Agenda

Intro

Types of dp

Singleton

What are design patterns?

- design → software
- patterns → frequently occuring

well established sol"s of some frequently occuring software design problems.

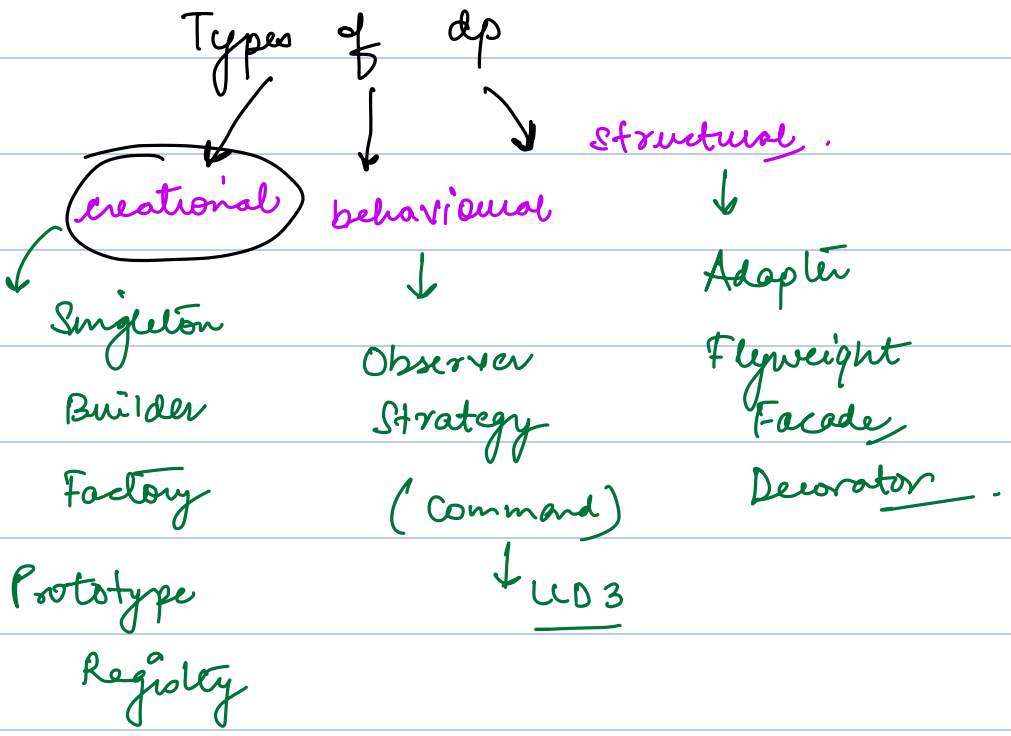
Why to learn dp?

- shared vocabulary
- save dev time.

~23 → gang of four.

~10 design patterns

① dev ② interview

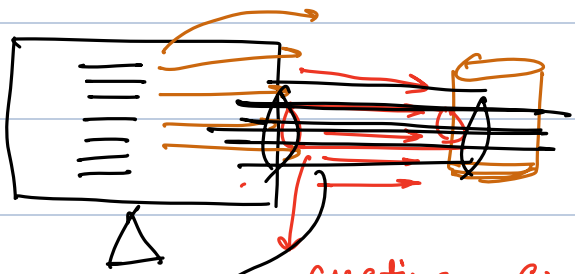


Creational :- deals with problem statements around creation of objects.

Singleton Design pattern

↓

when you want to enforce only a single Obj of a class.



creating a conn^n is costly

class DBconnPool {

url = ——————;

one object creation

}

ThreadPool

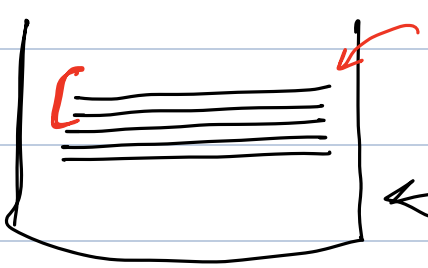
DBconnPool

Logger

ConfigurationManager

CacheManager

SpringBoot



← bucket

one

```
class dbconn {

}
```

```
Dbconn dbconn1 = new Dbconn();
Dbconn dbconn2 = new Dbconn();
```

constructor.

private

```
class dbconn {
    private dbconn() { — }
    static
public  dbconn getInstance() {
        retu new dbconn();
}
```

```
dbConn.getInstance();
```

```
class Dbconn {
private static DBconn instance = null;

private Dbconn (){
  —
}

static DBconn getInstance (){
  if( instance == null)
  {  instance = new DBconn();
  }
  return instance
}
}
```

Lazy

we should only create the obj once when it doesn't exist!

private const → static getInstance → instance

instance → create obj only once & store it

IS IT THREAD SAFE ?

inst = null

T1

```
getInst(){

    if(inst==null)
    {
        inst = new DBcom();
    }

    return inst;

}
```

T2

```
getInst(){

    if(inst==null)
    {
        inst = new DBcom();
    }

    return inst;

}
```

```
class Dbconn {
private static DBconn instance = null;

private Dbconn (){
  —
}

static DBconn getInstance (){
  if( instance == null)
  {
    instance = new DBconn();
  }
  return instance
}

}
```

new DBconn();

class is loaded

Eager instantiation.

1. App start time
2. we might never use it.
3. custom config to create obj

Option II — make the method (getI) synchronised

performance is bad

It will block the threads even after obj created.

