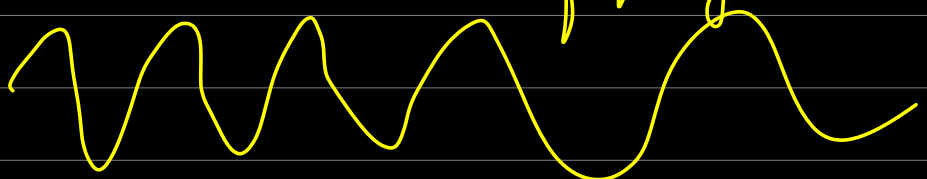
Agenda (Will start at 9:10PM)

→ Intro to Design Patterns
→ Type of Design Patterns
→ Creational Design Pattern
  → Singleton
  → Builder
  → Prototype } Next Class
  → factory

What are design pattern ⇒ Something that occurs frequently



Software design

→ Well established solution to common software design problems

GoF ⇒ Gang of Four : Design Pattern ⇒ 23 diff DP

~10 design patterns ⇒ Very common in real life
⇒ Very common in interviews

Why learn DP

① Shared vocabulary
② Saves a lot of time → Adapter
③ Interviews

Types of Design Patterns

→ Object oriented Design

→ Sol^n to common problems in OOD

Object

① Creational Design Patterns

→ How will an object be created } new

→ How many object be created

② Structural Design Patterns

→ How will a class be structured

→ What attributes will be there in a class

→ How will a class interact with other classes.

③ Behavioural Design Patterns

→ How to code an action

Design Patterns

→ Implementations design principles

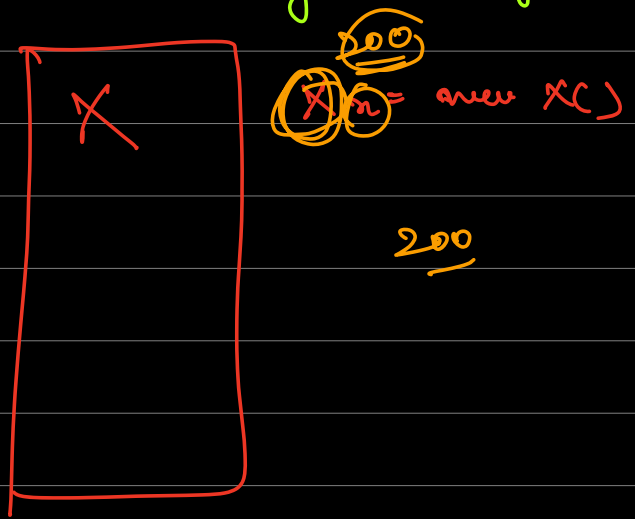
CREATIONAL DESIGN PATTERNS

- Singleton
- Builder
- Factory
  - Simple factory
  - Factory Method
  - Abstract factory
- Prototype

Singleton Design Pattern

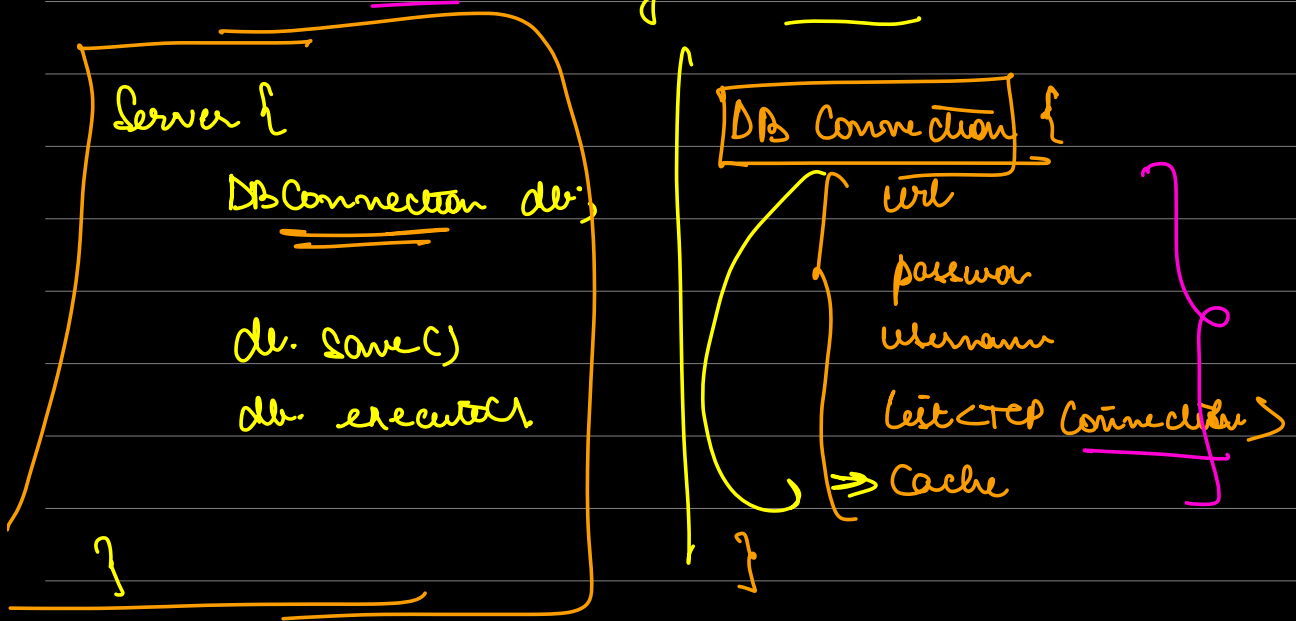
- Def^n
- Problem Statement (Why?)
- How (How to implement)
- Pros
- Cons

