OOPS Concept

OOP :→ Refers to languages that use objects in programming.

→ Aim: to bind the data and the functions that operate on them so that no other part of code can access this data except that function.

- Few points →

In java, every .java file can have only 1 public class. Also, the name of that public class should be same as filename.

CLASS :

1) Its like a blueprint of an object
2) It is a user-defined data-type.
3) Consists of data members & functions, which can be accessed by creating instance of that class
4) It represents set of properties & methods common to all objects of one type.

Eg:

```
Class Person {
    String name;
    int age;
    void walk() {
    }
    void eat() {
    }
}
```

properties

behaviors

OBJECT :

1) Its an instance of a class
2) When a class is defined, no memory is allocated but when it is instantiated, i.e object is created then memory is allocated.
3) It has identity, state, behavior
4) Each object contains data and code to manipulate the data.

Constructor →

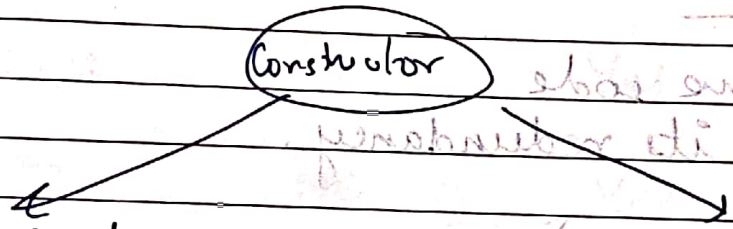
→ Used to initialize an object's state
→ Each time we use new keyword, atleast one constructor (it could be default constructor) is invoked to assign initial values to data members of same class.
→ Consists of collection of instructions that are executed at the time of object creation.

- No return type or value
- [Public] [same name as class]

⇒ We can call 1 constructor from another constructor.

Eg :→ class Person {
String name;
int age;
static int count;

```
public Person() {
    count++;  // Counts total objects created
}

public Person(int age, String name) {
    this();  // Calls the other constructor
    this.name = name;
    this.age = age;
}
}
```



→ A constructor that has parameters is a parameterized Constructor.

→ If we want to initialize fields of the class with your own values, then use parameterized constructor.

⇒ A constructor that has no parameter is called default constructor

⇒ If we don't define a constructor, then the compiler creates default constructor with no arguments for the class. But if we write a constructor, default constructor is not created.

→ Constructor can only be called one time, while methods can be called many times.

INHERITANCE :-

1) Capabilities of a class to derive properties and characteristics from another class is called inheritance.
2) So when we create a class, we do not need to write all properties & functions again and again, instead these can be inherited from other classes that possess it.

Benefits :-
⇒ Reuse the code
⇒ Reduce its redundancy.

Eg:
```
class Person {
    String name;
    int age;

    public Person (String name, int age) {
        this.name = name;
        this.age = age
    }
}

class Developer extend Person {
    public Developer (int age, String name) {
        super (age, name);
    }
}
```

ENCAPSULATION:

Access modifiers → Restrict the scope of a class, constructor, variable, method, data member.

1) Default → no keyword required
2) Private
3) Protected
4) Public

	Default	Private	Protected	Public
Same class	Yes	Yes	Yes	Yes
Same package sub class	Yes	NO	Yes	Yes
Same package non-sub class	Yes	NO	Yes	Yes
Different package sub class	No	NO	Yes	Yes
Different package non-sub class	No	No	No	Yes.

⟹ Encapsulation is defined as wrapping up of data in a single unit.

→ The variables or data of a class are hidden from any other class and can be accessed only through any member function of their class in which they are declared.

→ As the data in a class is hidden from other class it is also called data hiding.

ABSTRACTION:

1) Used to provide only essential information about the data to the outside world, hiding the background details or implementation.