— use locks

(I)

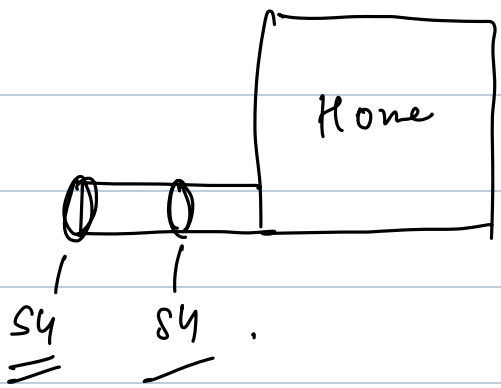
```
getInst(){
    if(inst==null)
    {
        lock();
        inst = new DBcom();
        unlock();
    }
    return inst;
}
```

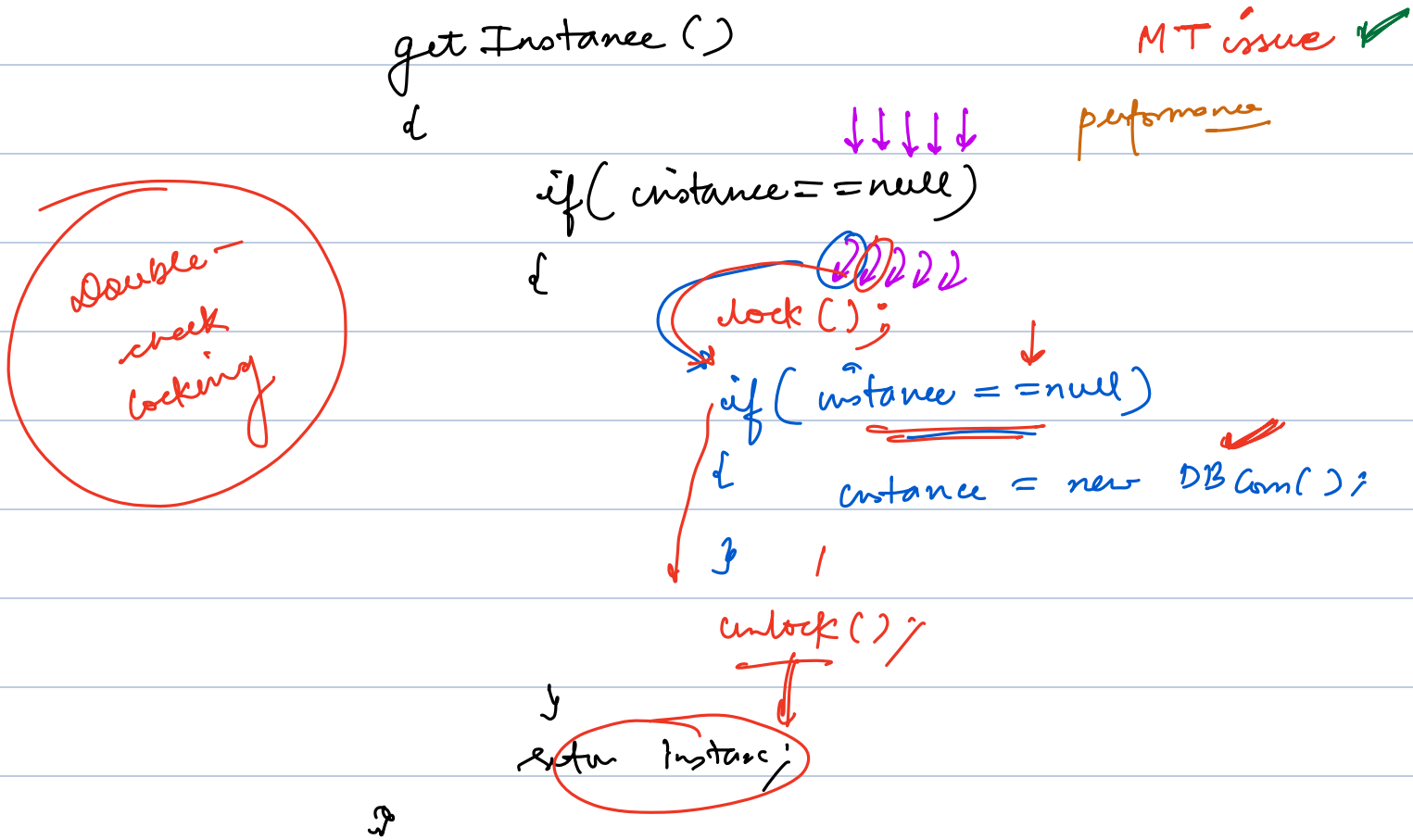
this will not even work.

(II)

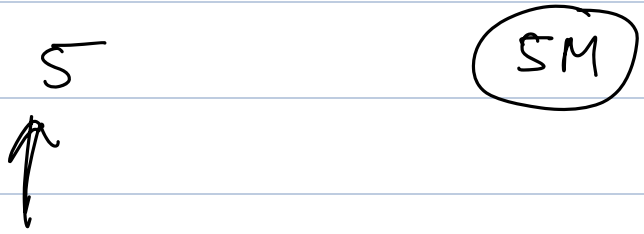
```
getInst(){
    lock();
    if(inst==null)
    {
        inst = new DBcom();
    }
    unlock();
    return inst;
}
```

Is it any way better than synchronized?





- V0 : Lazy
- V1 : Eager
- V2 : Synchro
- V3 : Double-Check



https://dzone.com/articles/java-singletons-using-enum
https://www.digitalocean.com/community/tutorials/java-singleton-design-pattern-best-practices-examples
https://www.baeldung.com/java-patterns-singleton-cons
https://baeldung.com/java-singleton-double-checked-locking
https://docs.oracle.com/javase/tutorial/java/javaOO/enum.html