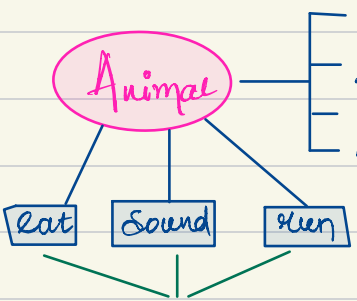
Object Oriented Programming (OOPs)

* Classes and objects
  - Entities in the real world
  - Group of these entities

an Animal Can be object
a Watch Can be object
a Pen Can be object



These are called instances of a class.

These are called method.

* Classes are acts like blueprint
* Write once, use as many times as we want to create multiple number of objects.

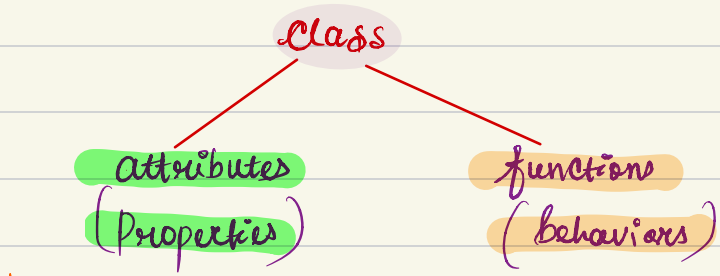
For example

Let's say a Car manufacturer wants to create a Car called "Lambahri"

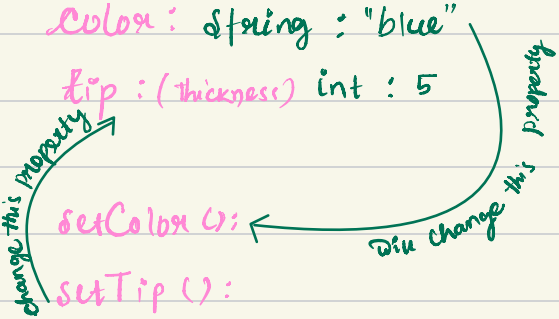
→ But in future, he/she wants to manufacture multiple Car, that time "Classes" comes handy.

→ Because using that same blueprint of class, they can manufacture different so called "Objects".

So,



Let's take an another example:-



Let's write the Code for pen class:

```
class   Pen {
    String color = "black" // default Value
    int   tip  = 5 // default Value

    Void   setColor (String colors){
    }          color  =  colors;
    Void   setTip ( int  tipsize) {
    }          tip  =  tipsize;
}
```

```
Public class main {
    pubstic static Void main (String[] args) {
        Pen  elros = new Pen (); // One object Created
        System.out.Println (elros. colon) //defauct colon
        elros. setColon ("Red");
        System. out. Println (elros. colon) // Print Red.
        elros. setTip (7);
        System. out. println (elros. tip) // Print 7.
    }
}
```

✱ Access modifiers

→ Make restrict to some properties or behaviour of a class. So that it can give access to a certain level of point.

Access modifiers	Within class	Within package	outside package by subclass only	Outside package
Public	Yes	No	No	No
default	Yes	Yes	No	No
Private	Yes	Yes	Yes	No
Protected	Yes	Yes	Yes	Yes

Use Cases

Let's say there was class called - "BankAccount"

→ It holds some properties like } Ac. Number
IFSC Code
Ac. holder name
ATM debit Card no
ATM pin

→ Here, we don't want to share "ATM pin" to anyone, so we will Make the Access level of this properties to "Private"

For example :

```
class pen {

    Public String userName;
    Private int password;

    //Class definition goes here

}
```

Getters and Setters

Get : To return the value (getters)
Set : To modify the value (Setters)

this : this keyword is used to refer to the Current object

In OOPS, there are mainly four pillars that Support OOPS Concept.

→ OOPS Can't happen without one of this

+ These are

1- encapsulation
2- Polymerphism
3- abstraction
4- Inheritance

Encapsulation

* encapsulation is defined as the wrapping up of data & methods under a single unit. It also implements data hiding

* encapsulation can be done using Access Specifiers.

Constructors

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

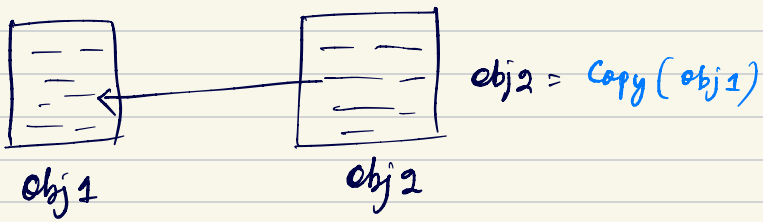
  * Constructor have the same name as class or structure.
  * Constructor don't have a return type (Not even void)
  * Constructor are only called once, at object creation.
  * Memory Allocation happens when Constructor is called.