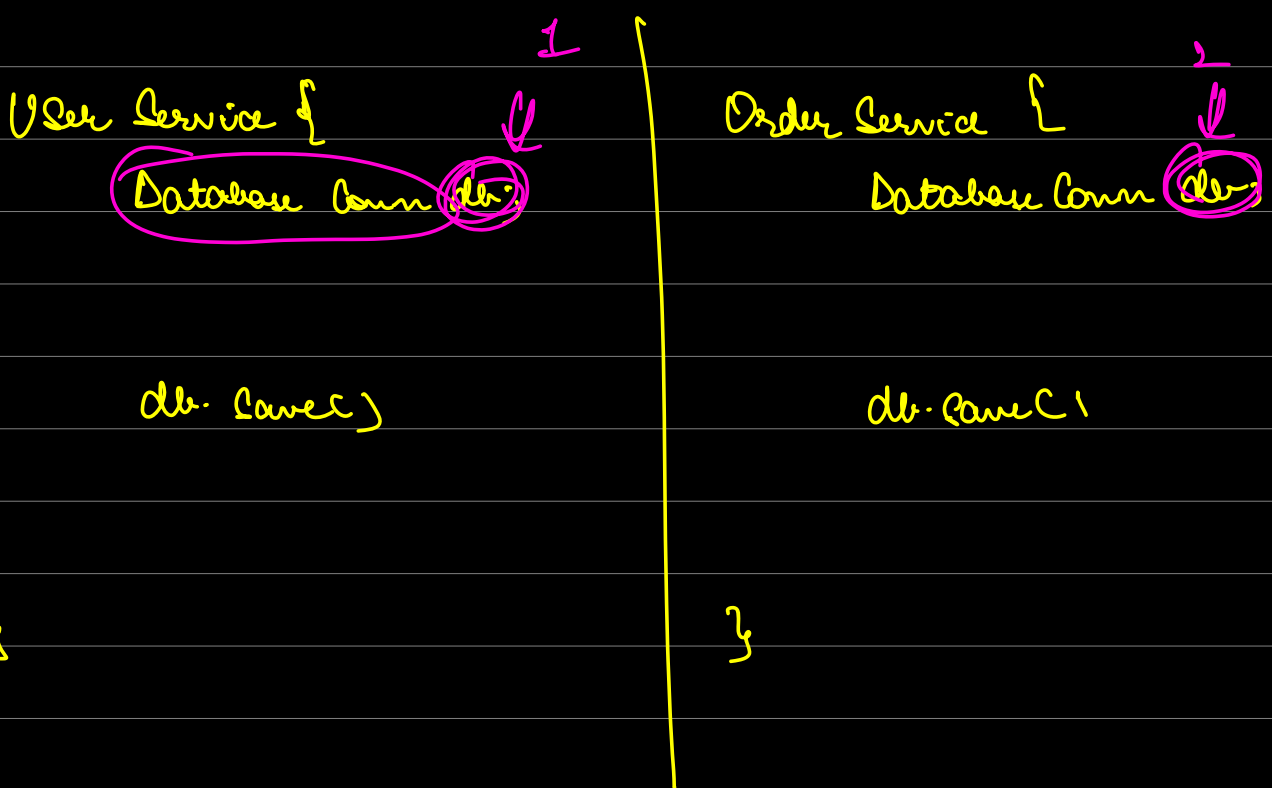
Def^n → Allows you to create a class for which only one object can be created



Why?

