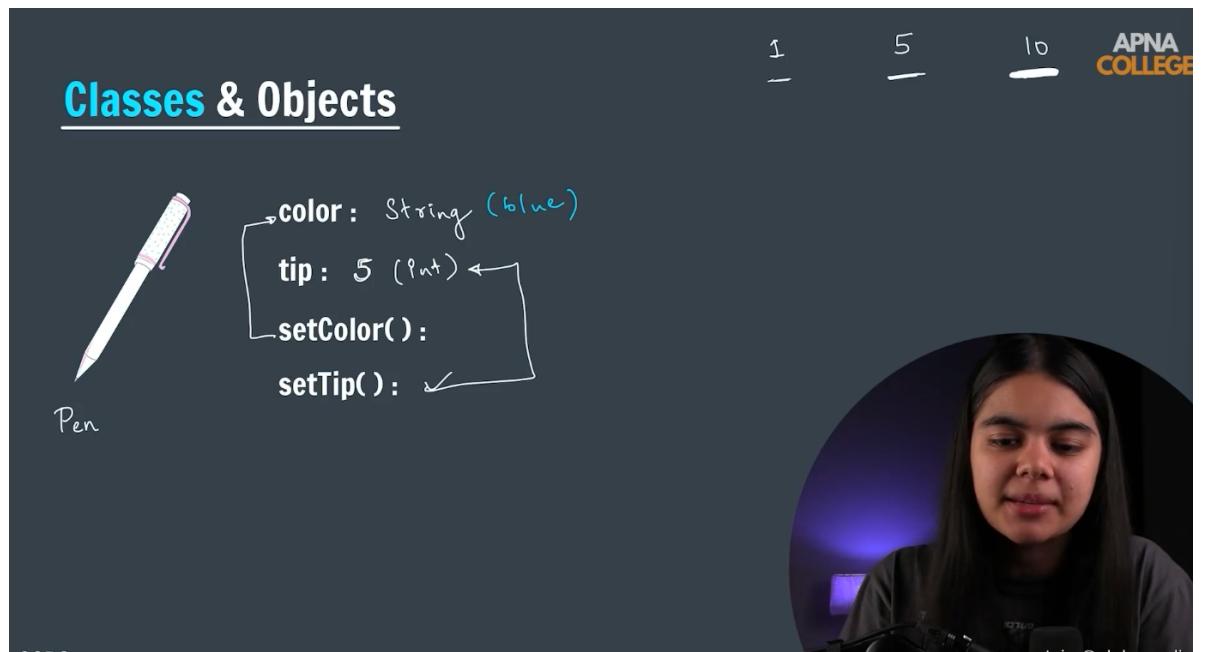


OOPS

▼ Objects :

- Objects are entities in the real world.
- Objects can have its own properties and functions.
- Class is a blueprint of an object defining the functions and properties of the objects.
 - Attributes : e.g. color of a pen
 - Functions : Change the color of the object



▼ Access Modifiers :

Access Modifiers

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	

▼ Getters and Setters :

- Get : to return the value
- Set : to modify the value

Getters & Setters

Object → prop

Pen → color
tip { private }

AP
COL

Get : to return the value

Getters

Set : to modify the value

Setters

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



```
public class OOPS {
    Run | Debug
    public static void main(String args[]) {
        Pen p1 = new Pen(); //created a pen object called p1
        p1.setColor("Blue");
        System.out.println(p1.getColor());
        p1.setTip(5);
        System.out.println(p1.getTip());
        // p1.setColor("Yellow");
        p1.setColor("Yellow");
        System.out.println(p1.getColor());
    }
}
```

PROBLEMS 8 OUTPUT TERMINAL DEBUG CONSOLE

```
adhakapra@Shradhas-Air Classroom Codes % java OOPS.java
Blue
adhakapra@Shradhas-Air Classroom Codes %
```



this.color , here this is used to refer the current object. Mainly used when there are multiple properties with the same name.

