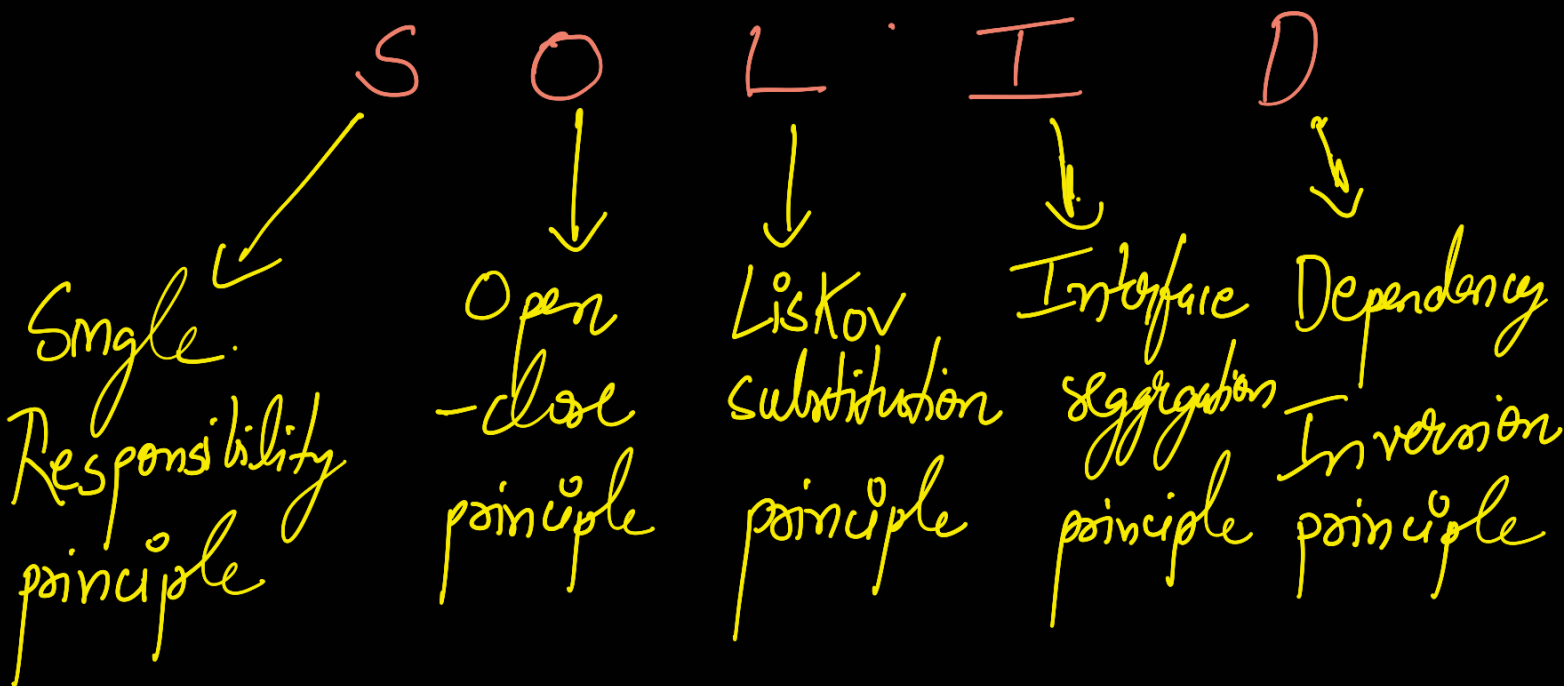# SOLID Principles

* We have learnt OOPS and concept of classes and object but designing a good class and Object oriented design is hard.
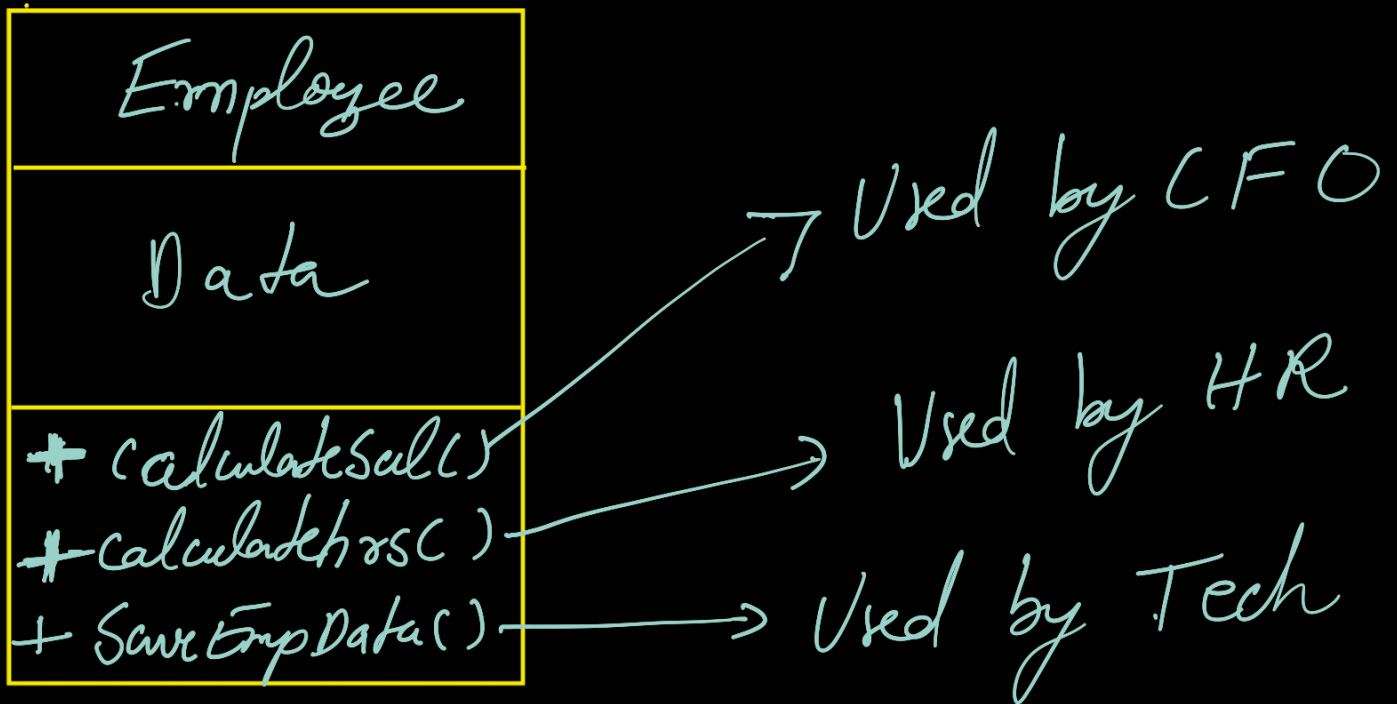
* To design classes, there are some principles which ensures the source code is extendable, managable and bug free.
  ↳ Their principles are :

S   O   L   I   D

Single Responsibility principle

Open -close principle

Liskov substitution principle

Interface segregation principle

Dependency Inversion principle

1. Single Responsibility principle

* Any Module should be changed by only "1 Actor"

```
┌─────────────────────────┐
│       Employee          │
├─────────────────────────┤
│                         │
│       Data              │
│                         │
├─────────────────────────┤
│  + CalculateSal()       │
│  + CalculateHrs()       │
│  + SaveEmpData()        │
└─────────────────────────┘
```

→ Used by CFO

→ Used by HR

→ Used by Tech

Let's say there's a private method getHrs(); this is used inside

CalculateSal()   CalculateHrs()

* Now there is some change required in CalculateSal() which also changes

gethrs() $\longrightarrow$ this will impact calculatehrs() as well.

So, change made by one person is affecting other functions which may be used by some other team.

* This is bad design of class, because it is possible that this class is used by various other teams and code change by one team affects the other team.

$\Downarrow$ solution

Break the class

Cal salary class   Cal hrs class   Store data class