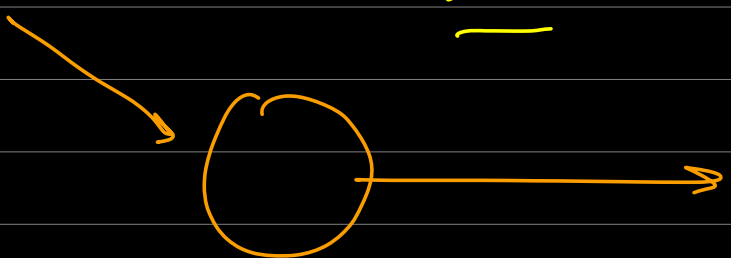
① A class which is having a shared resource behind the scenes

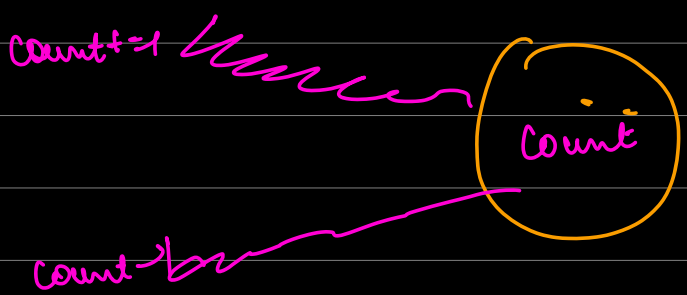


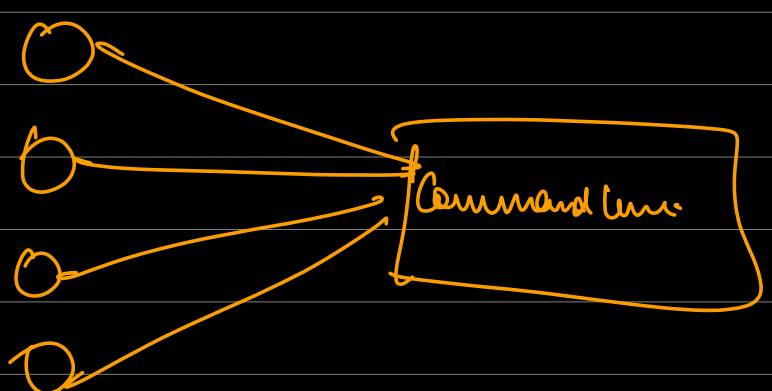


Multiple DB Connection are nothing but a waste of CPU resources

Logger → Log/Print information to the command line.



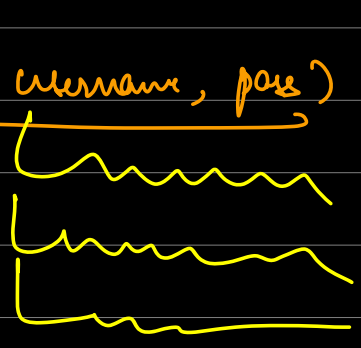


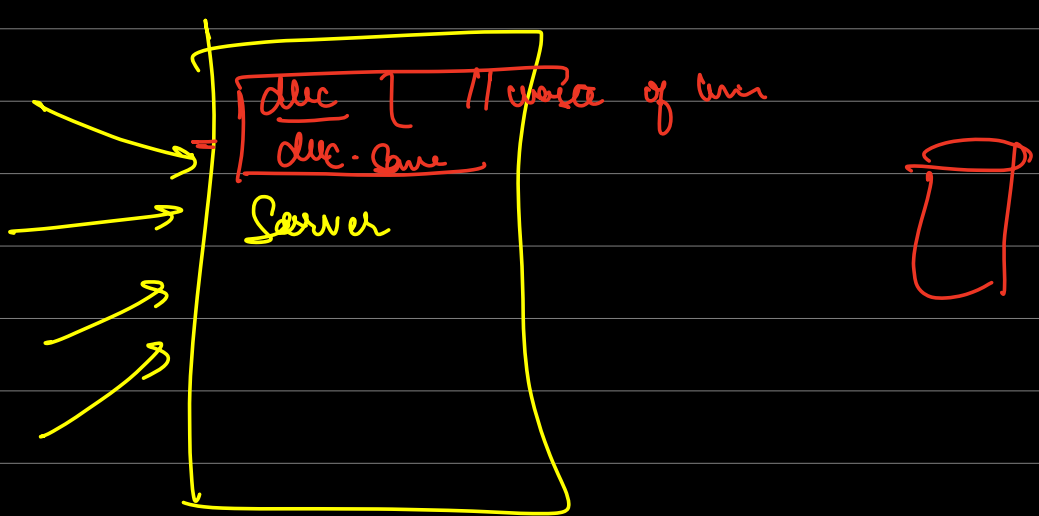


(2) Creation of an object is expensive.

{ to create a new dbc, what all do we need to do?

DBC (host, username, pass) → Creating a new obj is slow.