▼ Concepts :

▼ Encapsulation :

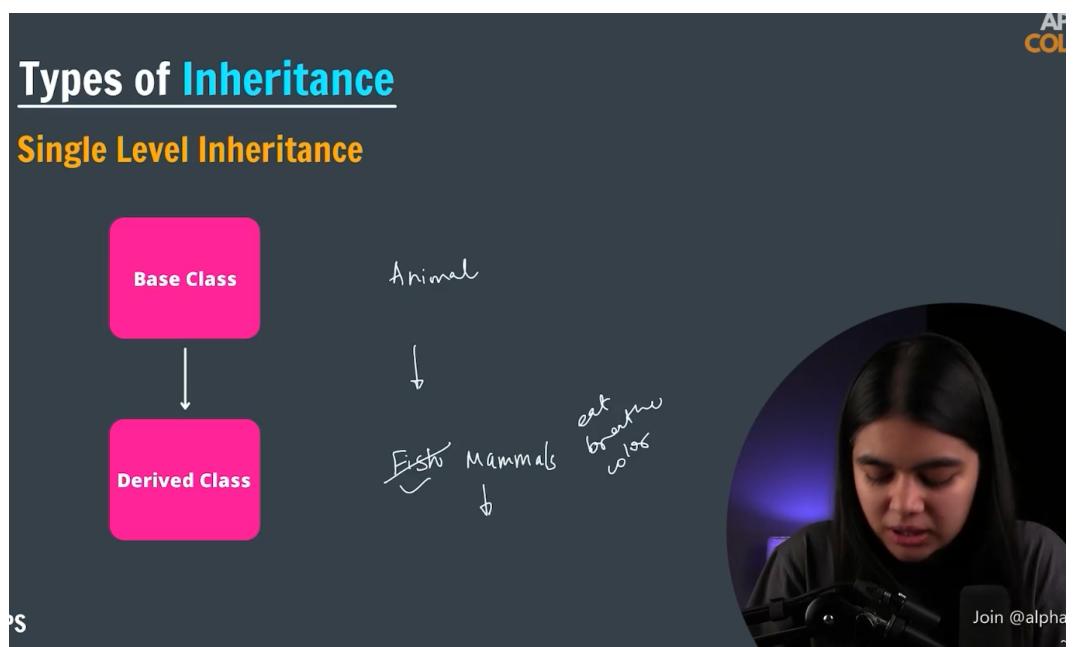
- Encapsulation is defined as the wrapping up of data (properties) & methods (function) under a single unit. It also implements data (sensitive data or Useless data) hiding .

▼ Inheritance :

- Inheritance is when properties & methods of base class (parent class) are passed on to a derived class (child class).
- ‘extends’ keyword is used to establish inheritance connection

▼ Types of Inheritance

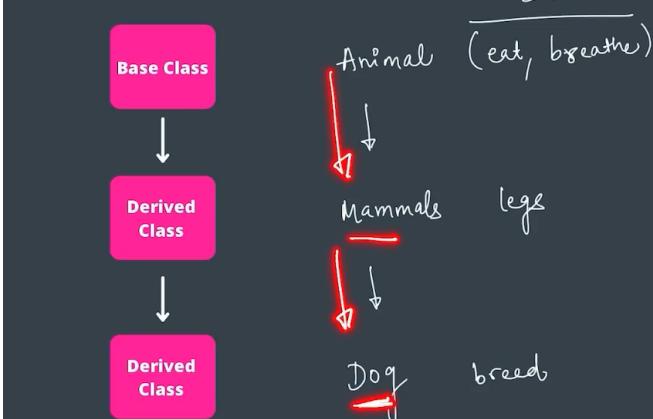
- Single Level Inheritance



- Multi-level Inheritance : More than one inheritance class can inherit from base class

