#Why engineer software? dmec

Difficulties → ic - dr - hm

SDLC → RD Dental DepartMent

Nature of Software → IRLU W f M

Software engineering code of ethics → ipl chief

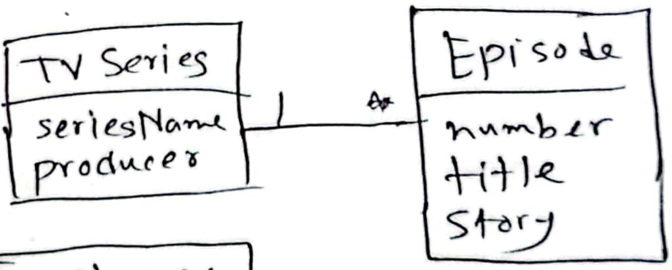
Software engineering projects → Green Care

Programming style guidelines → up dog clp

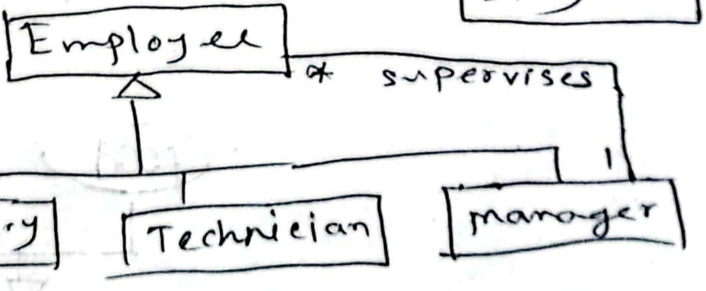
