Feature	C++ Struct	Java Class
Definition	Primarily used for holding data without methods.	Used for encapsulating data along with methods (functions).
Access Modifiers	By default, members are public.	By default, members are package-private (visible only within the same package). Can use access modifiers (public, private, protected) for control.
Inheritance	Supports inheritance, but members are public by default.	Supports inheritance, and access to members can be controlled using access modifiers.
Methods	Cannot have methods (functions) within a struct.	Can have methods (functions) within a class.
Encapsulation	Limited support for encapsulation. All members are public by default.	Encourages encapsulation. Members can be marked as private to control access.

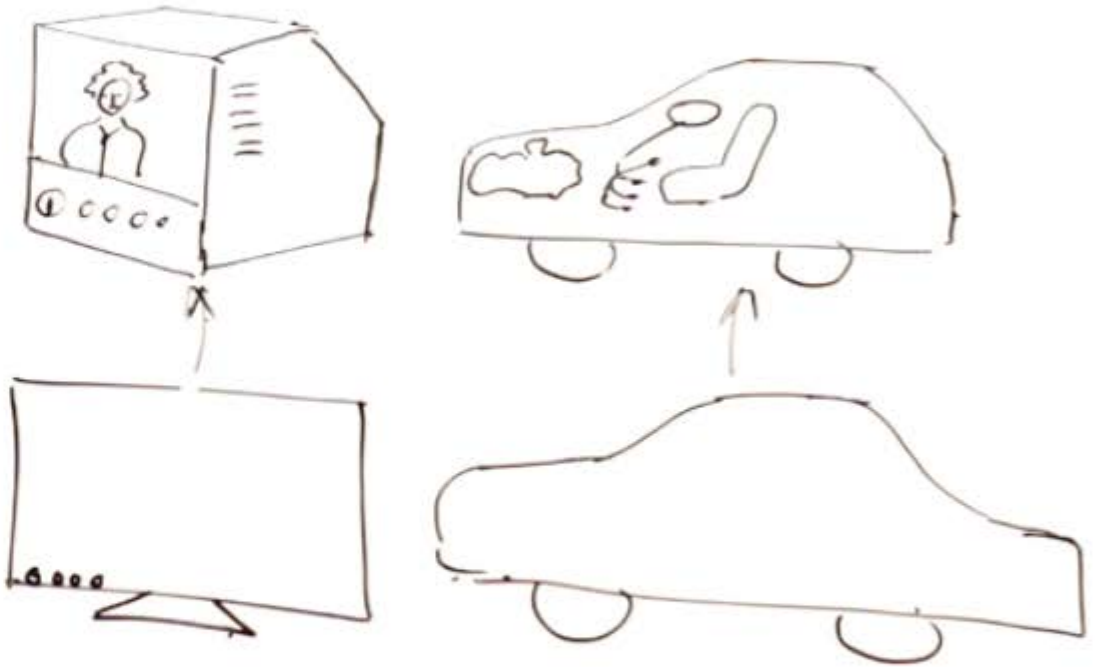
Constructor and Destructor	Can have constructors and destructors.	Can have constructors but no destructors (Garbage Collection takes care of memory).
Default Accessibility	Members are public by default.	Members are package-private by default.
Usage	Typically used for simple data structures.	Used for creating objects with behaviors and properties.
Memory Allocation	Memory allocation and deallocation are manual.	Memory management is automatic through Garbage Collection.
Syntax	`` `cpp struct MyStruct { int data; };` ``	`` `java class MyClass { int data; }` ``

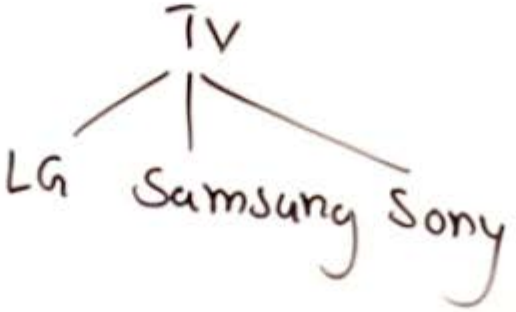
It's important to note that while C++ structs and Java classes have some syntactic and structural differences, their usage and purpose are influenced by the programming paradigms of each language. C++ provides more flexibility and manual control over memory management, while Java focuses on encapsulation, automatic memory management, and object-oriented programming principles.

