

---

---

---

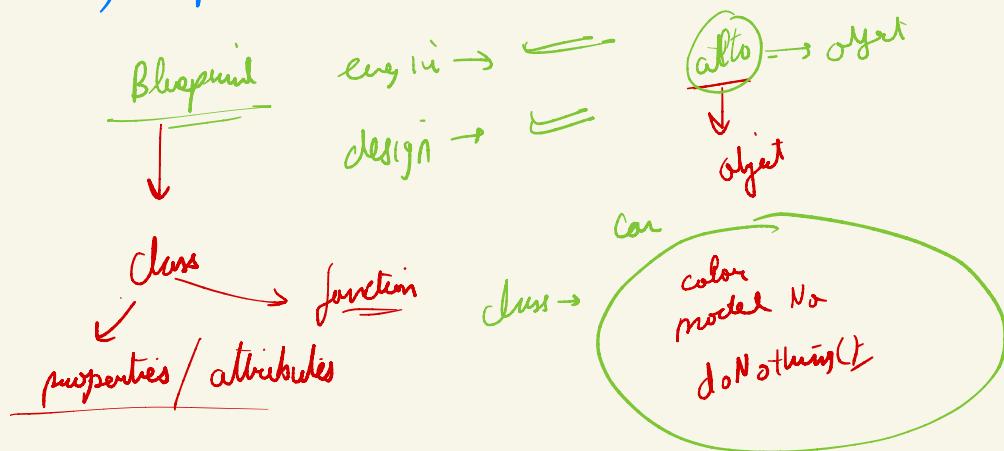
---

---



# # OOP ↳ Object Oriented Programming.

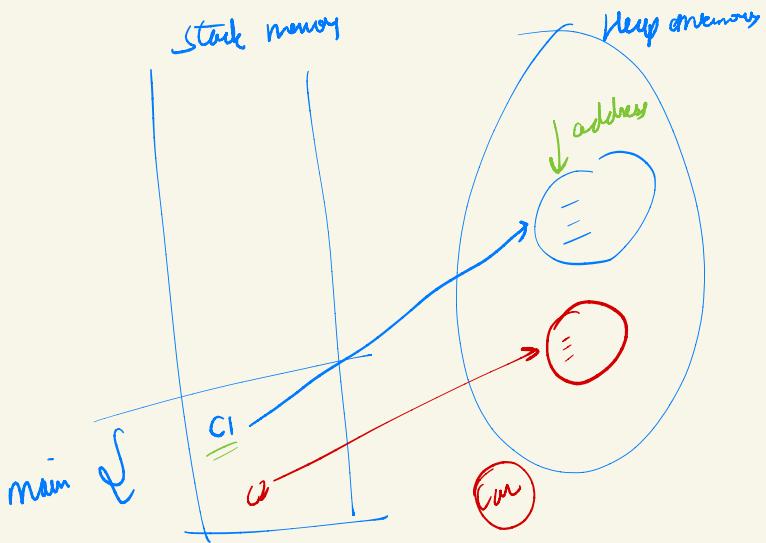
car making company → Hyundai / Honda / BMW



# Syntax      class Car {      Car c1

# Object Syntax

class Name {  
 objectName = new Classname();  
 objectName = new Classname();  
 ↓  
 address  
 C1.  
 # function { property access }



Ques) Create a class Student  
 give attributes → name, roll\_no, age  
 give function → print "anything"  
 # default values

# Access Specifiers  
 anyone can access (even outside the class)

1) public:

2) private

no one from  
outside can access

3) protected → only sub  
class can access.

4) default

↓  
same package

# Constructor

↳ function inside your class with name same as that of class and no return type.