type of reuse → calfed

UML diagrams → acid

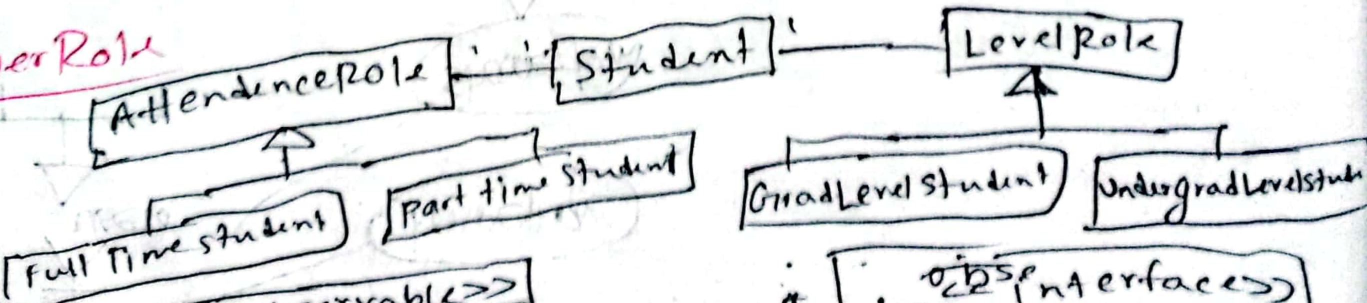
Abstraction-Occurance



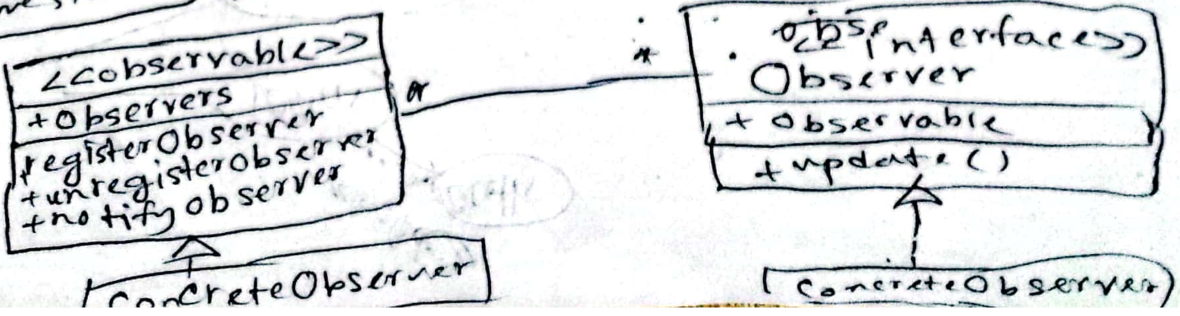
General-Hierarchy



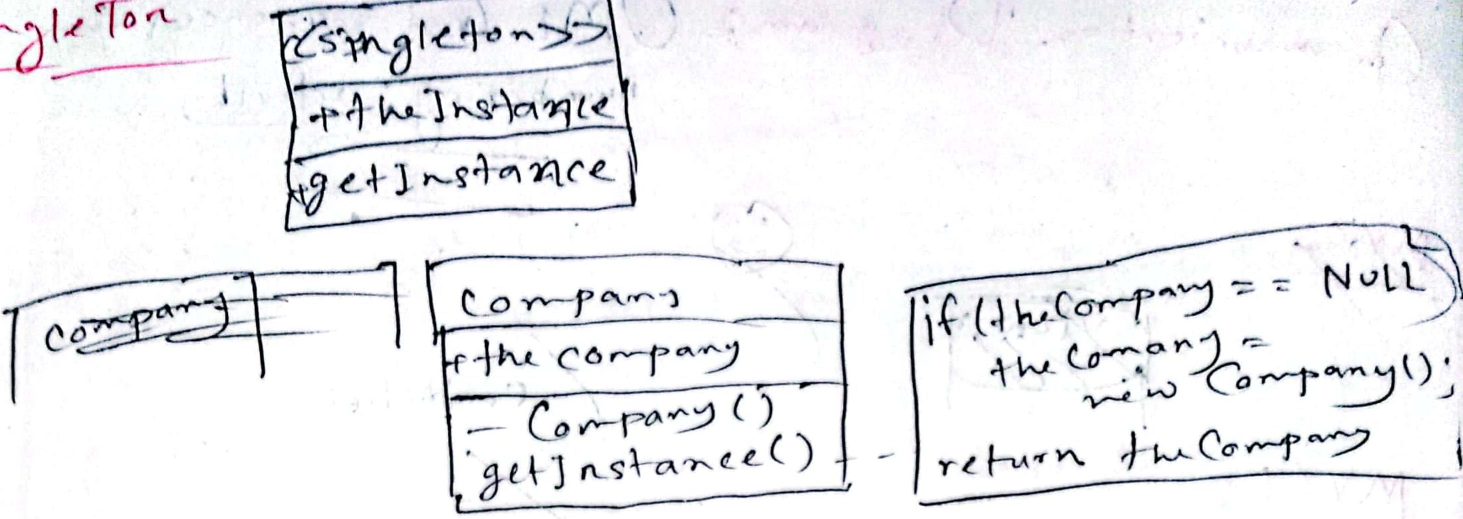
PlayerRole



Observer

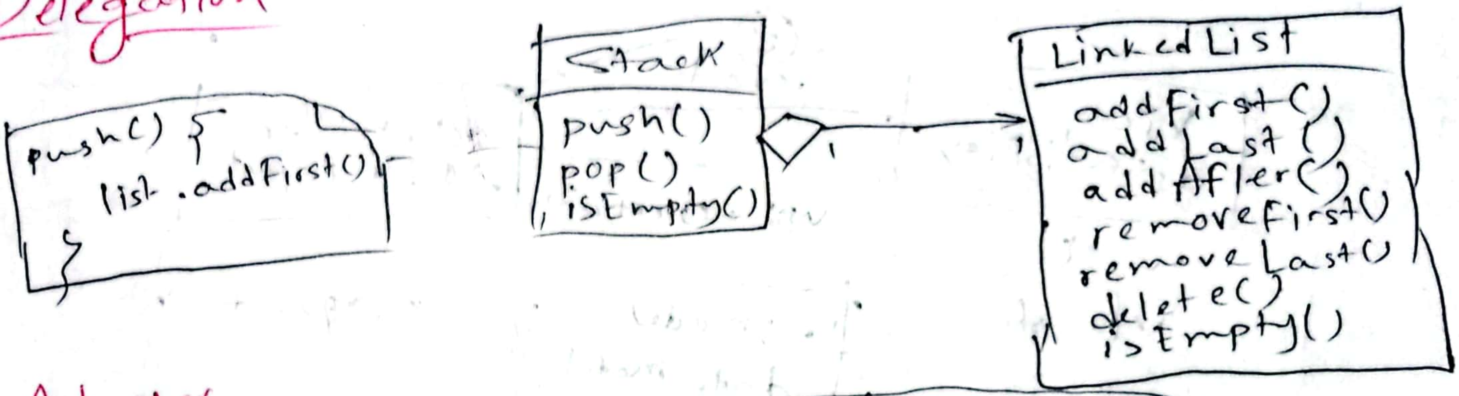


SingleTon