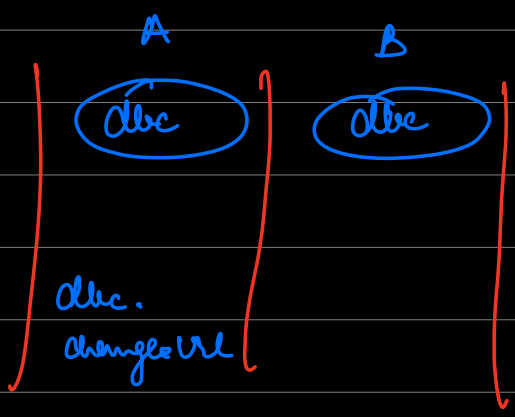
Database Connection {

url

changeURL() {

}

}



⇒ Singletons are always immutable

What is Singleton: Class for which only one object can be created

Why?

ⓐ Shared Resource

ⓑ Creation of an object is expensive and only 1 object is needed

ⓒ Common Sense

How to implement Singleton

```
class DatabaseConnection {
    String url;
    String username;
    String password
    List<Connection> pool;
}
```

⇒ DBC dbc1 = new DatabaseConnection();
DBC dbc2 = new DatabaseConnection();

⇒ Till the time a class has a constructor, can it be singleton?

↓

Constructor always creates a new object

```
Class DatabaseConnection {
    {
        private DatabaseConnection() {}
    }
}
```

⇒ :(Now I can't even create 1 instance

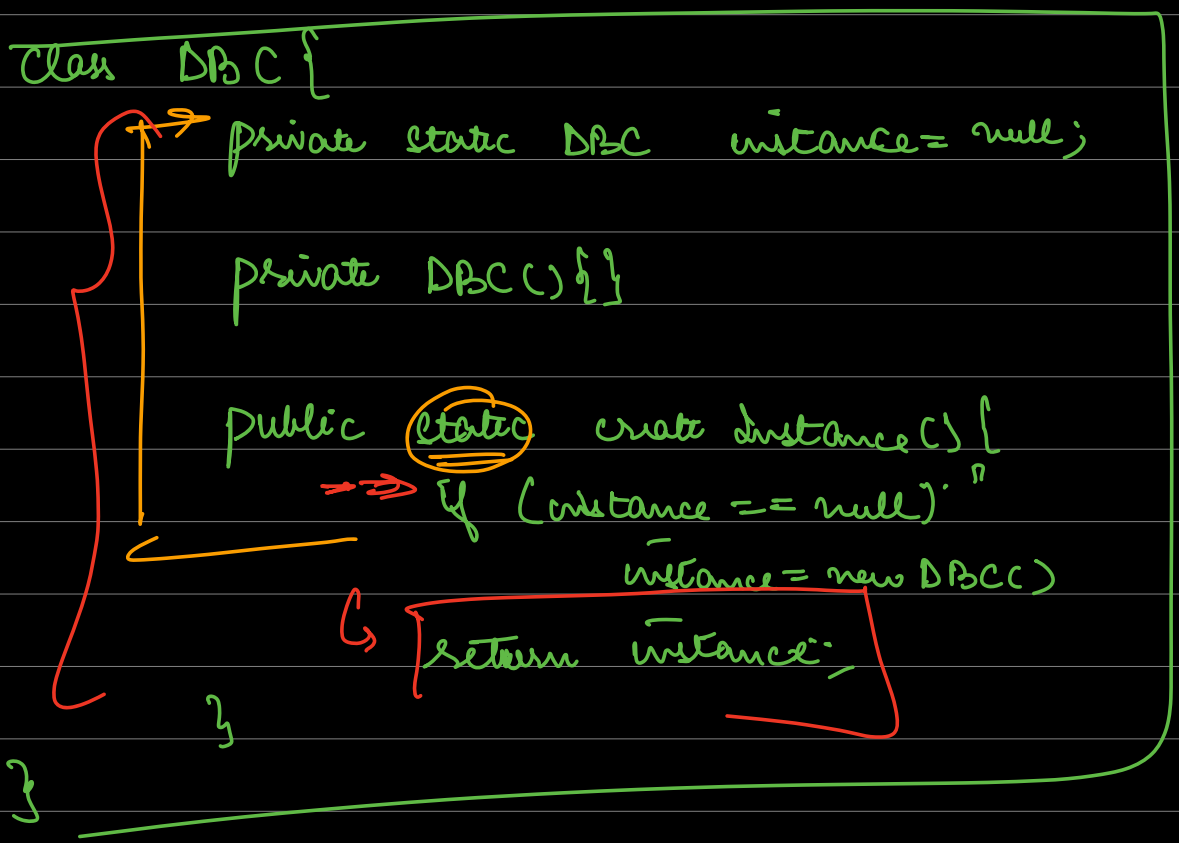
PS :

