creating anonymous objects like then it is after this execution of the particular line then it is done for garbage collection.

Methods:
(i) finalize(); : ~~It is called before garbage is~~ calling is done before object is garbage collected (It is present in object class so need to ~~create an object to~~ perform operation)

(ii) gc(); : Used to invoke garbage collector to perform the clean up processing