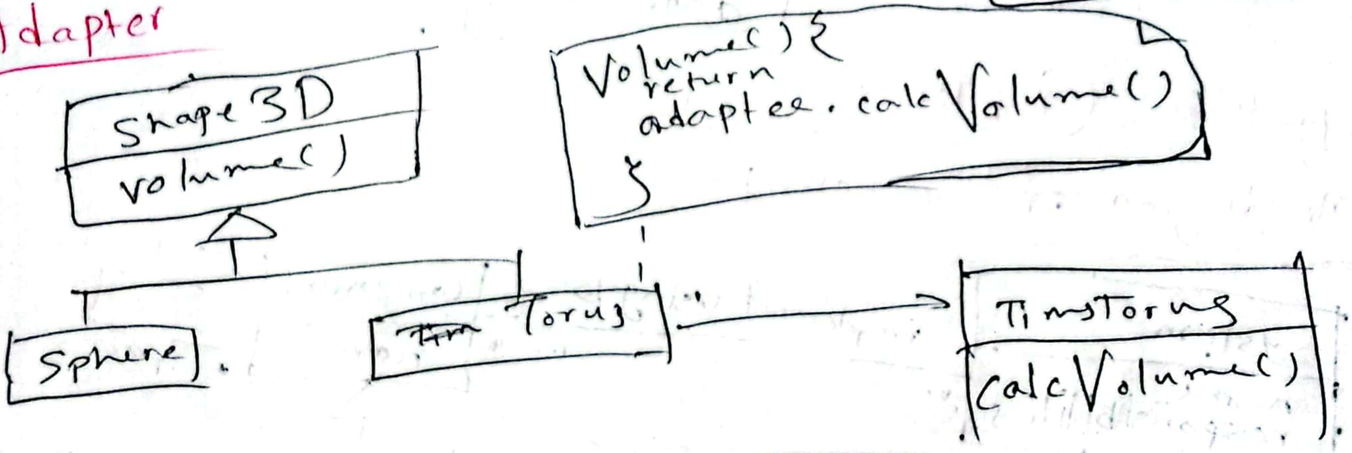
```
if (theCompany == NULL
    theComany =
        new Company();
return theCompany
```

Delegation



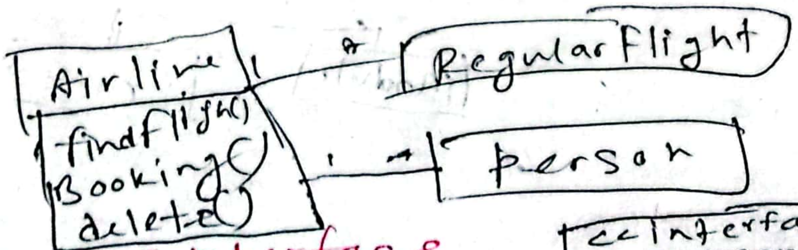
```
push() {
    list.addFirst()
}
```

Adapter

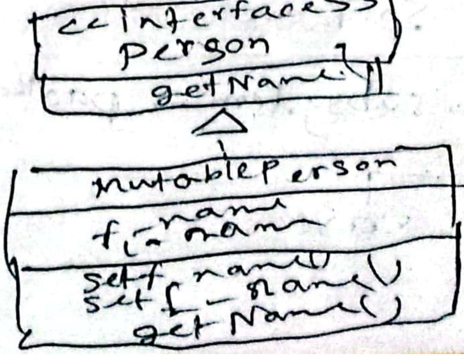


```
Volume() {
    return
    adaptee.calcVolume()
}
```