Types of Inheritance

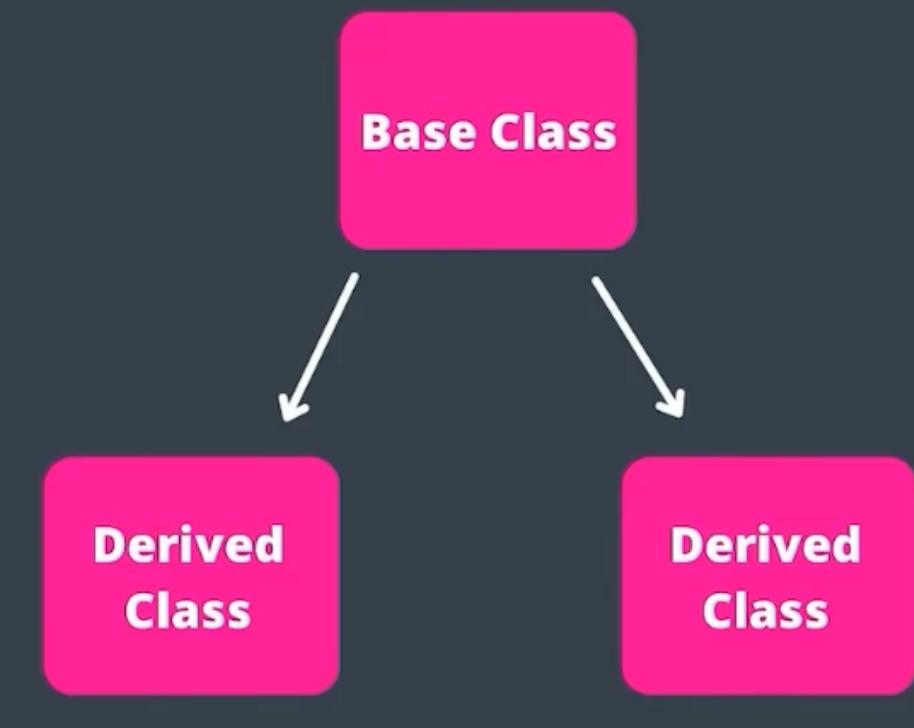
Multi Level Inheritance



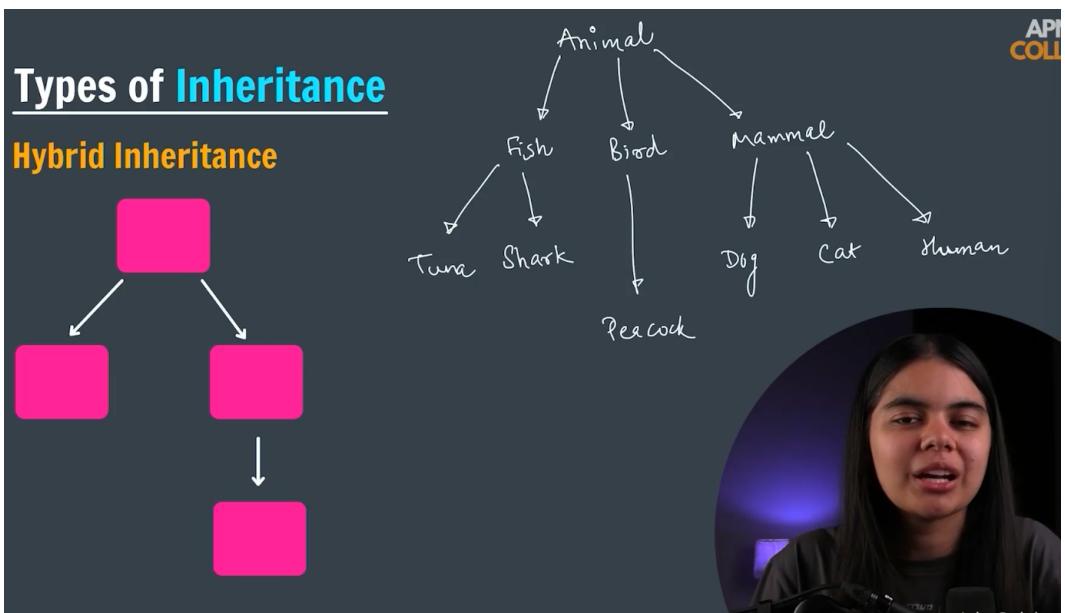
- Hierarchical Inheritance : Multiple Inherited classes form one single base class

Types of Inheritance

Hierarchial Inheritance



- Hybrid Inheritance :



- Multiple Inheritance : Is not available in JAVA (can be done using interfaces) , only in C++

▼ Polymorphism :

- When we try to do the same thing using different methods.
- Compile Time Polymorphism / Static :
 - Method / Function Overloading : Multiple functions with the same name but different parameters

Method Overloading

Multiple functions with the same name but **different parameters**

Calculator {

sum (int a, int b)

sum (float a, float b)