(1) We need to ~~dec~~ keep constructor private
(2) But need to create one object
(3) But to create an obj I need to call cons

⇒ Private Members are still accessible inside the class.

```
Class DatabaseConnection {

    private DBC() { }

    static
    public DBC createInstance() {
        DBC dbc = new DBC();
        return dbc;
    }
}
```

DBC dbc = DBC.createInstance()
DBC dbc2 = DBC.createInstance()



Steps

1. Make constructor private
2. Create a static method to create an instance
3. Static method checks if it has already created. If yes → return. Else create new.

DBC { default value }

DBC() {

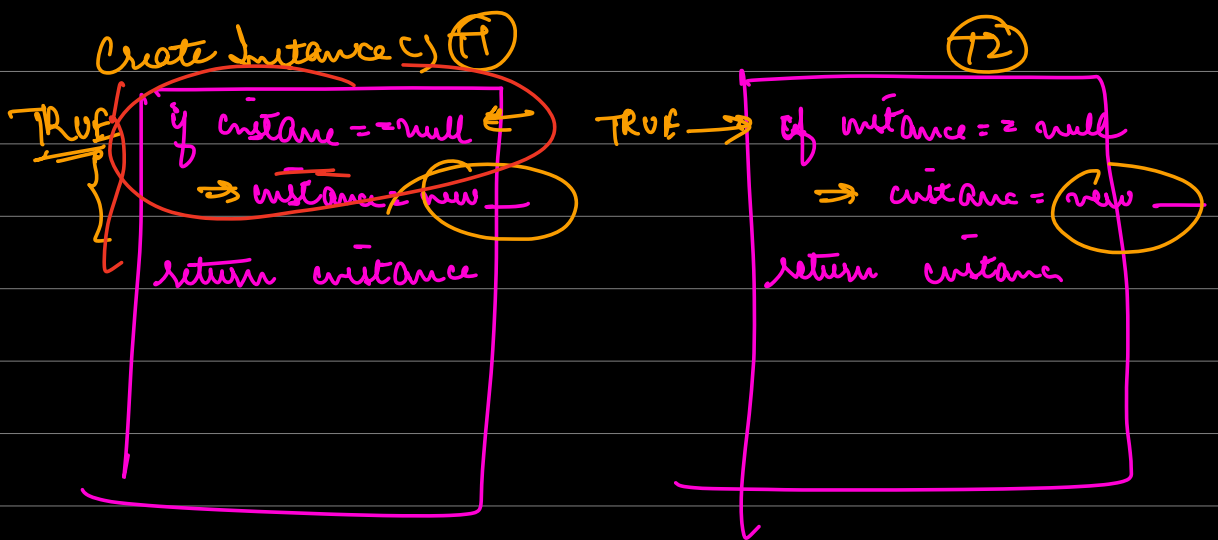
If instance == null:

instance = DBC()

return instance()