Facade



Read-Only-Interface

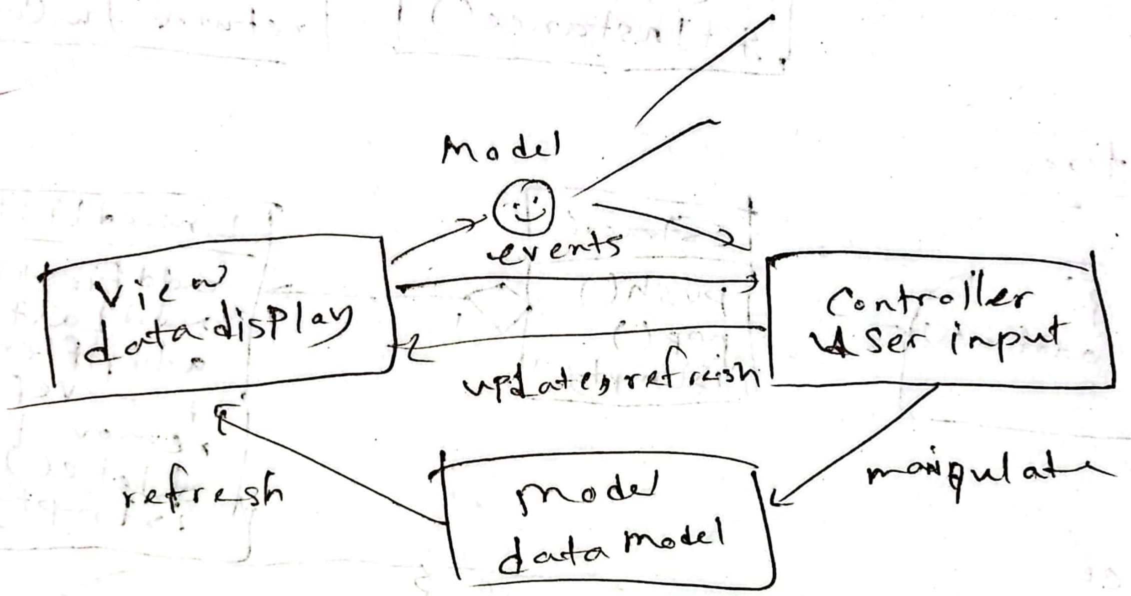


Communication Diagram: UML (not sequence) + numbering: activity

→ or ←

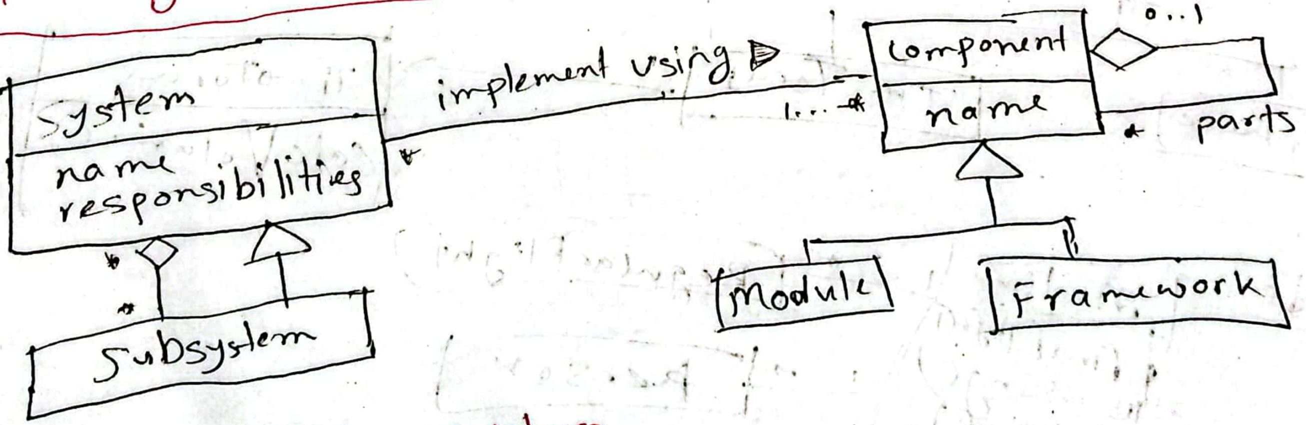
~~MVC:~~

MVC:



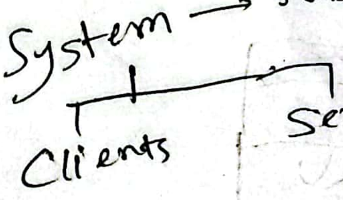
Aspects of Usability: eela

UML diagram of system parts



Dividing Software system

System → subsystem → packages → class → method



Coupling: DSCECC

Data: By passing data.

Stamp: By passing of data structure

control: using control information

External: Measure dependency betⁿ a software system and external entities

Common: Share some global data

Content: Part of content of another module.

Routine call: One routine calls another.

Type use: uses a data type defined in another module.

Inclusion or import: one component imports a package or one component includes another

Cohesion: CLTPCSF

Coincidental: Only relationship between functions in a module is coincidental.