sum (int a, int b, int c) ✓

1 + 1 1.5 + 1.5

2 + 1 8.2 + 2.8

1 + 2 + 3

type
count

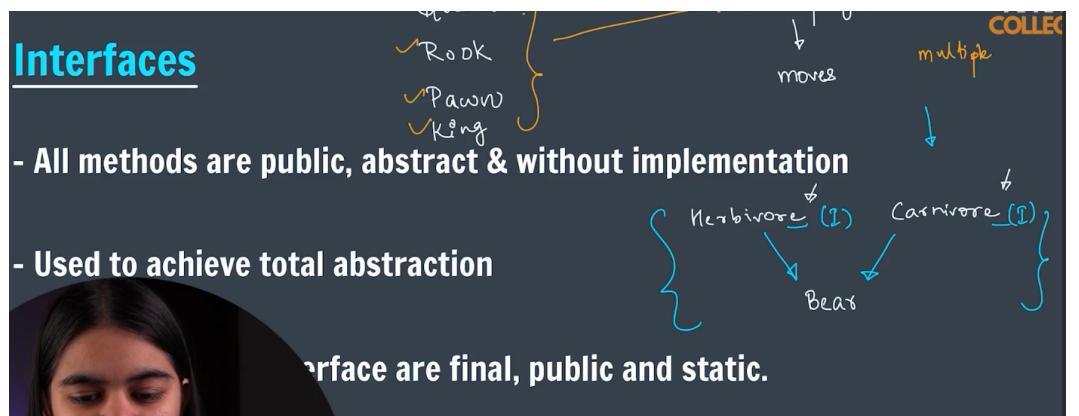
- Method / Function Overriding : Parent and child classes both contain the same function with a different definition.
- Always the child class function would be called

▼ Abstraction :

- Hiding all the unnecessary details and showing only the important parts to the user.
- How Abstraction is implemented :
 - Abstract Classes
 - e.g. The implementation need not to be defined. The implementation can be individually decided by each sub-class on there on .
 - It provides a kind of idea for the subclasses

```
abstract void walk();
```

- Interfaces : Interface is a blueprint of a class
 - Is used to implement multiple inheritance



```
public static void main(String args) {
    Queen q = new Queen();
    q.moves();
}
}

interface Herbivore {

}

interface Carnivore {

}

class Bear implements Herbivore, Carnivore {
}

interface ChessPlayer {
    void moves();
}

class Queen implements ChessPlayer {
    public void moves() {
}
```

Restart Visual Studio Code



- 100% abstraction is implemented using interfaces.
- It is defined using interface keyword
- Classes are extended whereas Interface is implemented
-

▼ Constructor :

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

```
Pen p1 = new Pen();
```

AP
COL

Constructors

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

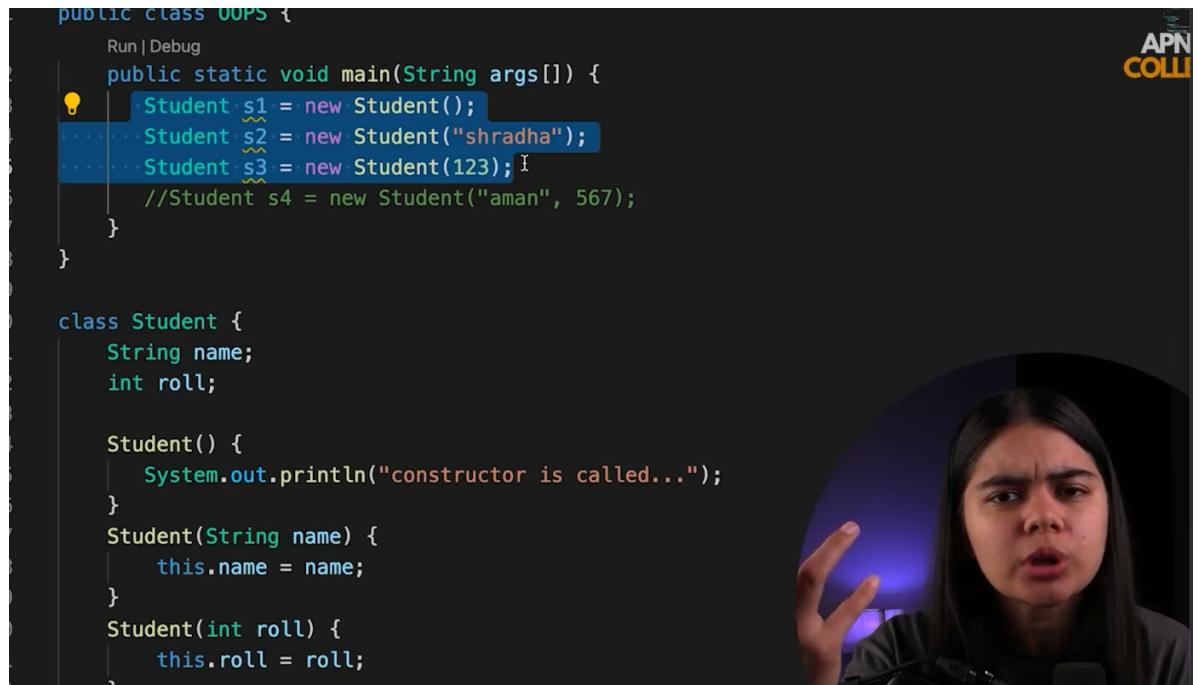
- Constructors have the same name as class or structure.
- Constructors don't have a return type. (Not even void)
- Constructors are only called once, at object creation.
- Memory allocation happens when constructor is called.



