Day 07 Design Patterns : Singleton.

① Design patterns.

② Singleton

- Lazy loaded.
- Eagerly loaded

|
- Sync
- DCL.

① OOP
- Abstraction.
  - Encapsulation
  - Inheritance
  - Polymorphism

② SOLID

③ Design Patterns
- Solutions to common problems.

Gang of four ☆

① Creational design pattern.

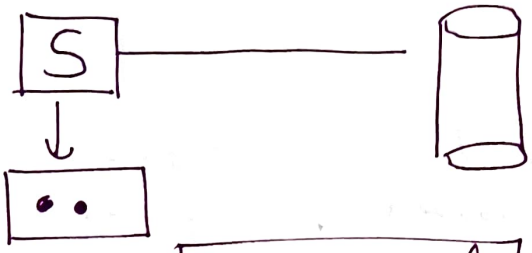
→ How do you create immutable objects?

→ How do you create objects without having a dependency on concreate classes?

② Structural
③ Behavioural.

DRY → Dont repeat yourself.

Connection = overhead



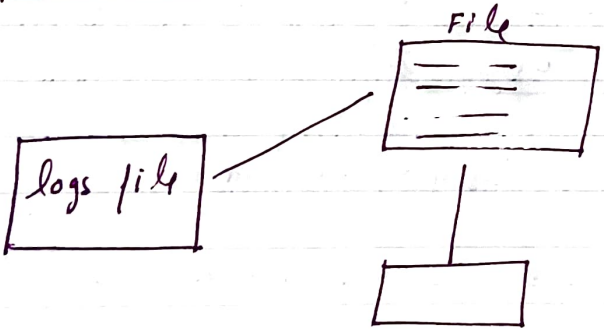
Connection pool

class Connection Pool { → Shared resource.

collection < Connection >

}

○ Only have one instance ☆



Configuration

why not multiple instances?

① Performance

② Inconsistency issues

{ dbURL →
SecretKey →
API Key →
env → local

} dev.

singleton DP

Spring instances
= Beans
=> singleton

Tom calculator {

}.

---

class ConnectionPool {

instance

Private ConnectionPool () { }

public CP getInstance () {

if (instance) { return i }

return new CPC);

}

}.

① 0 instances.

step 1 – construction hiding

② Global access point step-2

```
if (instance created) {
    return existing
}
return new instance
```

☆

```
if (instance == null) {
    this.instance = new connection P();
}
return instance;
```

→ Step 1 - Construction hiding.

Step 2 - Global access point - getInstance().

- Step 3 - Add local variable `instance`

Step 4 - if instance is null, create & set to instance.
else
return.

```
Connection Pool {
    instance;
    getInstance () { }
}.
```

On demand creation
Lazy loading

Con - cold start

vs

Eager creation

Multithreading

↳ Sync method } performance.
↳ Sync block.
↳ Double checked locking

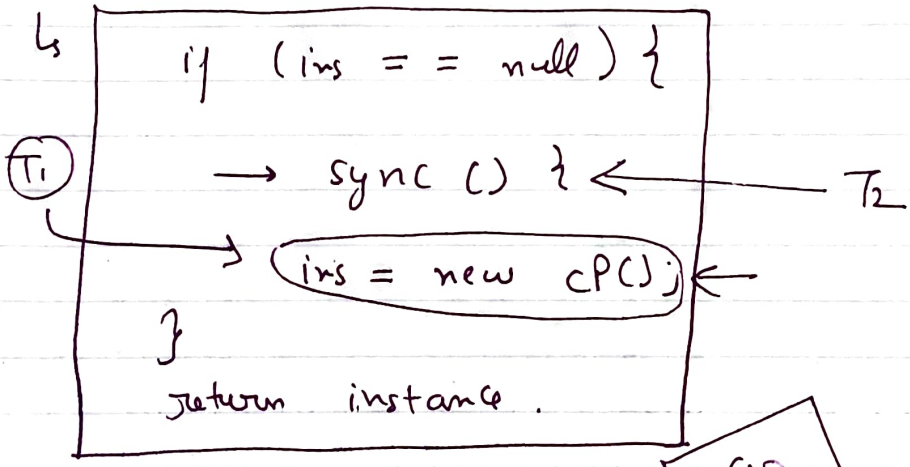
```
if (ins == null) {
```

||

||

```
} return
```

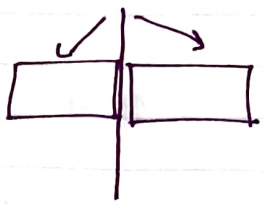
↓




```
→ if (i == null) {
      sync () {
          if (i == null) {
              i = new CP();
          }
      }
}
```

cloning

- clone () ⟶



@ override

clone () {

throw an error

}