Logical: All the elements of a module perform similar or slightly similar operations.

Temporal: Elements of a module execute at the same time.

Procedural: ~~Ee~~ Functions of a module are related to each other through a data flow.

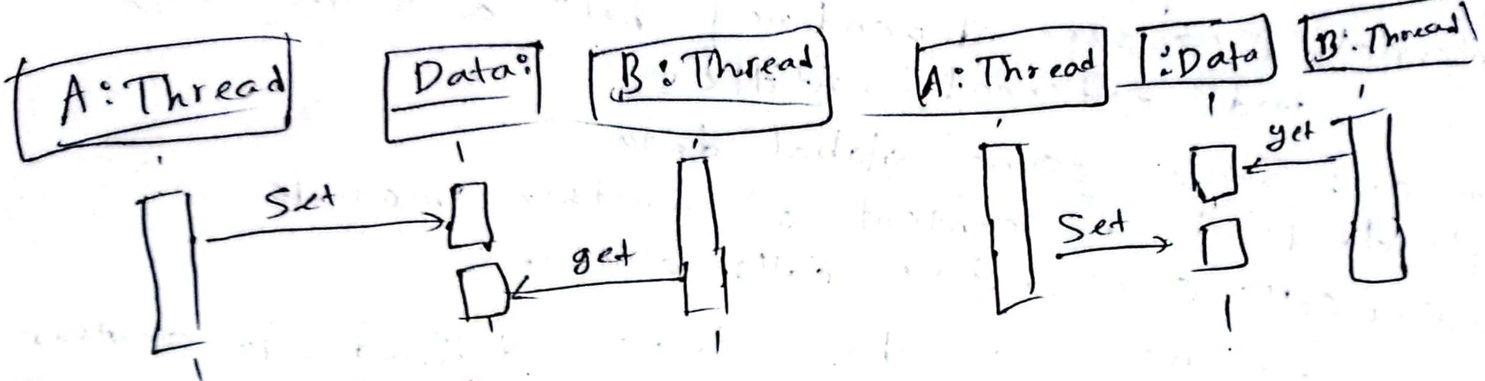
Communicational: Different functions are working on the same data structure

Sequential: Elements are executed in a specific sequence to perform a single task.

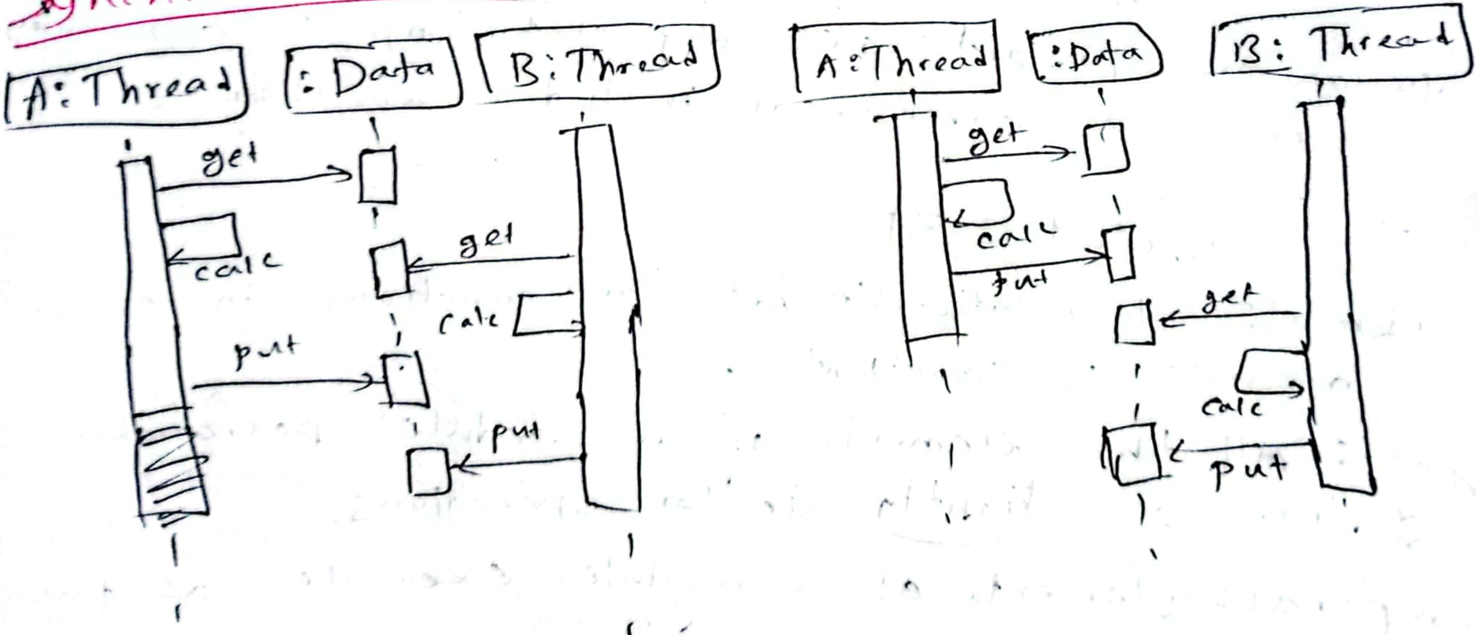
Functional: Functions co-operate with each other to perform a single functionality.