* Types of Constructor

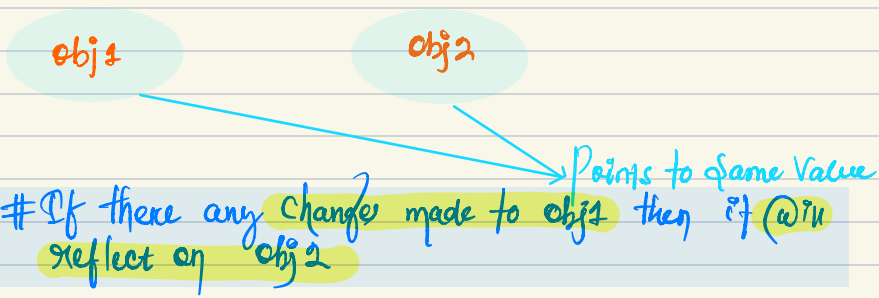
  * Non-parameterized
  * Parameterized
  * Copy Constructor

```
class Student {
    String name;
    int roll;

    Student() {
        System.out.println("Constructor is Called ---");
    }

    Student(String name, int roll) {
        this.name = name;
        this.roll = roll;
    }
}
```

Copy Constructors




```
// Copy Constructor.   → object of Student Class.
Student (Student S1) {
        this.name = S1.name;
        this.age = S1.age;
}
```

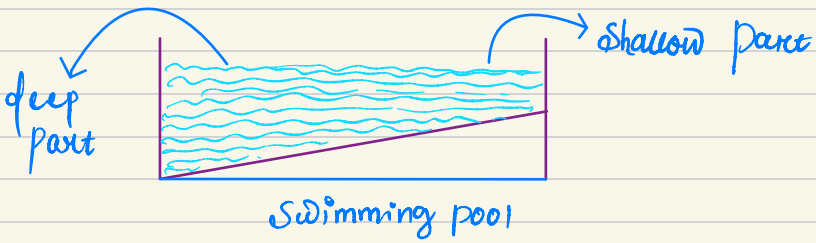


Date - 19.jun.2024

Shallow and Deep Copy

↳ references

↳ New fully modified object.



for example :

```
// Shallow Copy Constructor

Student (Student s1) {
    int[] marks = new int[3];
    this.name = s1.name
    this.roll = s1.roll
    this.marks = s1.marks;
}

//Deep Copy Constructor

Student (Student s2) {

    marks = new int[3];
    this.name = s2.name;
    this.roll = s2.roll;
    for (int i=0; i<marks.length; i++) {
        this.marks[i] = s2.marks[i];
    }
}
```

Destructor (Opposite of Constructor)

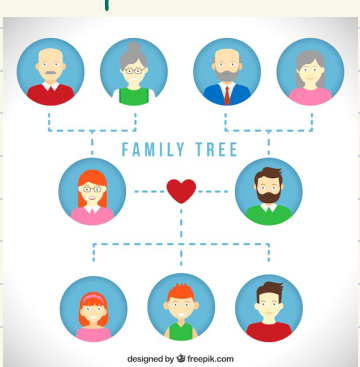
In java there is a feature available called "Garbage Collector"

