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.
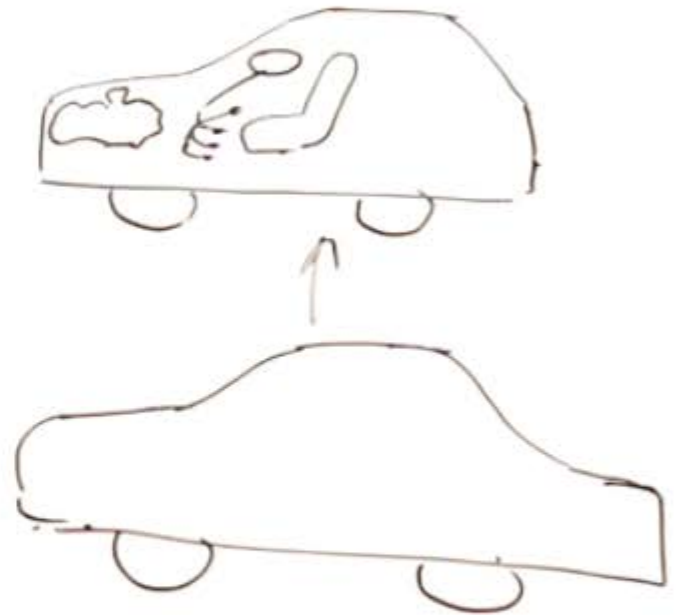
Class vs Object