Critical race

one thread experience failure because of another thread's interfaces the ~~the~~ normal sequence of events

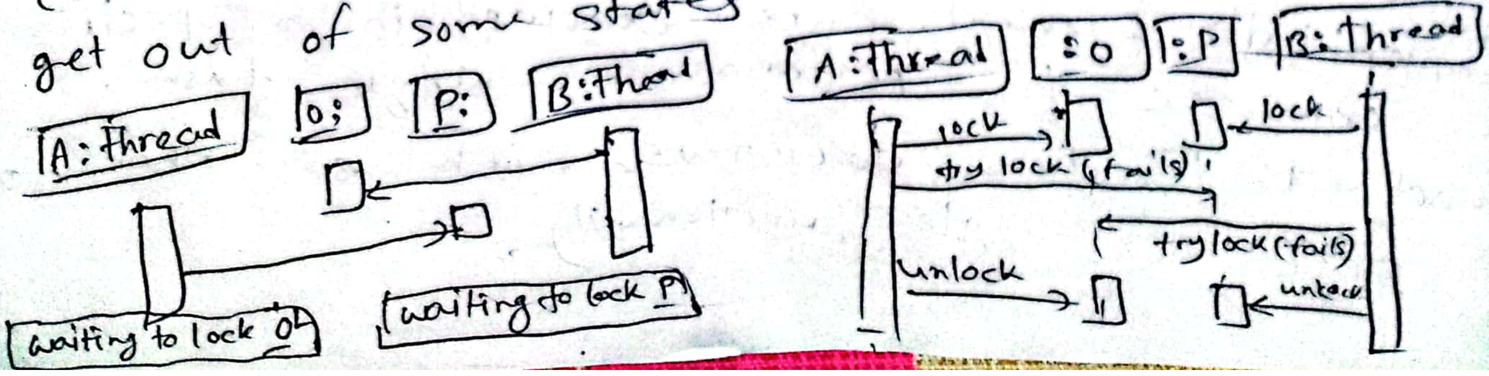


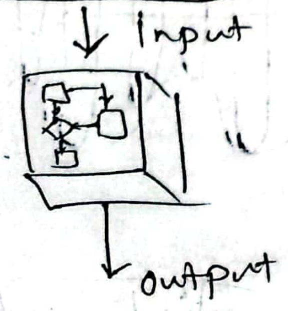
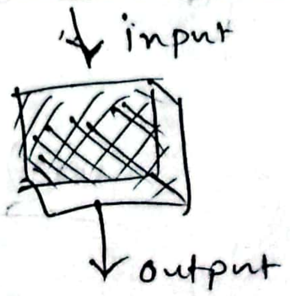
Synchronization:



Deadlock: Two or more threads are stopped. waiting for each other to do something

Livelock: Similar to deadlock. but the system can do some computations, but ~~not~~ can never get out of some states



White Box	Black Box
Developers can perform	Test engineers can perform
What the software suppossed to do, also aware of how it does to do	What, the software supposed to do, but not aware of how to do
Should have understanding of programming languages.	No need to have an understanding of the programming language
We will look into the Source code and test the logic of the codes.	We will verify the functionality of the applicat based on requirement specification
Developer should know about the internal design of the code.	No need to know about the internal design of the code
Applied mainly at ~~unit~~ lower ~~higher~~ testing level.	Can be applied ~~virtually~~ to ~~every~~ ~~lower~~ higher level of testing
Easy to automate	Tough to automate
Structural means test or interior test	means functional test or Rxternal tesit
 input output	 input output