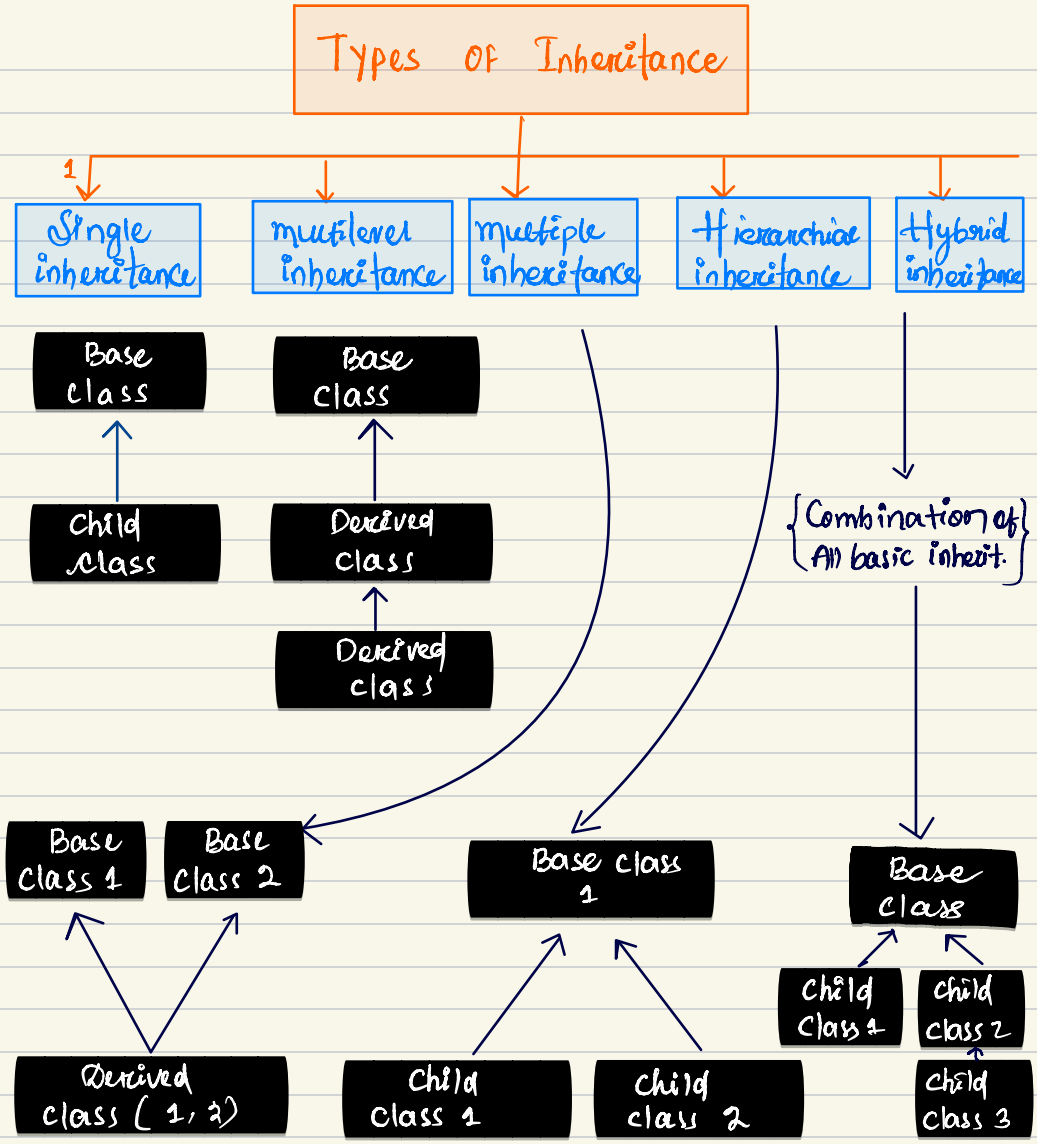
→ Garbage Collector ensures that All the Variable and objects that are not used in further future should be free from memory.

→ So in java we don't have to write "destructor" for garbage Collection, java automatically do it.

Inheritance

Inheritance is when properties and method of base class are passed on to a derived class.





* Java doesn't support multiple inheritance
→ But We can implement it indirectly using Interface

* Polymorphism

Poly → (many)

morphism (forms)

{ similar method can work on different number of parameters

* Compile time polymorphism

Method overloading

(Inside same class, same method name, different parameters, return type not matters)

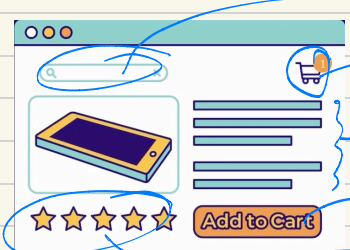
* Run time polymorphism

Method Overriding

(Difference class in inheritance : same method name, same return type, same parameters)

Packages in Java

*→ Packages is a group of similar type of classes, interfaces and sub packages.



Search → type, paste, delete

Cart → delete, add

Product description → change / modify (private)

Cart → Add, delete, buy / checkout

rating → give rating(s):
rating() [1-5]
delete, edit

Abstraction

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

Abstract classes

- Can't create an instance of abstract class
- Abstract keyword used for abstract classes
- Can have abstract - non abstract methods
- Used extends keyword for inhit abstract
- Can have Constructors

Interfaces

- Interface is a blueprint of class
- Interface keyword used for interface class
- Interface gives 100% abstraction while abstraction gives 0 - 100% abstraction it depends
- All methods are public, abstract and without implementation
- Used to Achieve total abstraction
- Variable in interface are final, public and static
- Used implements keyword for inherit interfaces.

Static Keyword

→ Static keyword in Java is used to share the same variable or method of a given class.

- Properties
- Methods
- blocks
- Nested classes

} can be static

Super Keyword

→ Super keyword is used to refer immediate parent class object.

- To Access parent's properties
- To Access parent's functions
- To Access parent's Constructor.