Eg:
```
abstract class Car {
    int price;
    abstract void start();
}

class Audi extends Car {

    @Override
    void start(){
        System.out.println("Audi started");
    }
}

class BMW extends Car {

    @Override
    void start() {
        System.out.println("BMW start");
    }
}
```

Two ways to achieve abstraction:→

1. Abstract class
2. Interface

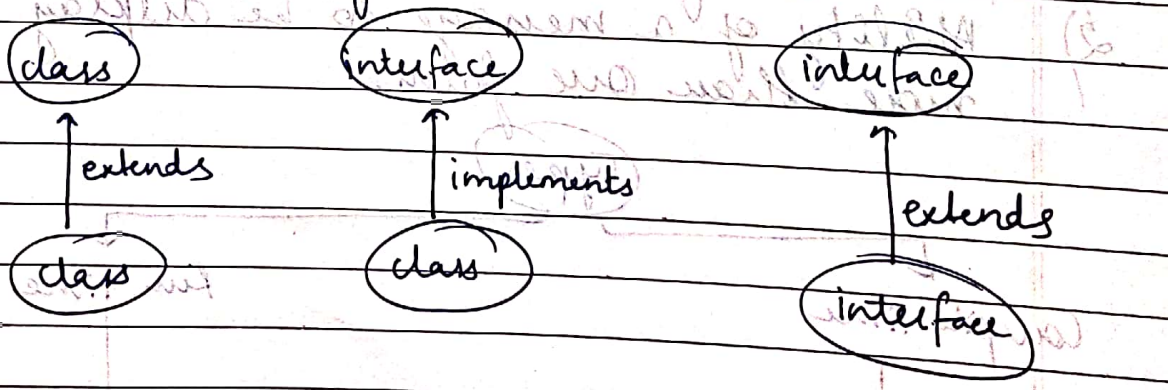
Abstract class:

1) Class declared as abstract is known as abstract class.
2) Can have abstract & non-abstract methods.
3) Cannot be instantiated.
4) It has to be extended and its method need to be implemented.

(Non-abstract methods can have a body, abstract can't)

Interface

1) It is the blueprint of a class.
2) It has static constants & abstract methods (only)
3) There can be only abstract methods in the interface not the method body.
4) Used to achieve abstraction & multiple inheritance in Java.

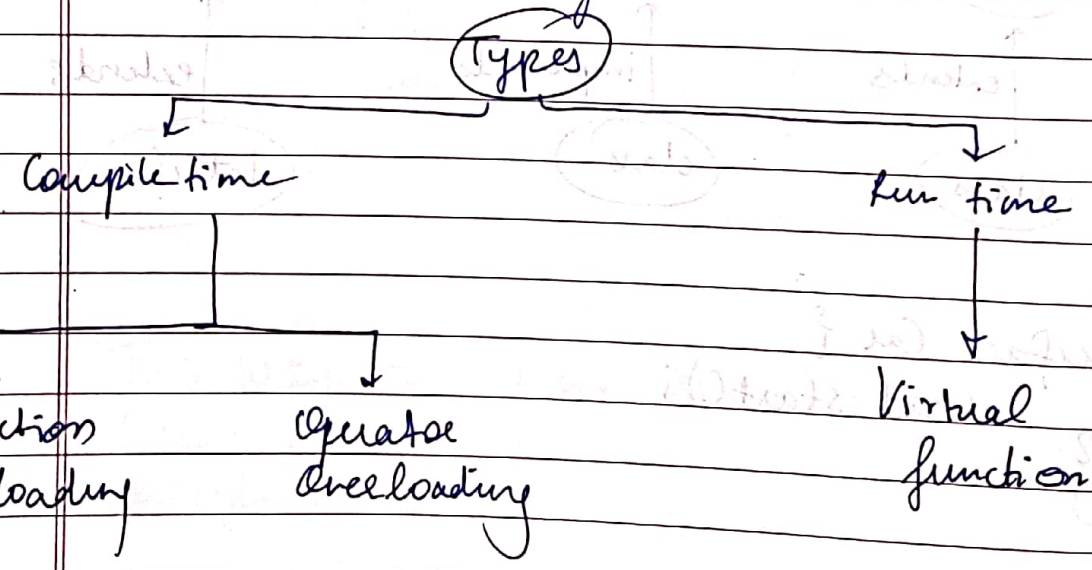


Eg:
```
interface Car {
    void start();
}
```

```
interface Person {
    void walk();
}

class Transformer implements Car, Person {

    @Override
    public void start() {

    }

    @Override
    public void walk() {

    }
}
```

Polymorphism:

1) Means having more than one form.
2) Ability of a message to be display in more than one form.



Eg:
```
class MultiplyNum{

    Static int Multiply(int a, int b)
    {
        return a*b;
    }

    Static double Multiply(double a, double b)
    {
        return a*b;
    }


    Static int Multiply(int a, int b, int c)
    {
        return a*b*c;
    }
}
```