}

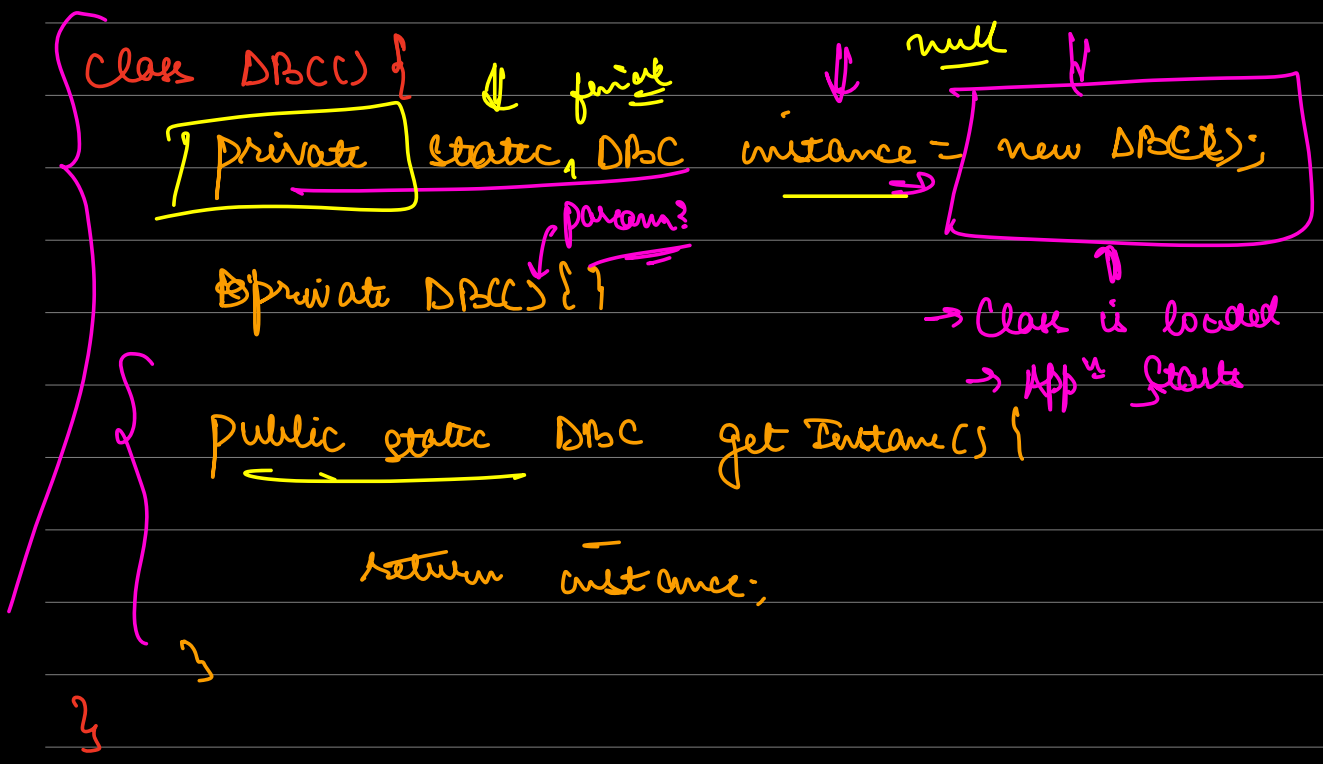
Constructor can't have return.



⇒ This is prone to error when the object is being created the first time

Singleton in concurrent env

Early loading / Eager Execution



Con ① App^n startup time will increase.

② Can't give variable config while creating the ferist obj.

```
Class Logger {
    [redacted]

    private Logger (String env ) {
        if env == prod

            else: ____

    }

} public static getInstance(env) )

    }

}
```

