Object Oriented Programming

Classes and Objects

Classes → group of these entities.

Objects → entities in real world

Syntax

```
class Class_name {

}
```

Object creation –

```
Class_name Obj_name
    = new Class_name();
```

```
public class filename {
    public static void main
}
```

Access Modifiers

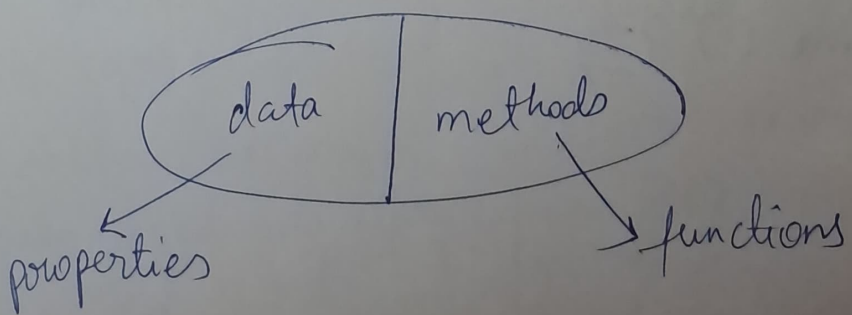
Classes can't be private or protected in JAVA

Access Modifier	within class	within package	outside package by subclass only	outside Package
Private	Y	N	N	N
Default	Y	Y	N	N
Protected	Y	Y	Y	N
Public	Y	Y	Y	Y

Learn for interview

#1 Encapsulation

Encapsulation is defined as the wrapping up of data & methods under a single unit. It is also called data hiding.



Constructors.

Constructor is a special method which is invoked automatically at the time of object creation.

Constructors —

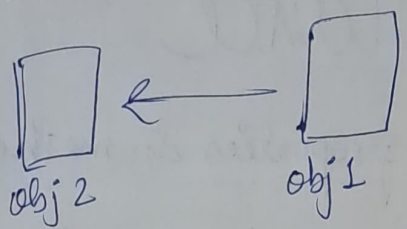
- have the same name as class
- don't have a return type
- only called once, at object creation.
- Memory allocations happens, when const. is called.

Syntax

```
class Cname {
    ---------
    ---------
    Cname () {=}
}
```

Types -

1) Non parameterized
2) Parameterized
3) Copy constructor



copies one object to another.

```
Student ( Student S1) {
    ____
    ____
    ____
}
```

Here student is a class

see example in VS code.

Two types

Shallow	Deep
* while copying, data structures like array are only copied via reference	* new array banake copy
* Changes reflect in copied one when done in original one	* changes don't reflect eg in VS code.

Destructors

We do not use destructor in JAVA because JAVA has garbage collector.

#2 Inheritance

Inheritance is when properties & methods of base class are passed on to derived class

Keyword
extends

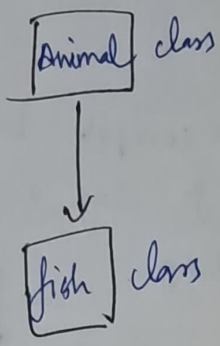
eg: Animal class banai
Ab banani hai fish class
then copy animal → fish

Syntax

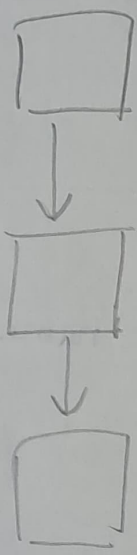
```
class Base {

}

class Derived extends Base {

}
```