- If a constructor is not defined , JAVA assigns an default constructor to the object
- To set up some initialization values such as name or age we need to define the constructor ourself

▼ Types of Constructors :

- Non-parameterized
- Parameterized
- Copy Constructor



```

public class OOPS {
    Run | Debug
    public static void main(String args[]) {
        Student s1 = new Student();
        Student s2 = new Student("shradha");
        Student s3 = new Student(123);
        //Student s4 = new Student("aman", 567);
    }
}

class Student {
    String name;
    int roll;

    Student() {
        System.out.println("constructor is called...");
    }
    Student(String name) {
        this.name = name;
    }
    Student(int roll) {
        this.roll = roll;
    }
}

```

- Constructor Overloading : When we have created multiple constructors , so that multiple types of initialization statements could work.
 - It is a type of polymorphism

▼ Copy Constructor :

- Copy the properties of one object to another properties
- It is found as default in C++ but not in JAVA.

▼ Destructor :

Garbage Collector : Randomly checks and removes any unnecessary items from the memory

▼ Packages in JAVA :

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

▼ Keywords :

▼ Static Keyword

- Static keyword in Java is used to share the same variable or method of a given class.
- The static variable would be same for any and every other objects
- Static variable is only assigned value once.

- Same value can be used without saving the string multiple types.
- It also saves memory as all other events of the static keyword are used to refer to that already existing memory point.

▼ Super Keyword

- Super keyword is used to refer immediate parent class object.
- It is used to call the constructor of the immediate parent class
- Java by default call the super keyword.
- We can also alter the values of the parent using the super keyword.

```
class Horse extends Animal {  
Horse() {  
super.color = "brown";  
System.out.println("horse constructor is called");  
}  
}
```

- to access parent's properties
- to access parent's functions
- to access parent's constructor

▼ Question

▼ Find out the correct statement to assign name to object

- s.name = value (It is used to assign name to the name , s here is the object)

Practice Question 1

Find out the correct statement to assign name to object

```
class Student{
    String name;
    int marks;
}

public class OOPS {
    Run | Debug
    public static void main(String[] args) {
        Student s = new Student();
        //Fill here
    }
}
```

dot operator
s.name = "value"

- a. s->name = "aman" (C++)
- b. Student.name = "aman"
- c. s.name =



OOPS

Practice Question 2

Which variable(s) can the class Person access in the following code?

- a. name
- b. weight
- c. rollNumber
- d. schoolName

```
class Person{
    String name;
    int weight;
}
class Student extends Person{
    int rollNumber;
    String schoolName;
}
```



OOPS

▼ Which of the following modifiers are not allowed in front of class.

- In JAVA no outer/base class is allowed to be neither Public nor Protected as both these properties render the class as unusable

Practice Question 3

private class A {

class APNA
COLLEGE
A a = new A();

Which of the following modifiers are not allowed in front of class.

- a. private ← unusable
- b. protected
- c. public ✓
- d. default ✓

OOPS



JAVA

APNA
COLLEGE

	private	default	protected	public
Class	No	Yes	No ✗	Yes
Nested Class	Yes	Yes	Yes	Yes
Constructor	Yes	Yes	Yes	Yes
Method	Yes	Yes	Yes	Yes
Field	Yes	Yes	Yes	Yes



- ▼ Which of the following is a correct statement?

Practice Question 4

parent reference = child obj
 $\text{child ref} = \text{parent obj}$

Which of the following is a correct statement? (both classes in same package)

→ class Vehicle {}
 class Car extends Vehicle {}

Reference → Object
 (variable) (instance)

- a. Car c = new Car();
- b. Vehicle v = new Vehicle();
- c. Vehicle v = new Car();
- d. Car c = new Vehicle();



▼ Function Overriding

Practice Question 5

Vehicle
 ↓
 Car

→
 Car
 print BC
 APNA
 COLLEGE

What will be output of this code? (both classes in same package)

```
public class inheritance {
    public static void main(String[] args){
        Vehicle obj1 = new Car();
        obj1.print(); → fnx overriding (Derived Class(Car))

        Vehicle obj2 = new Vehicle();
        obj2.print(); → (base class (Vehicle))
    }
}

class Vehicle{
    void print(){
        System.out.println("Base class(Vehicle)");
    }
}
class Car extends Vehicle{
    void print(){
        System.out.println("Derived class(Car)");
    }
}
```