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

1. Specialization
2. Generalization
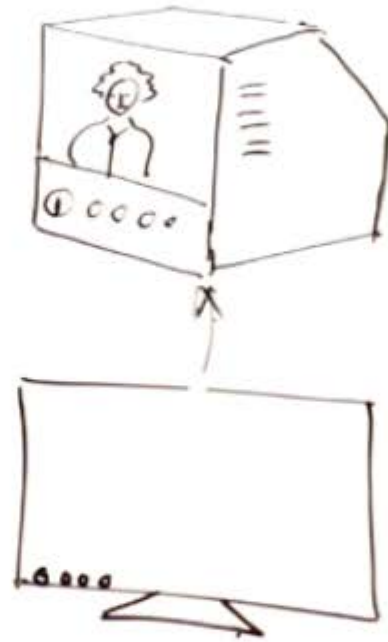
TV

LG   Samsung Sony

Car

Toyota Suzuki Hyundai

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
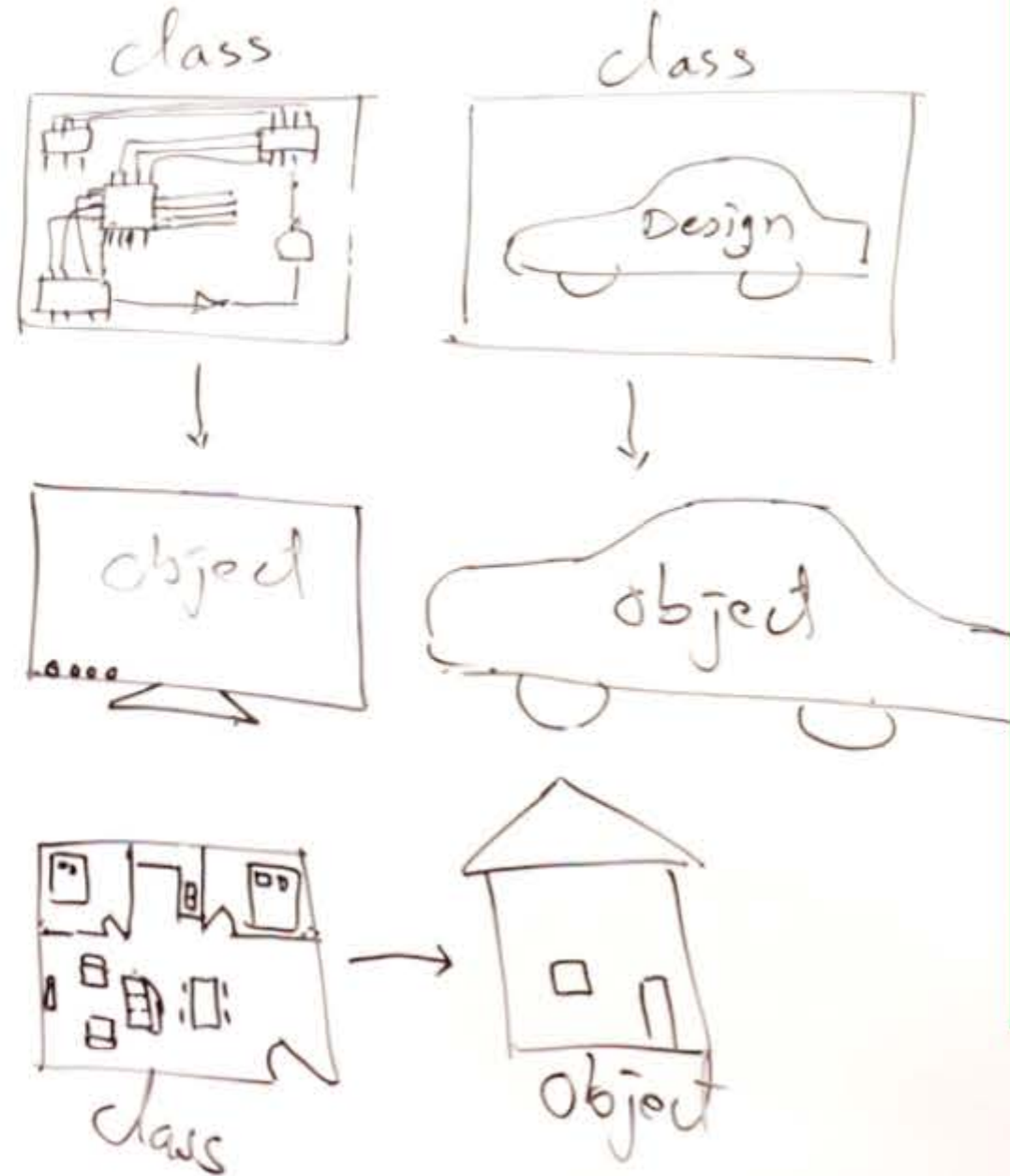
Class vs Object

```
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 → object (television)

class → Design → object (car)

class → Object (house)

--> 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.
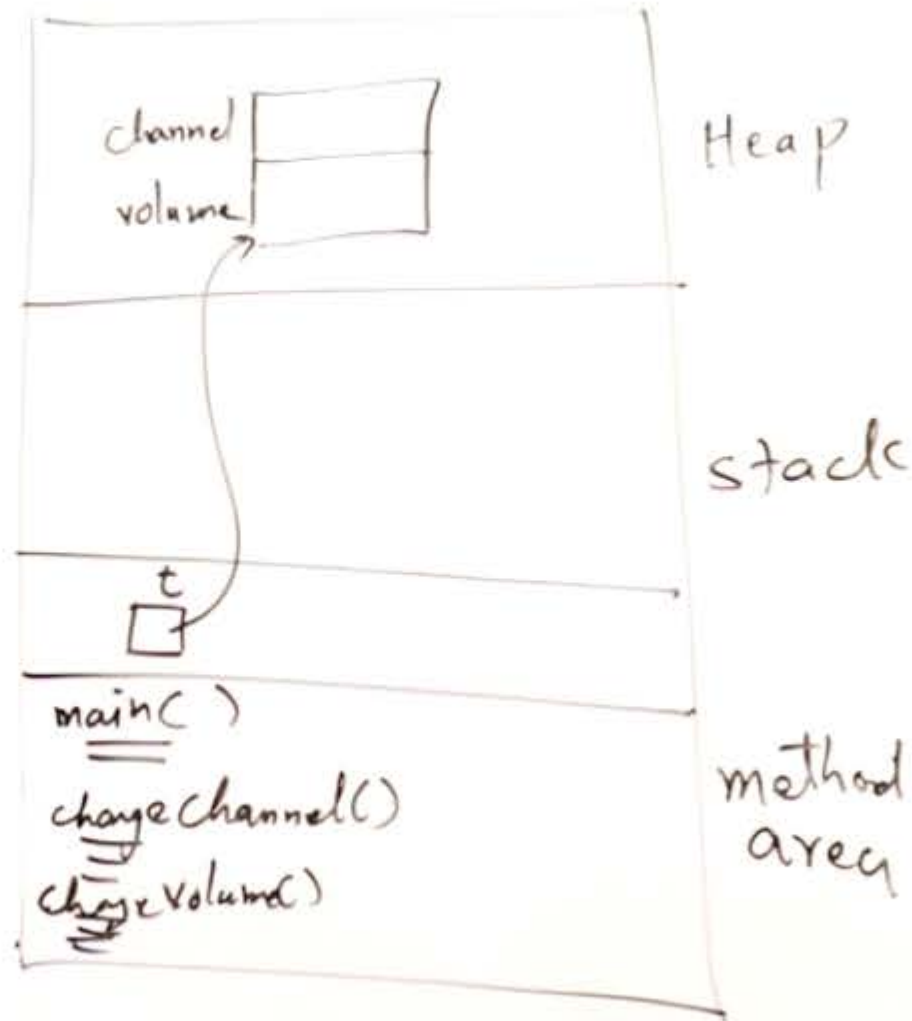* create karna ka method same waisa hi hota jaise ham "scanner" ka liya use karta hai.

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);
    }
}
```