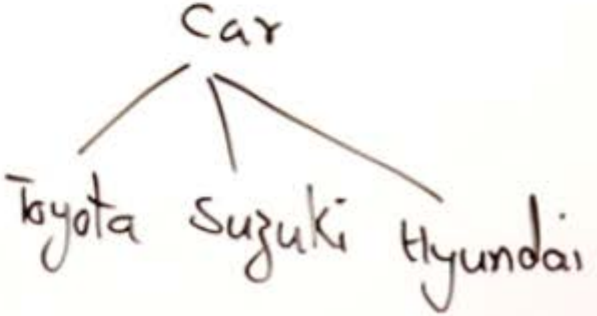
Class vs Object

1. Abstraction
2. Encapsulation
3. Inheritance
4. Polymorphism

1. Specialization
2. Generalization

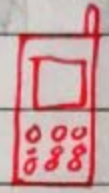






OOPs Terminology

1. Abstraction → Hiding internal details [show only essential info!]



⇒ use this phone without bothering about how it was made

2. Encapsulation → The act of putting various components together (in a capsule).

⇒ Laptop is a single entity with Wifi + Speaker + Storage in a single box!

In Java, encapsulation simply means that the sensitive data can be hidden from the users

3. Inheritance → The act of deriving new things from existing things.

Rickshaw ⇒ E-Rickshaw
Phone ⇒ Smart Phone

Implements DRY!

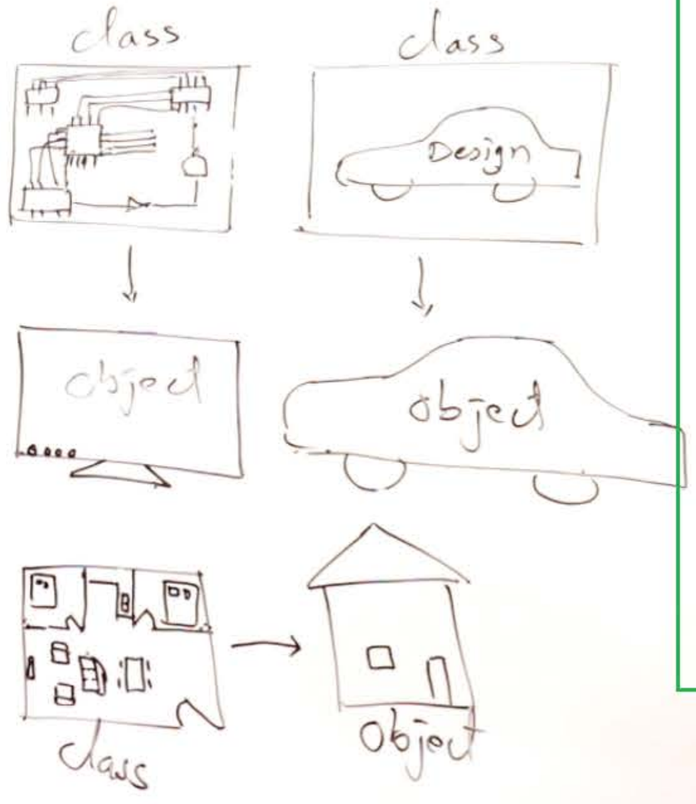
4. Polymorphism → One entity many forms

Smartphone → Phone Smartphone → Calculator

Class vs Object

```
class Television
{
    private int channel;
    private int volume;

    public void changeChannel()
    {
    }
    public void changeVolume()
    {
    }
}

Test

P.S.V. main()
{
    Television t=new Television();
    t.changeChannel(10);
}
```

object create karna ka method.



--> Class:

* Class is a group of objects which have common properties.
* ya ek trah template or blueprint hota hai jiska help sai object banta hai.

--> Object:

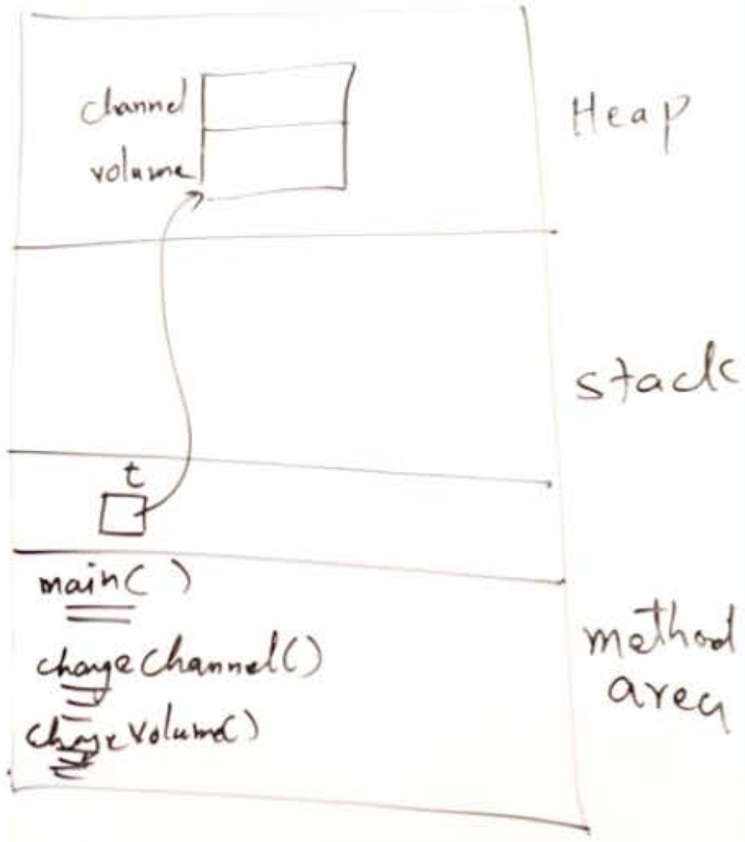
* ek real world object jo class ki help sai bnaya jata hai.
* iska khud ka ek property and behaviour hota hai.

Ex: change channel & volume of television.

* isa create karna ka liya mostly ham "new" keyword ka use karta hai.
* create karna ka method same waisa hi hota jaise ham "scanner" ka liya use karta hai.

Class vs Object

```
class Television
{
    private int channel;
    private int volume;
    public void changeChannel()
    {
    }
    public void changeVolume()
    {
    }
}

class Test
{
    P.S.V.main()
    {
        Television t=new Television();
        t.changeChannel(10);
    }
}
```



* Ham jab bhi koi reference create karta hai object ka liya (ise case mai "t") wo phela stack memory mai banta hai.

* Uska baad " Reference " ka instance jo bhi hota hai wo heap mai create hota hai.

* Class ka andhar jo bhi method define hota hai wo memory ka method area mai create hota hai.

Class Circle

Properties:

double radius

Methods:

double area()
" perimeter()
" circumference()

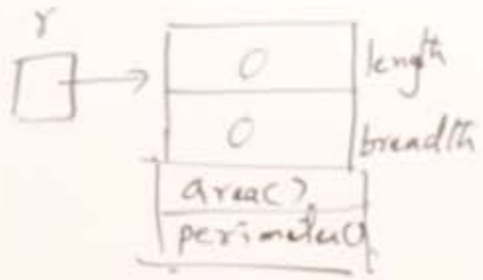
class Circle
{
=
=
}

class My
{
p.s.v.m main()
{
}
}

--> Class bnata waqt hmsa 2 chij ka use karta hai.

i) property
ii) methods

Data Hiding




```
class Test
{
  p.s.v.main(..)
  {
    Rectangle r=new Rectangle();
  X r.length=10;
  X r.breadth=5;
    S.O.P(r.area());
  }
```

```
class Rectangle
{
  private int length;
  private int breadth;

  public int area()
  {
    return length*breadth;
  }
  public int perimeter()
  {
    return 2*(length+breadth);
  }
}
```

--> Data hiding ek concept ha jiska help sai ham, important information ko user sai hide kar sakta hai and program ko aur user friendly bna sakta hai.

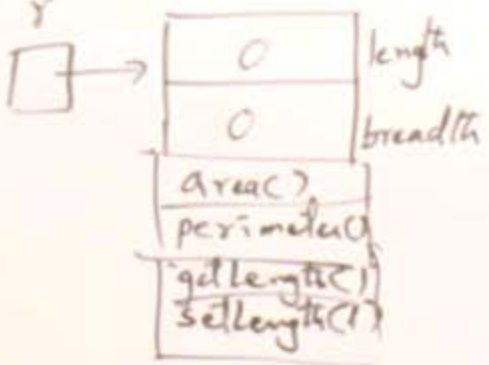
--> jab ham data type ko private bna deta hai to usa class ka bahar access nahi kiya ja sakta hai.
--> phir usa access karna ka liya hma getter - setter method ka use karna hota hai.

```
class Rectangle
{
    private int length;
    private int breadth;

read    int getLength( )
        {
            return length;
        }

write   void setLength(int l)
        {
            length=l;
        }
```

--> private ko access karna ka liya jo get.....set.... method ka use hota hai ya wahi method hai.

