AGENDA

- Intro to design Pattern
- Types of DP
- Singleton DP

*) DESIGN PATTERNS

Repeated s/w Patterns in code

Established solutions to common design Problems.

Eg:

1.) Should we create Parent/child class ?
2.) Abstract class / Interface

Hence, if you face any Problem → use existing solution

Into the facts:

-) Initially discovered by book GOF (Gang of four)
-) talks about 23 Problems & solutions

NOte: we would cover ~10 most famous / commonly used DP in industry.

*) WHY D.P. are Important :

1.) They are standard practise used by people in Industry.

2.) saves lot of time → creating own solution always Not Good.

3.) V.V. Important in Interviews.

(eventually - Implementation of design Principles)

*) TYPES OF DP:

classified based on OOP. (JOURNEY of Object)

1.) OBJECT CREATION:

dp which deal with Object creation flow

(How created / How Many...) singleton

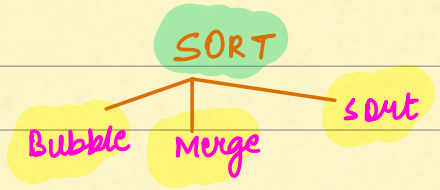
2.) STRUCTURAL DP:

deals with how a class is structured

attributes / Methods / Interaction

3.) BEHAVIOURAL DP:

decides behaviour execution



CREATIONAL DP:

SINGLETON DESIGN PATTERN:

structure:

definition ①
↓
Problem statement ②
→ Implementation ③
↓
Pros/cons ④

Says:

Allows us to create only (1) Object at a time for Any class that is SINGLETON.

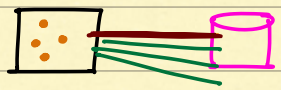
Eg:

X → singleton

```
X x = new X();
```

x → #a101 (Address)

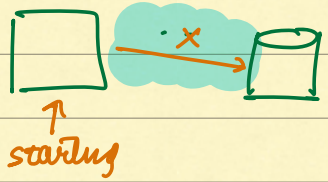
Every where in code, (x) points to #a101



NEED FOR SUCH CLASS:

1.) class which has shared resource
(Eg: DBConnection)

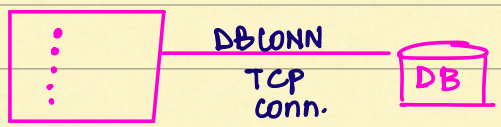
DBConnection
- URL
- userName
- Password
- cache



This DBConnection class is accessed from Many Places in the code

Eg:

UserService
InstructorService
StudentService



T.C.P conn. are slow

Q.) do we Need (1) dbconn OR Many ??

-> Multiple connections is actually waste of cpu usage

-> you can have only (1) connection object & utilize the same.

↳ leads to better efficiency of my system.

Need of SINGLETON object:

when creating same object is expensive and Not useful. [when object remains same across system]

When NOT to use SINGLETON DP:

when data for object can be changed

NOTE: SINGLETON objects are Immutable

A) IMPLEMENTING:

Many ways:

eg:

class DBConnection {
- URL
- uname
- Password

Client

main() {

- pool

}

}

As of now: clients can create multiple objects...

HOW TO RESTRICT OBJECT CREATION

PRIVATE CONSTRUCTOR ✓

Private constructor:

```
class DBConnection {
    - URL
    - uname
    - Password
    - pool

    Private DBConnection() {
    }
}
```

How do we create CI instance now ??

Private constructor would restrict object creation ...

Private members can be called by self class

↳

- can have Private constructor
- can have Public Method which can call PRIVATE CONSTRUCTOR:

```
class DBConn {

    Private DBConn() {
    }

    Public static createInstance() {
        return new DBConn();
    }
}
```

client:

```
DBConn.create...

DBConn.create...
```

Issue with above code ??

- creates Multiple objects, Not validates
- cannot be called at 1st place.....