Student s1 = new Student();  
↳ constructor call

public Student (int f) {

}

# Pillars of OOPs

⇒ Encapsulation

⇒ Abstraction

⇒ Inheritance

⇒ Polymorphism

# Encapsulation ↳ enclosing something in a capsule.

→ It can be achieved by using "private".

```

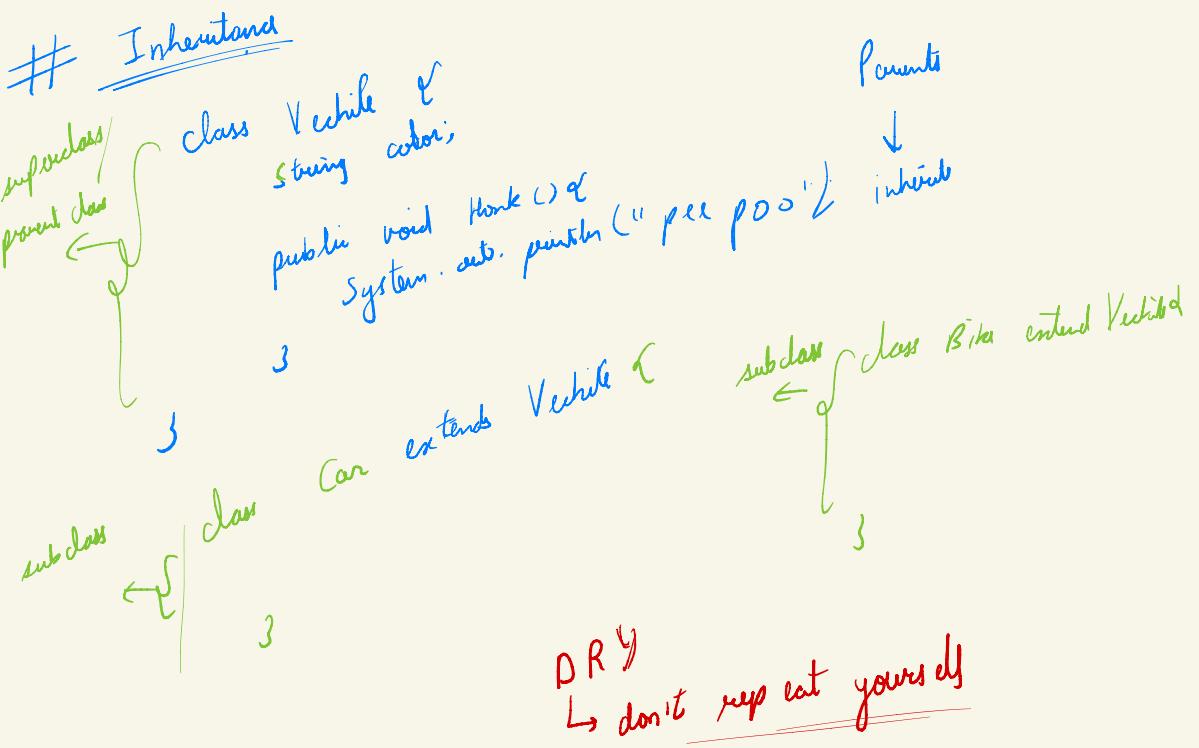
class Student {
    private String name;
    int age;
}

```

```

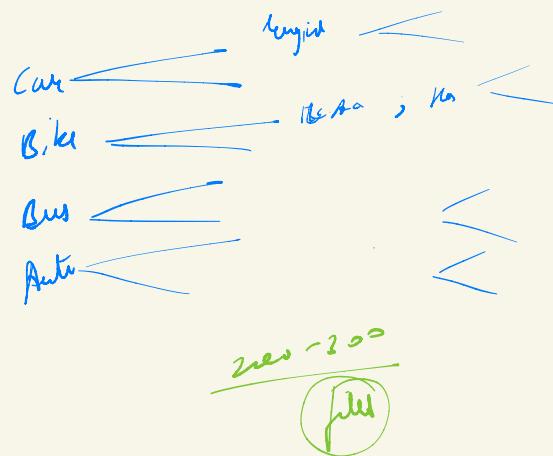
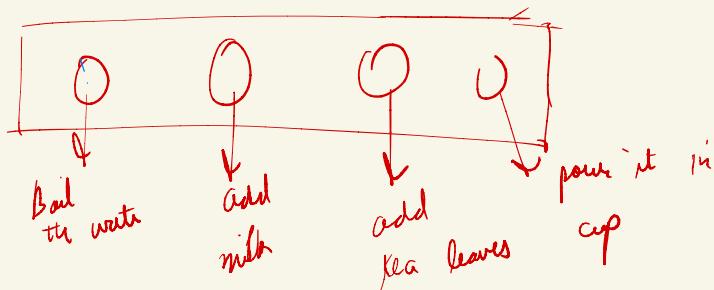
class DOL {
    public main() {
        Student s = new Student();
        s.name = "John";
    }
}