Types of Inheritance

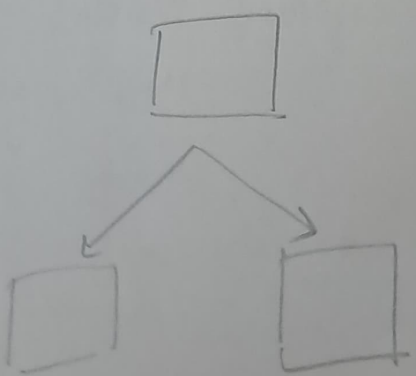
1) Single level inheritance



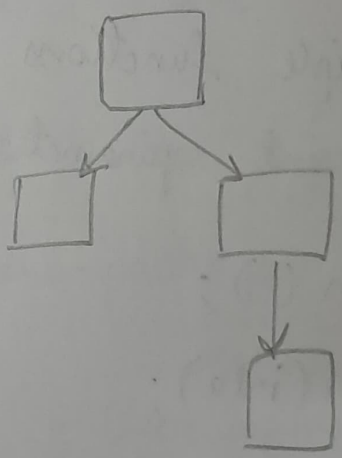
2) Multi level inheritance



3) Hierarchial



4) Hybrid inheritance



#3 Polymorphism

mary form

- Compile time (static)
  Overloading

compile krne ke time pe jo alag alag forms dikhenge

- Runtime (dynamic)
  Over riding

✡ Overloading :-

Multiple functions with same name but different parameters.

```
sum ( );
sum (int a);
sum ( float a);
```

Overriding –

Parent and Child classes ~~have~~ both contain the same function with a different work. (definition).

In such a case,

function of <u>child class</u> is preferred

i.e. gets to work.

Packages in Java

Package is a group of similar types of classes, interfaces and sub-packages.

- Inbuilt packages
  eg: util
  from
  java.util.*

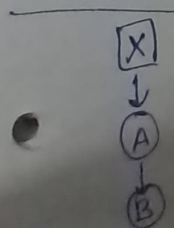
- User defined pack--

#4 Abstraction

Hiding all the unnecessary details and showing only the important parts to the user.

Abstract Class – Keyword abstract

* Cannot create an instance (object) of abstract class.
* Can have abstract / non-abstract methods
* Can have constructors.
* Abstract methods (functions) are compulsory to be implemented in derived class



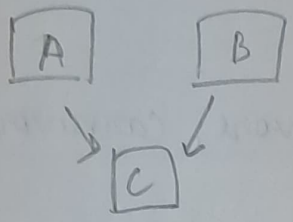
If we make obj of B then first X const the A const & then B constructor is called.

eg in VS

Interfaces

Interface is a blueprint of a class.

* This is same as multiple inheritance of C++.



Keyword
Interface
↓
implements

* Requires 100% Abstraction.

Interface -

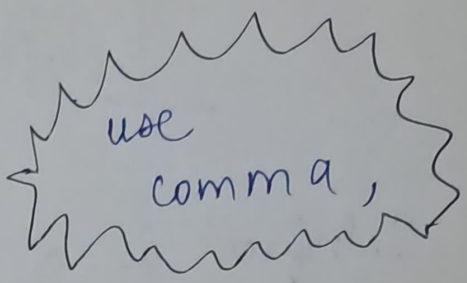
* All methods are public, abstract & without implementation.
* Used to achieve total abstraction.
* Variables in interface are final, public and static.

eg: Pieces
Chess ~~Player~~
↳ moves
Queen
Rook
Pawn
king

implementation in VS

eg:

```
interface herbivore{
}
interface carnivore{
}
class bear implements herbivore, carnivore{
}
```



Static Keyword

Static keyword in java is used to share the same variable or method (fn) of a given class.

(Share between objects)

things that can be made static →

- properties (variables)
- functions
- Blocks
- Nested classes.

this ~~keyword~~ pointer.

To object variable ya func ko call kr raha usse point krne ke liye.

Super keyword. super();

is used to refer immediate parent class object.

- to access parent's
  - properties
  - functions
  - constructors.

Reference variable explanation via example.

(1)

```
class Vehicle {
    void print() {
        "Base Class (Vehicle)";
    }

class Car extends Vehicle {
    void print() {
        "Derived Class (Car)";
    }

ps void main(__) {
    Vehicle obj1 = new Car();
    obj1.print();        ——> "Derived Class (Car)"

    Vehicle obj2 = new Vehicle();
    obj2.print();        ——> "Base Class (Vehicle)"
}
```

This is reference variable
This determines capacity of object
what it can access
But here obj1 is calling Car print due to fn overriding otherwise it can't.