--> "get" ka use read ka liya and "set" ka use write ka liya karta hai.

Data Hiding




```
class Test
{
    p.s.v.main(--)
    {
        Rectangle r=new Rectangle();
        r.setLength(10);
        r.setBreadth(5);
        S.O.P(r.area());
```

```
class Rectangle
{
    private int length;
    private int breadth;

    public int getLength()
    {
        return length;
    }
    public void setLength(int l)
    {
        if(l>0)
            length=l;
        else
            length=0;
    }

}
```

Type of Properties

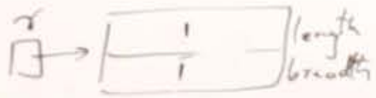
1. Read & Writable
2. Read Only
3. Write Only

--> Read & write method mai ham "get - set" dono ka use karta hai.
* mostly ise mth ka use hota hai.

--> Read method mai sirf "get" ka use karta hai.

--> Write method mai sirf "set" ka use karta hai.
*ya dusra sab sai important property hai jiska widely use hota hai.

Constructors




```
class Test
{
  p.s.v.main(..)
  {
    Rectangle r=new Rectangle();

  X r.setLength(10);
  X r.setWidth(5);
```

```
class Rectangle
{
  private int length;
  private int breadth;

  public Rectangle()
  {
    length=1;
    breadth=1;
  }
  public Rectangle(int l,int b)
  {
    length=l;
    breadth=b;
  }

}
```

--> Constructor is special type of method whose name must be same as "class" name.

--> Jab haam chahta hai ki "object" creation ka time hi kisi value ko initialize karna then haam use code ko constructer mai rakh sakta hai.

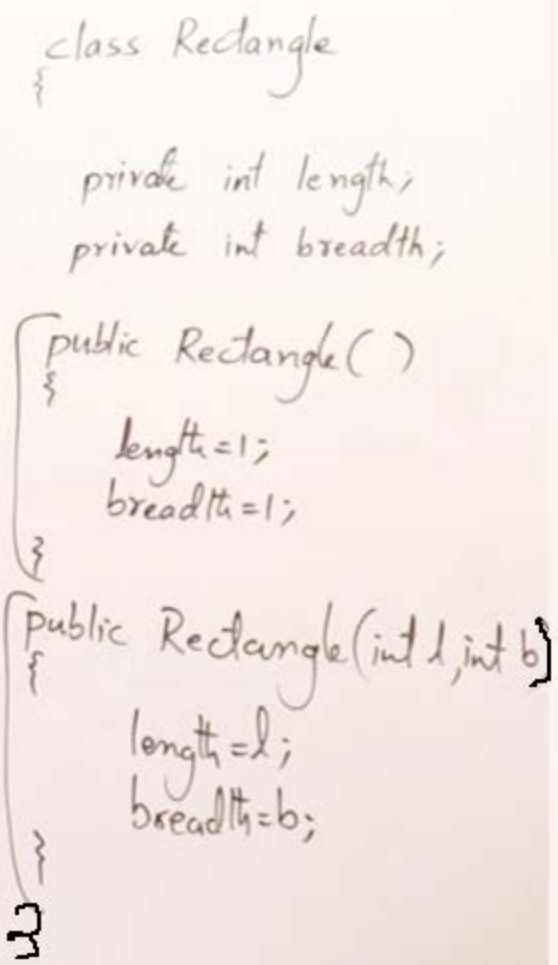
--> Agar ham koi aapna constructor define nahi karta hai too, java hma khud ka ek default constructor deta hai.

* Way to create constructor:

i) same name as class name.

ii) may be public or private.

iii) koi bhi return type data nahi hona chahiya.



※ Type of constructor:

i) Parameterized Constructor

--> Aisa constructor jiska defination ham kuch parameter pass karta hai.

ii) Non-Parameterized Constructor

--> Aisa constructor jiska defination ham kuch parameter pass nahi karta hai.

```java
public class SCoops3
{

    public static void main(String[] args)
    {
        Subject subs[]=new Subject[3];
        subs[0]=new Subject("s101","DS",100);
        subs[1]=new Subject("s102","Algorithms",100);
        subs[2]=new Subject("s103","Operating Systems",100);

        for(Subject s:subs)
            System.out.println(s);

    }

}
```

--> Way to create "object of array".