```



# Absstraction ↳ hiding implementation

Mocha Tea



# Interface ↳ can be used to achieve abstraction -

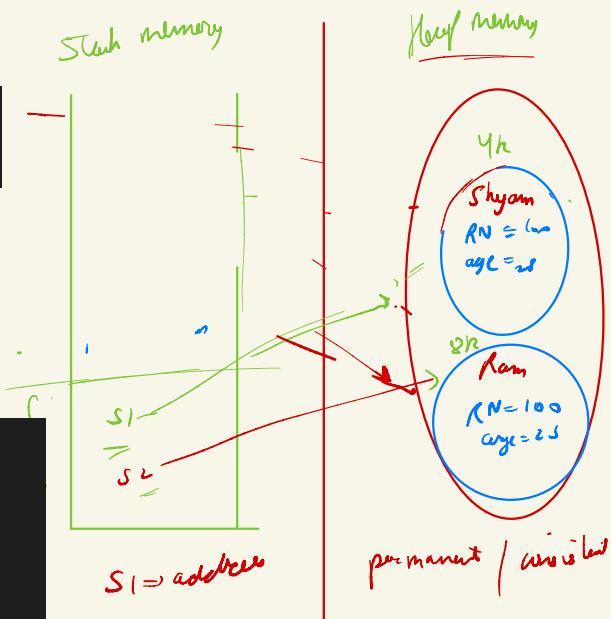
Java

## # Memory mapping

```
class Student {  
    private String name="Samrat";  
    int roll_num=100;  
    int age=25;
```

```
public static void swapNames(Student s1, Student s2){  
    String nameOfS1=s1.getName();  
    String nameOfS2=s2.getName();  
  
    s1.setName(nameOfS2);  
    s2.setName(nameOfS1);  
}
```

```
run| debug  
public static void main(String[] args) {  
    // className objectName = "new" + "ClassName" + "();"  
  
    ✓ Student s1=new Student();  
    ✓ Student s2=new Student();  
  
    ✓ s1.setName(newName: "ram");  
    s2.setName(newName: "shyam");  
  
    ✓ swapNames(s1,s2);  
    ✓ s1.getName();  
    ✓ s2.getName();  
    // Student s3=new Student(25);
```



## # Inheritance

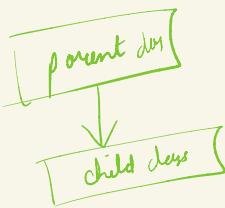
class Parent {  
 int id;  
 String color;

↓  
class A extends Parent {

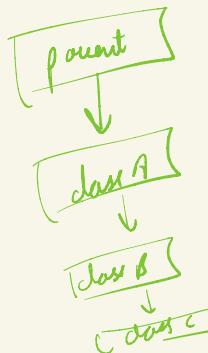
}

this, super

## \* Single Inheritance

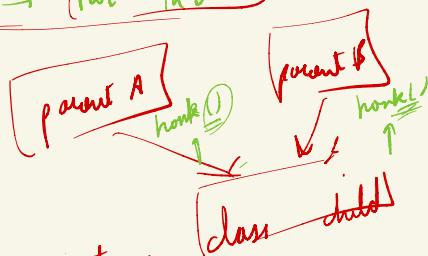


## \* Multi-level Inheritance

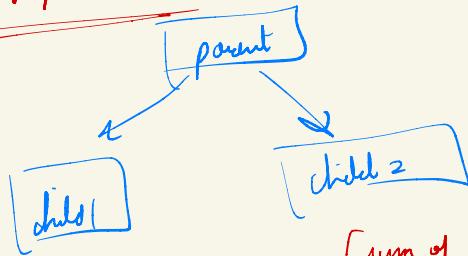


to reduce complexity  
to reduce ambiguity()

