Object Oriented Lang

- Uses object & classes
- Data → encapsulated within objects
- Support inheritance, making reuse easy
- Data security

Java, C++, Python, Kotlin, Swift, C#

Procedural Lang

- Uses procedures & for
- Data is separated from for
- Code reuse is possible but not as structured
- C, Pascal,

OOPS

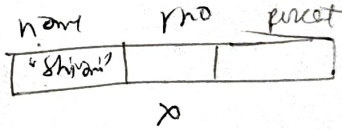
User defined data type

int, char, double
↓
Store data efficiently

```
Public static class Student
    String name;
    int rno;
    double percent;
}
```

← Create a data type
No Memory allocation

```
Public Static void main (String [] args){
    Student s = new Student();   // declaration
    s.name = "Akki";
    s.rno = 76;
    s.percen = 92.5
    Sout ( )
}
```



ACCESS Modifier

1. Public - all packages
2. Private - Same class
3. Default - Same package ← Same package

OOPs

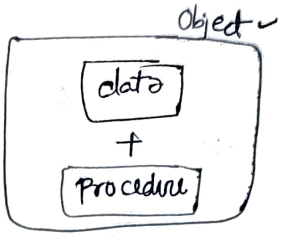
Procedural programming

operation → data

Writing Procedure or Methods

- More Organize
- Java follow DRY
  - "Don't Repeat Yourself"

OOPs — Bind data & fun



- faster to execute
- Binding data & fun preventing unauthorized access from other part of the code.

4 Pillars

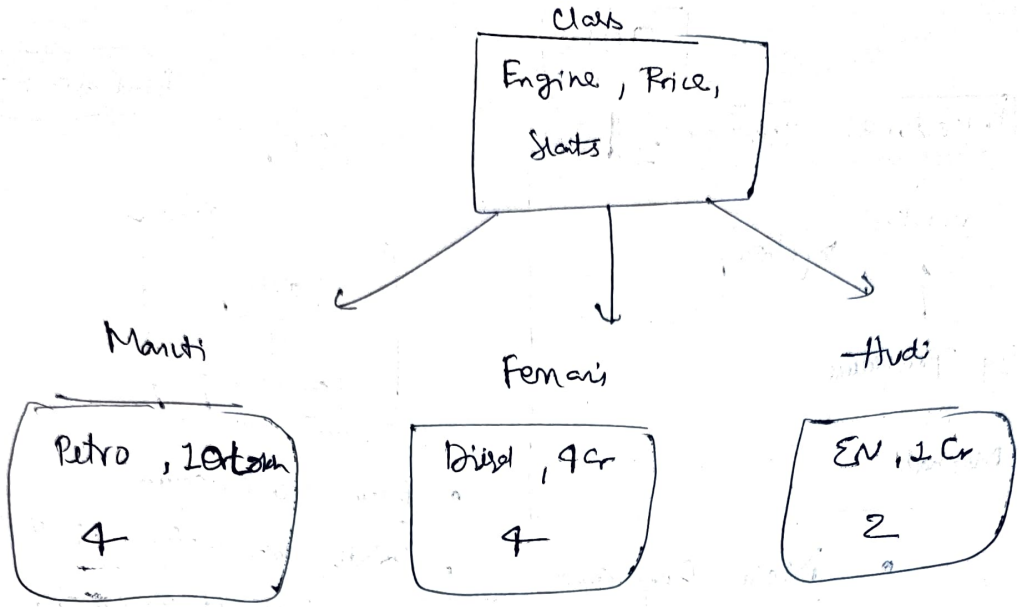
1. Abstraction
2. Encapsulation
3. Inheritance
4. Polymorphism

Class & Objects

6. Group of properties & fun

Classes & Objects

Class	Object
1. Logical Construct ↓ No memory allocation	1. Physical Entity ↓ Memory allocated when obj is created
2. Class is templet of object or blue print	2. Object is instance of class
3. Declared only once	3. Object can be created many times
4. Group of similar objects	4. Real world entity — Bank, car etc
5. Create a data type for you	



dot (.) operator

Creating class of type Student

```
Class Student {
    int rno;
    String name;
    float marks;
}
```

Creating object

```
Student S1 = new Student ();
```

dynamically allocate memory & return a ref to it.

Constructor

default constructor

```
Student S1 = new Student ();
```

Compile time — Runtime

int & char ← not new

- Java primitives data types are not implemented as objects.

Constructors → To initialize Properties

Special method used to initialize object when they are created.

↓

Automatically called when object is instantiated using new keyword

this ⟷ Akhil

```
class Student {
    int rno;
    String name;
    float marks = 90;

    Student() {                    ← Constructor
        this.rno = 13;
        this.name = "Akhil"
        this.marks = 88.5f
    }

    Student (int roll, String nam, float marks) {
        this.rno = roll;
        this.name = nam;
        this.marks = marks;
    }
```

Cash	V/s	Methods
① Name		
② Return type		Must have
③ Invocation (automatic)		Called explicitly
④ Purpose		

Code Running

① Class { }

② Void main()

To make more efficience

int a = 10

Java	Python
a = 10	a → 10
Only Store in Stack Memory	As object (10)

int a = 10;

Wrapper Class

Integer num = 45;

To create as object.

num.

Using primitive as object

```
static void swap (int a, int b){
    int temp = a;
    a = b;
    b = temp;
}
```

3 3

In Java

Pass by reference ✗

" " value ✓

No swapping in Java

Pass by reference ✗ No swapping

Integer num = 10

Integer class is final class

Final keyword

Prevent content not to be modified.

final keyword ⇒ All capital

Always initialise while declaring it.

IMMUTABILITY is only for Primitive data type

final Student stds = new Student();

stds.name = "new name";

Final = after object

distructor

What to do when object is destroyd. ✓

object destroy ✗

@Overrid
protected void finalize()
Sout ("Object is destroyed")
}

Packages, Static, Singleton Class, In-built Methods

↓

Containers for classes ✓

Static
↓ Can be used without creating a class

public static void main (String[])

}

Static void greet

Static
Static ✓
Non static ✗

Static Method can only access to static data.

Not Static → Belong to object

static {
Sys.out ("I am in static block")
b. 005;
}