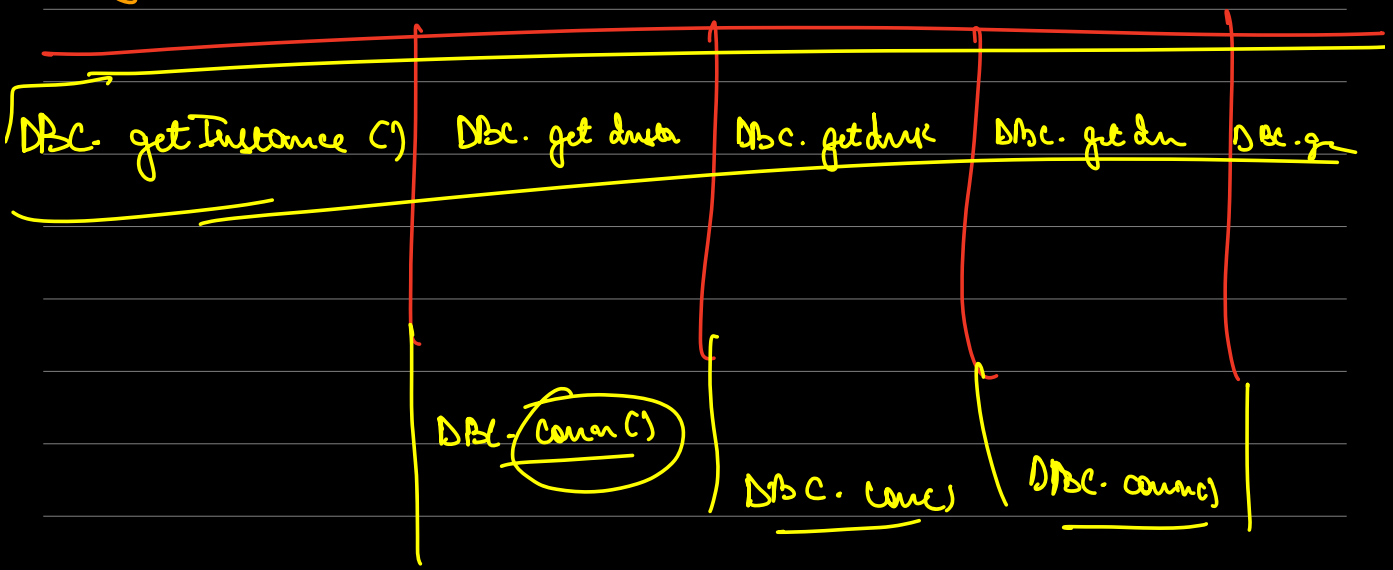
---

① Create Singleton at AT

② Have it work fine in Conce environment

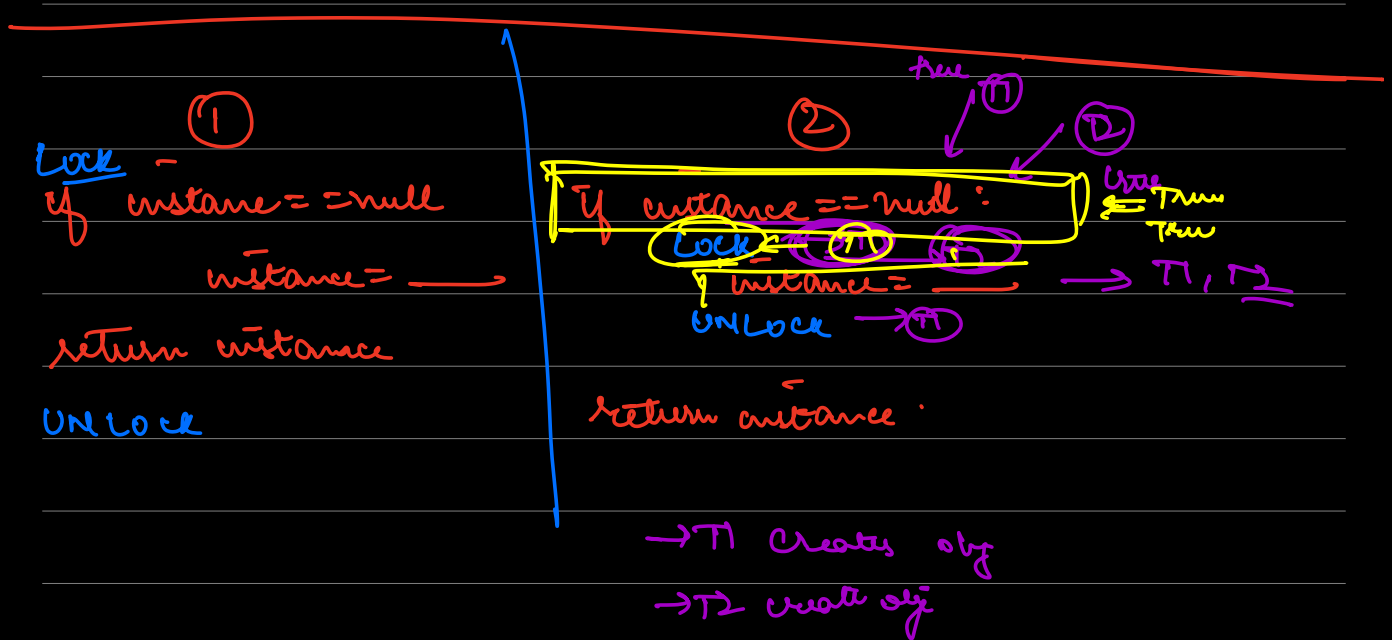
⇒ Lock!

```
class DBC {

    private static DBC instance;

    private DBC() {}

    public synchronized static DBC getInstance() {
        if (instance == null)
            instance = new ...
        return instance
    }
}
```

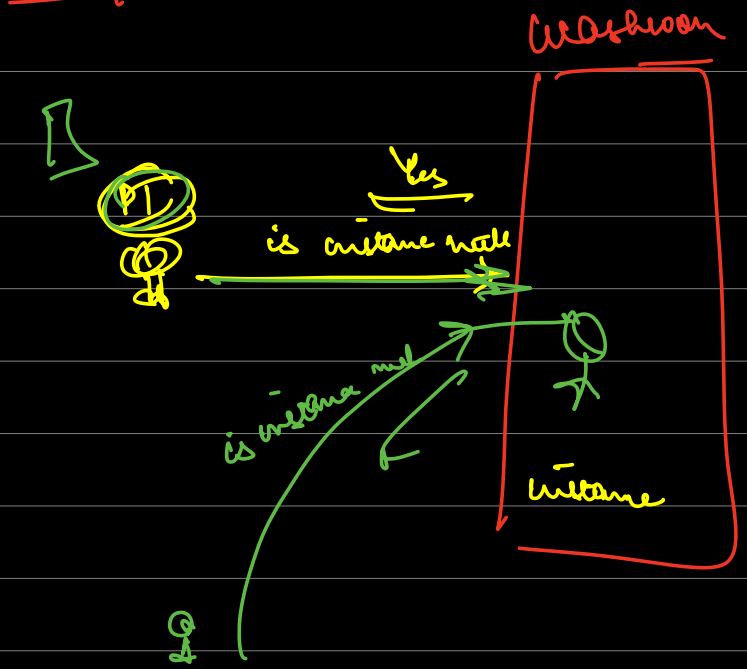


⇒ Perf is slow!

```
Class DBC {

    private DBC(){}

    Public synchronized static getInstance(){
        if (instance == null)
            instance = new ___

        return instance;
    }
}
```



```
if (instance == null) {
    LOCK
        if (instance == null)
            instance = new DBCC();
    UNLOCK
}

return
```