channel
volume

Heap

t

stack

main()
change Channel()
change Volume()

method
area

Class Circle

Properties:

double { radius

Methods:

double { area()
"   perimeter(.)
"   circumference()

class Circle
{

≡

}

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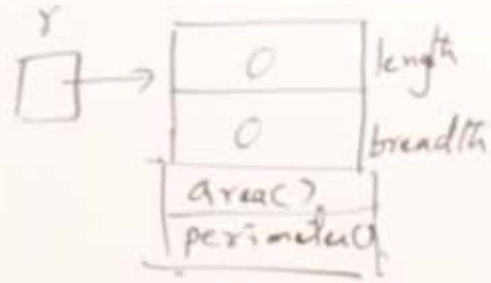   }
}

Data Hiding

class Test
{
  p.s.v.main(..)
{

    Rectangle r=new Rectangle();

    ✗ r.length = 10;
    ✗ 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.

Q. Explain concept of data hiding in java in detail.

Ans.: Data hiding is one of the core principles of **Object-Oriented Programming (OOP)**, and it refers to the practice of restricting direct access to some of an object's data and methods. It allows an object to shield its internal state (data) from outside interference and misuse, ensuring better control over how data is modified or accessed. In Java, this is achieved primarily through **encapsulation** by using **access modifiers** and getter/setter methods.

Data hiding helps to:

1. **Protect data integrity** by controlling modifications.
2. **Reduce complexity** by hiding implementation details from the user.
3. **Improve security** and maintainability.

Key Concepts of Data Hiding in Java

1. **Encapsulation**: This is the mechanism that helps in data hiding by bundling data (attributes) and methods (functions) that manipulate the data into a single unit (a class). Encapsulation is achieved by using:

   o **Private** variables (to hide the data).

   o **Public** methods (to provide controlled access).

2. **Access Modifiers**: Java provides four access levels (modifiers) to control visibility and access to class members (attributes and methods):

   o **Private**: The most restrictive access level. A private member is accessible only within the class in which it is declared.

   o **Default** (package-private): Accessible only within the same package.

   o **Protected**: Accessible within the same package and by subclasses (even if the subclass is in a different package).

   o **Public**: The least restrictive. A public member can be accessed from any class or package.