*] SOLUTION:

(Approach 1)

- constructor always creates a new object
- Method: can do whatever it want [logic]

```
class DBConn {
    Private DBConn instance = null,

    DBConn() {  } const

}

Public static ..... createInstance() {
    if( this.instance == null) {
        return instance = new DBConn();
    }
    return this.instance;
}
```

solving Purpose (✓)

Q:) who can call this method as of Now ??

*) STEPS TO CREATE SINGLETON:

S1:) Make constructor Private

S2:) create static method - responsible for checking / creating object

Approach 1

single threaded ✓

Multi Threaded ✗

new -(2)

Approach 2:- Eager loading

Problem: only when (1st) time object not present

Thread$_1$ → Try to create object.
Thread$_2$ →

solution : EAGER LOADING

```
class DBConn {
(Immutable) static → DBConn instance = new DBConn();

    Private DBConn() {
    }

    Public static createInstance() {
        return this.instance;
    }
}
```

```
CUSTOM const {
this.x = x
}
client {
DBConn.createInstance();
}
```

Q:- When DBConn() Object is created ??

*] PROBLEM WITH EAGER LOADING:

singleton classes load at App start time

↳

(1) APP start will take more time.

(2) Passing Params is which are Runtime is Not possible

↓

*] SOLUTION [Approach 3]

-> we Need way to create singleton at runtime

⊕

concurrent Env as well.

BEST SOLUTION TO CONCURRENCY: LOCKS!

A3.) use locks in createInstance()

No_Objcd

→ Public sync static DBConn createInstance() {

.....

}

*) PROBLEM:

Performance !

→ Prev. issue was mostly when 1st time Object creation took.

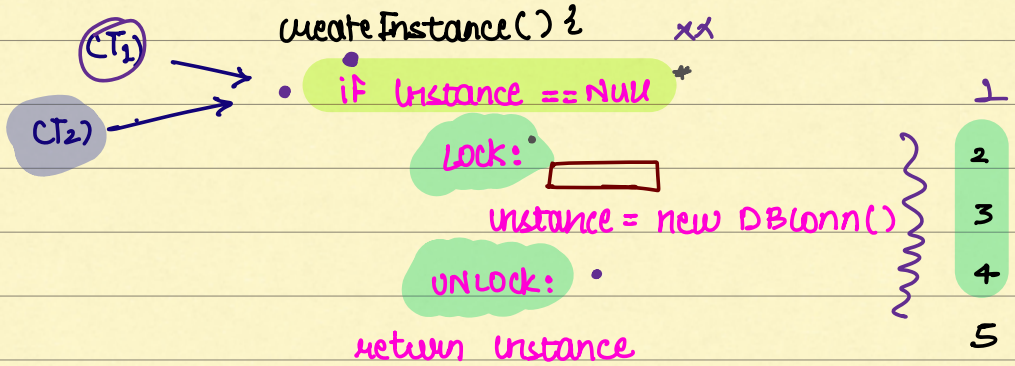
↳ Other cases were FINE

(*) Here: Everytime createInstance() method would take lock. (ALWAYS) ....

Performance is Reduced...

*] SLIGHT MODIFICATION FOR OPTIMIZATION:

(A4): Lock inside method....



- T_1 takes lock
- T_1 creates object
- T_1 releases lock
- T_2 takes lock....

Q.) PROBLEM:

More Objects are created...

SOLUTION → check again before creating object

```
createInstance() {
T1      if Instance == Null
T2          Lock: •
            if Instance == Null   ***
                Instance = new DBConn()
            UNLock: •
        • return Instance
}
```

1 2 3 4 5 6

**

known as DOUBLE CHECK LOCKING

- best way to implement locks in Multithreaded Prod. Env.
  → where Perf. is also CRITICAL

steps:

1.> create Pv const
2.> create static method
3.> double check for instance

Singleton using ENUMS → Homework ?!

DEMO