DOUBLE CHECK LOCKING

→ Best way to implement in prod environment where perf is also critical

1. Check without lock
2. If things are bad, check with lock

Lock free

① → if instance == null : ← false

99%

lock.lock()

if (instance == null) {

instance = new instance;

}

lock.unlock()

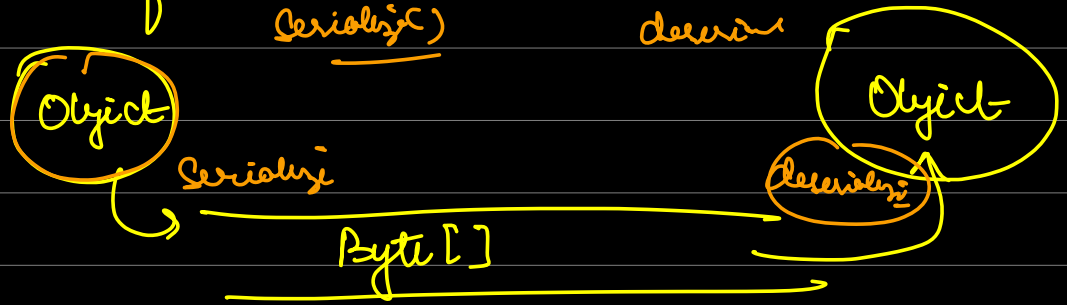
② → return instance;

Break till 10:50PM

→ dump singleton

Double check locking works against concurrency.

But, it fails to handle serialization.



Asgn

Enum : How to implement singletons using Enum

Also solves for Serialization

-> Java Specific

Pros =

1. Resource Efficiency
2. Creating new obj is inefficient => Database Conn

Singleton is like caching for future use

Cons

1. Difficulty testing a class that is singleton

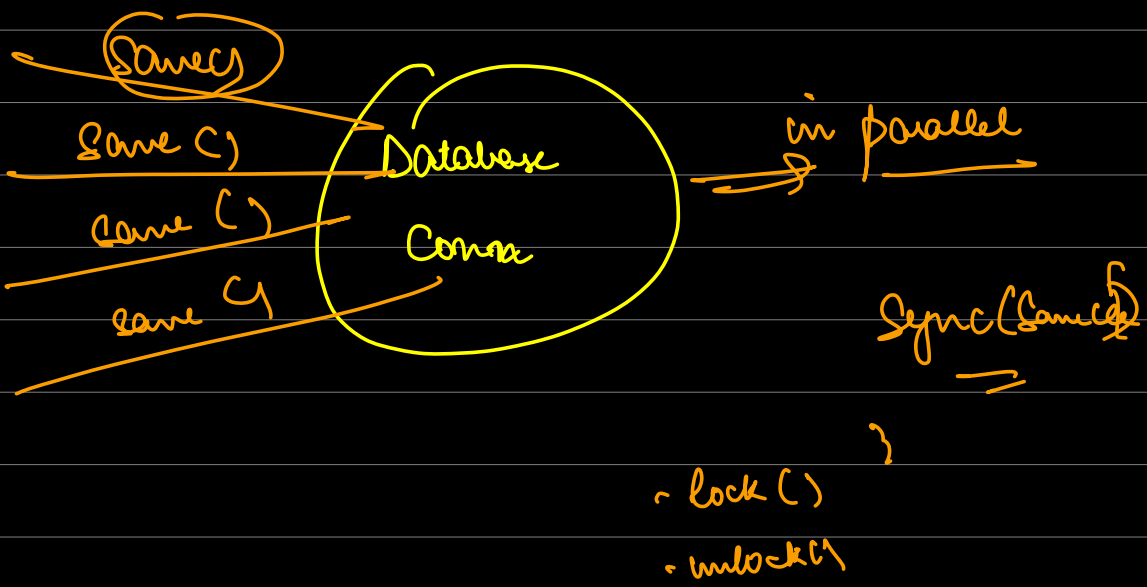
Mocking the dependencies

```
A(B, C) {
}
```

F

```
A(MockB, MockC) {
}
```

```
A(MockB2, MockB3) {
}
```



Adapter Design Patter