How Data Hiding is Implemented

Example of Data Hiding with a Class

```java
public class BankAccount {
    // Private attributes: hidden from outside classes
    private String accountNumber;
    private double balance;

    // Constructor
    public BankAccount(String accountNumber, double initialBalance) {
        this.accountNumber = accountNumber;
        this.balance = initialBalance;
    }

    // Public method to access the balance (getter method)
    public double getBalance() {
        return balance;
    }
}
```

```
// Public method to deposit money (validates the input)
public void deposit(double amount) {
    if (amount > 0) {
        balance += amount;
    } else {
        System.out.println("Invalid deposit amount");
    }
}

// Public method to withdraw money (validates the input)
public void withdraw(double amount) {
    if (amount > 0 && amount <= balance) {
        balance -= amount;
    } else {
        System.out.println("Insufficient funds or invalid withdrawal amount");
    }
}
}
```

Explanation:

1. **Private Data Members**:
   - In the BankAccount class, the attributes accountNumber and balance are marked as private. This means that no other class can directly access or modify them.

2. **Public Methods for Access**:
   - The getBalance(), deposit(), and withdraw() methods are public. These methods allow controlled access to the balance attribute while ensuring that only valid operations (like depositing a positive amount) can be performed.

3. **Data Protection**:
   - By using private variables and providing public methods, the class ensures that the account's balance cannot be arbitrarily modified by external classes or users, which protects the integrity of the data.

Why Data Hiding is Important

1. **Security and Integrity of Data**:
   - When class attributes are hidden using the private modifier, direct access from outside the class is not possible. This prevents unintended or malicious changes to the object's state. For instance, you can't directly set a bank account's balance to a negative number by mistake.

2. **Controlled Access**:
   - Data hiding allows you to control how external users interact with the object's data. By exposing only specific methods to access or modify the data, you ensure that the data is handled appropriately. For example, a bank account's balance can only be updated through deposit or withdrawal operations, which validate the inputs.

3. **Abstraction**:

   o Data hiding helps in achieving **abstraction**, where the internal workings of a class are hidden from the user, and only essential details are exposed. The user doesn't need to know how the balance is stored or manipulated internally; they only interact with the deposit and withdrawal methods.

4. **Loose Coupling**:

   o By hiding internal data, objects become more independent and loosely coupled. Other parts of the program don't need to know or rely on the exact implementation of the data and methods inside the class. This makes the code more modular and easier to maintain or update.

5. **Maintainability**:

   o Since the internal implementation details are hidden, you can change the inner workings of the class (like changing the way data is stored or validated) without affecting other parts of the program that rely on the class. The public methods act as a stable interface, providing consistency.

Key Takeaways:

- **Private Data Members**: Data hiding is achieved by marking sensitive variables as private to prevent direct access from outside the class.

- **Public Methods**: These provide controlled and validated access to the hidden data, ensuring that external users cannot tamper with the object's state inappropriately.

- **Security**: Data hiding ensures that the internal state of an object is protected, reducing the risk of data corruption or misuse.

- **Abstraction and Encapsulation**: Data hiding contributes to the principles of abstraction (hiding implementation details) and encapsulation (binding data and methods together), both essential features of object-oriented programming.

Data hiding is a fundamental concept in ensuring that a Java class is robust, secure, and well-structured. It promotes better code maintainability, reduces complexity, and protects the integrity of the data encapsulated in an object.
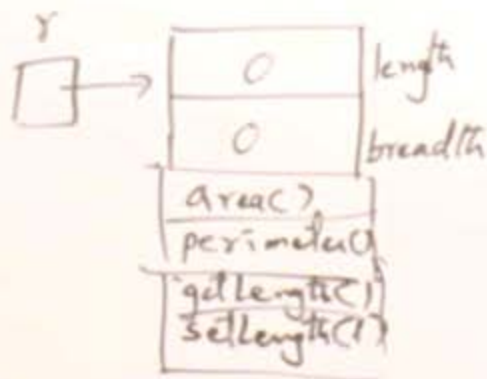
Data Hiding


```
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

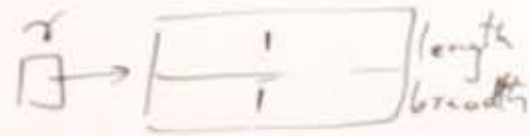--> Read & write method mai ham "get - set" dono ka use karta hai.
* mostly ise mth ka use hota hai.

--> 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.

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

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

※ 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".