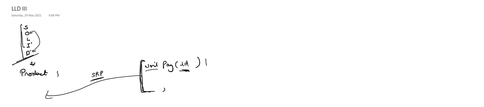
LLD III
Saturday, 25 May 2024 9:08 PM




```
class PaymentProcessor {
  void Pay(Str PId) {
    SqlProductRepo n = new SqlProductRepo();
    Product p = repo.getProductById(PId);
    // Process Payment.
}
```

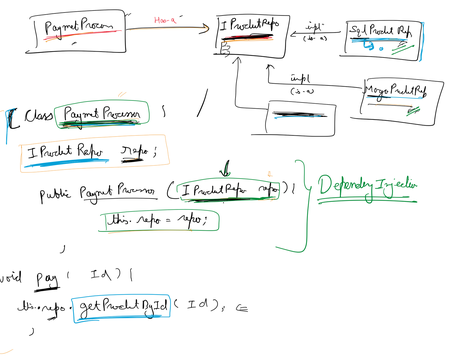
```
class SqlProductRepo {
  public Product getProductById(Str Id) {
    CRM.getProductById(Id);
  }
}
```

Spring Data JPA

Rails → Active Record

```
class MongoProductRepo {
  Product Product getProductById(Id);
}
```

Sql → mongo

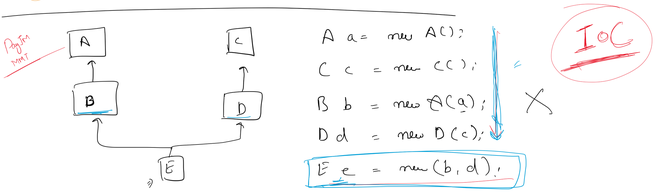



```
Class Seller {
  void Sell(Sellable s) {
    S.getSellingPrice();
  }
}
```

```
Class ProductRepoFactory {
  IProductRepo Create(repo) {
    if (repo == "Sql")
      ret new SqlProductRepo();
    if (repo == "mongo")
      ret new MongoProductRepo();
  }
}
```

abstract

```
PaymentProcessor p = new PayMntProcessor(ProductRepoFactory.Create("Sql"));
```




Design Pattern

1. Creational
2. Structural
3. Behavioral

Creational DP

Q Given a class. Restrict the no. of objects to 1.

```
Class SqlProductRepo {
  Static SqlProductRepo instance = null;
  Private SqlProductRepo () {}
}
Public Static SqlProductRepo getInstance () {
  if (instance == null) {
    instance = new SqlProductRepo();
  }
  ret instance;
}
```

Singleton

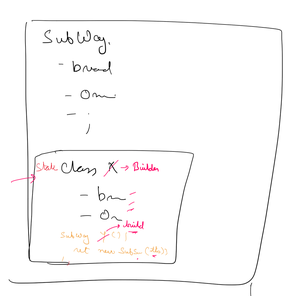
Simple Threaded

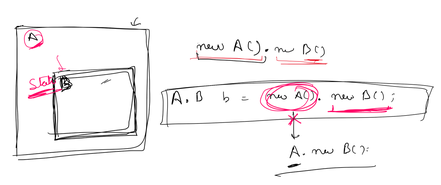
2 Builder

Easier.
→ Class with multiple attributes
→ Needs to create immutable objects (No public setts allowed)
→ Need flexibility of passing any parameter val in any order.

```
class SubWay {
  Str bread;
  Str Onion;
  Str tomato;
  Str sauce;
  ⋮
public SubWay(X x) {
  this.bread = x.bread;
  this.onion = x.onion;
  this.tomato = x.tomato;
  ⋮
}
```

```
Class X {
  Str bread;
  Str Onion;
  Str tomato;
  ⋮
  X setBread(b) {
    this.bread = b;
    ret this;
  }
  X setOnion(o) {
    this.Onion = o;
    ret this;
  }
  SubWay y() {
    ret SubWay(this);
  }
}
```



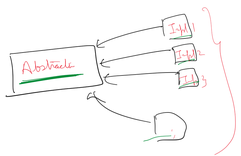




Client

```
X x = new X();
x.setBread(b);
x.setOnion(o);
⋮
SubWay s = new SubWay(x);
```

```
X x = new X().setBread(b).setOnion()
SubWay s = new SubWay(x);
```

```
SubWay s = new X().setBread(b)
           .setOnion(o)
           ⋮
           .y();
```

```
SubWay s = SubWay.X().setBread()
           ⋮
           .y();
           .build();
```

Obj of Builder
Builder()
sb.getInst()
Builder Class object
sb.toSubWay()
Subway obj