Object and class

Object is one of the OOPs concepts in java

Basically object do 2 things

**What object knows and what object do**

**What Object knows is variables**

**Variables: if you want to store a number or name or personal information you will store in variables.**

**What Object do is Methods:**

**If you want to print something**

**If you want to execute something**

**If you want to save something you required methods**

**Objects are created using class. That is interdependent.**

**When you want to create an object you required class,**

**When you want to work with class you required an object.**

**Object is an instance of a class.**

**Example:**

**Required a table**

**Table are made by carpenter**

**How will carpenter create a table he need design?**

**JVM will provide all objects**

**Objects are real instance and class are imaginary parts**

**Objects are having memory and memory provided by JVM.**

**Abstract Class and methods:**

**Abstract Method**

A method which is only declare not defined is called as Abstract methods.

Class mobile()

{

Public void camera()

**{**

**Know how it works then you can define this method**

}

Public void backlight()

{

**{**

**Know how it works then you can define this method**

}

~~Public~~ **Abstract** void front Light()

**{**

**Don’t Know how it works then you cannot define this method**

**Then we need to declare it as abstract method**

}

**Abstract Class**

1. **The class which has at least one Abstract method is called as Abstract Class.**
2. **We cannot create an object of abstract class since we don’t have any method definition. we don’t know how it works.**

In above mobile class we have one Abstract method, so we can declare that class as Abstract class like below,

Abstract Class mobile()

{

Public void camera()

**{**

**Know how it works then you can define this method**

}

Public void backlight()

{

**{**

**Know how it works then you can define this method**

}

~~Public~~ **Abstract** void front Light()

**{**

**Don’t Know how it works then you cannot define this method**

**Then we need to declare it as abstract method**

}

1. **If you want to define an abstract method in abstract class, then you have to take new class and extend new class with abstract class and define abstract methods in new class.**

Class new Mobile extends Mobile()

{

Should define all abstract methods in abstract class mobile

This class new Mobile is called concrete class.

If any chance you are not able to define all abstract methods in this new class means this class also become abstract class.

Then we have to define new class and that should extend new Mobile Class.

Under new class you have to define all your abstract method.

Now this new class called as concrete class

}.

**Interface**

**Suppose you have two Abstract classes**

**Abstract class A()**

**Abstract class B()**

**You want to define all abstract methods in A and B same new class C means we have to extend class A and B in Class C**

**Class C extends A,B()**

**This is called multiple inheritance . java will not support multiple inheritance.**

**In order to avoid this we will use Interface.**

**Abstract Class Interface**

**Declare methods Declare methods**

**Define some methods should not define any methods**

**We can define and declare methods we cannot define any method only declaration cannot create object of abstract class cannot create object of interface**

**Class C implements A,B()**

**{**

**Multiple interface.**

**Should define each and every method declared in interface A and B**

**}**

**Public interface A()**

**{**

**Public void show()**

**{**

**No need to define public as abstract, by default all methods under interface are abstract methods.**

**}**

**Encapsulation:**

Encapsulation is an OOP technique of wrapping the data and code. In this OOPS concept, the **variables of a class are always hidden from other classes**. It can only be accessed using the methods of their current class. For example - in school, a student cannot exist without a class.

Example:

Class A() Class B extends A()[

{ main()

Private int I; {

Public void set(int j) A obj=new A()

{ obj(5);

I=j;

{ System.out.println(Obj.getI());

Public int getI()

{

Return I;

}

**Exceptions:**

**Errors:**

**Syntax errors**

**Runtime error**

**Logical error getting wrong output**

**Multiple Inheritance:**

**Inheritance means where subclass is derived from another class**

**Java will not support multiple inheritance.**

**Polymorphism:**

**Poly means many**

**Morphism is called as behavior**

**One thing many behavior**

**Class A(){ A obj = new A();**

**Public void show() obj.show();// it will call show ()method**

**{ obj.show(5);// it will call show (int i)method**

**}**

**Public void show(int i)**

**{**

**}**

in above example we have 2 methods with same name but difference paramteres is called method overloading.

Same method behaving different ways called polymorphism.

Here one method is overloading other method.

**Class A(){ A obj = new A();**

**Public void show() obj.show();// it will call show ()method in class B**

**{**

**}**

**Class B extends A**

**Public void show()**

**{**

**}**

This is called method overriding

**Method Overloading:**

Same method name with different parameters

**Method Overridding**

Same method name with same parameters