## # Multiple Inheritance → [not in Java]

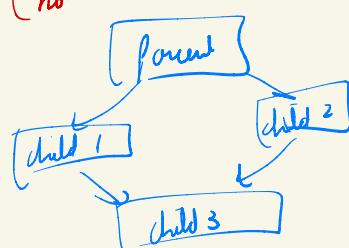


## # Hierarchical Inheritance



[sum of Hierarchical and ~~multiple~~]

## # Hybrid Inheritance



Ques

Create one multi level inheritance.

Create default constructors for each class.

Create the instance for lower most class.

# static keyword

↳ for those attributes, whose value will stay the same irrespective of the object.

→ Value for that attribute will be the same for all objects.

Ques

Create one class "Car". Then create some Car objects and keep a count of how many objects you created.

#

Constructor and constructor chaining → a constructor is called from a different constructor

↳ default constructor  
↓  
no parameters

→ parameterized constructor  
(parameters)

# Types of Constructor Chaining

- 1) same class
- 2) from other class.

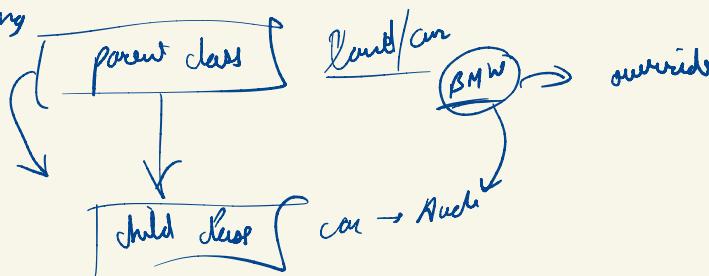
- ⇒ Rules for constructor chaining
- constructor calling should be first line.
  - order doesn't matter.
  - there should be at least one constructor without any chain call.
- 

## # Poly morphism

many forms  
many forms

- 1) Method overloading → same method different parameters
  - change in data type
  - change in number of parameters

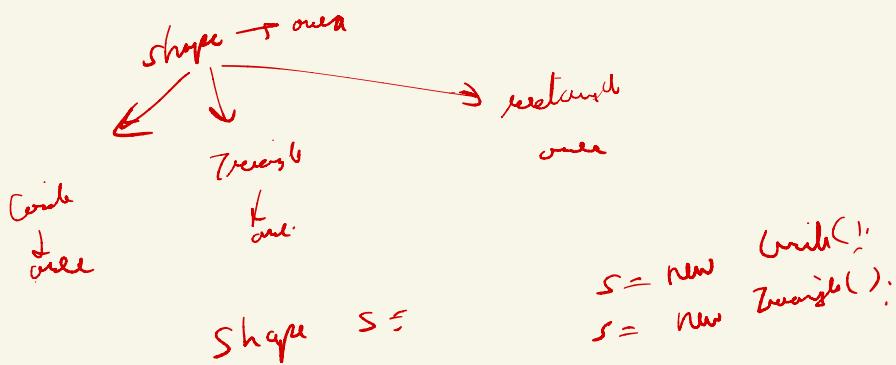
- 2) Method overriding



## Poly morphism

Compile time polymorphism  
↳ method overloading

Run time polymorphism  
↳ method overriding



# Static Binding (Early Binding)

- ↳ type of object is determined at compile time.
- (private, static, final)

# Dynamic Binding (late binding)

- ↳ type of object is determined at run time.

# instance Of

Ques Create ATM with functions.

1) Withdraw cash (with pin)

↳ If no cash available, "no cash available"

↳ limit exceeded ↳ wrong pin

2) deposit cash ↳ wrong account number

3) Bank balance

4) Change pin