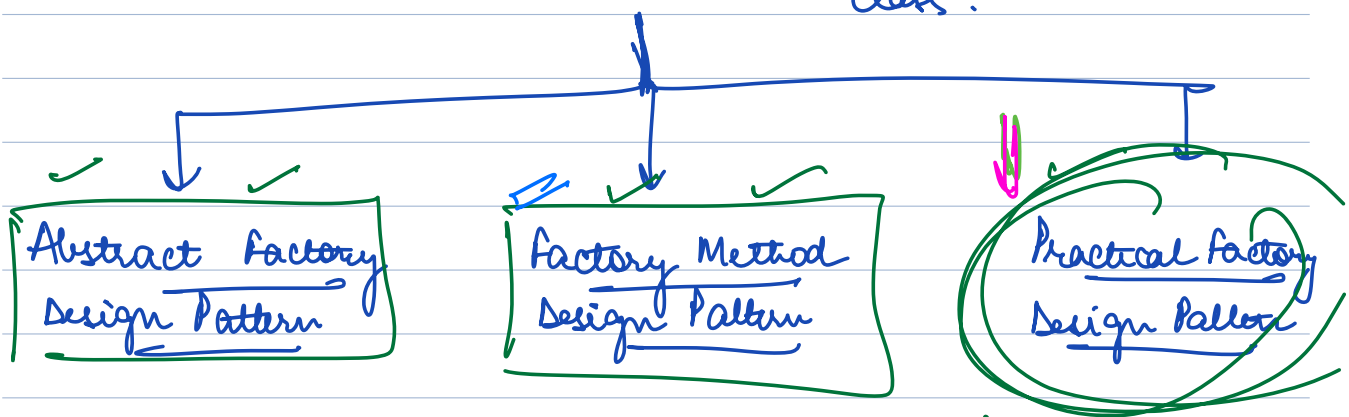
Factory Design Pattern

↳ Produce

↳ Something that produces something

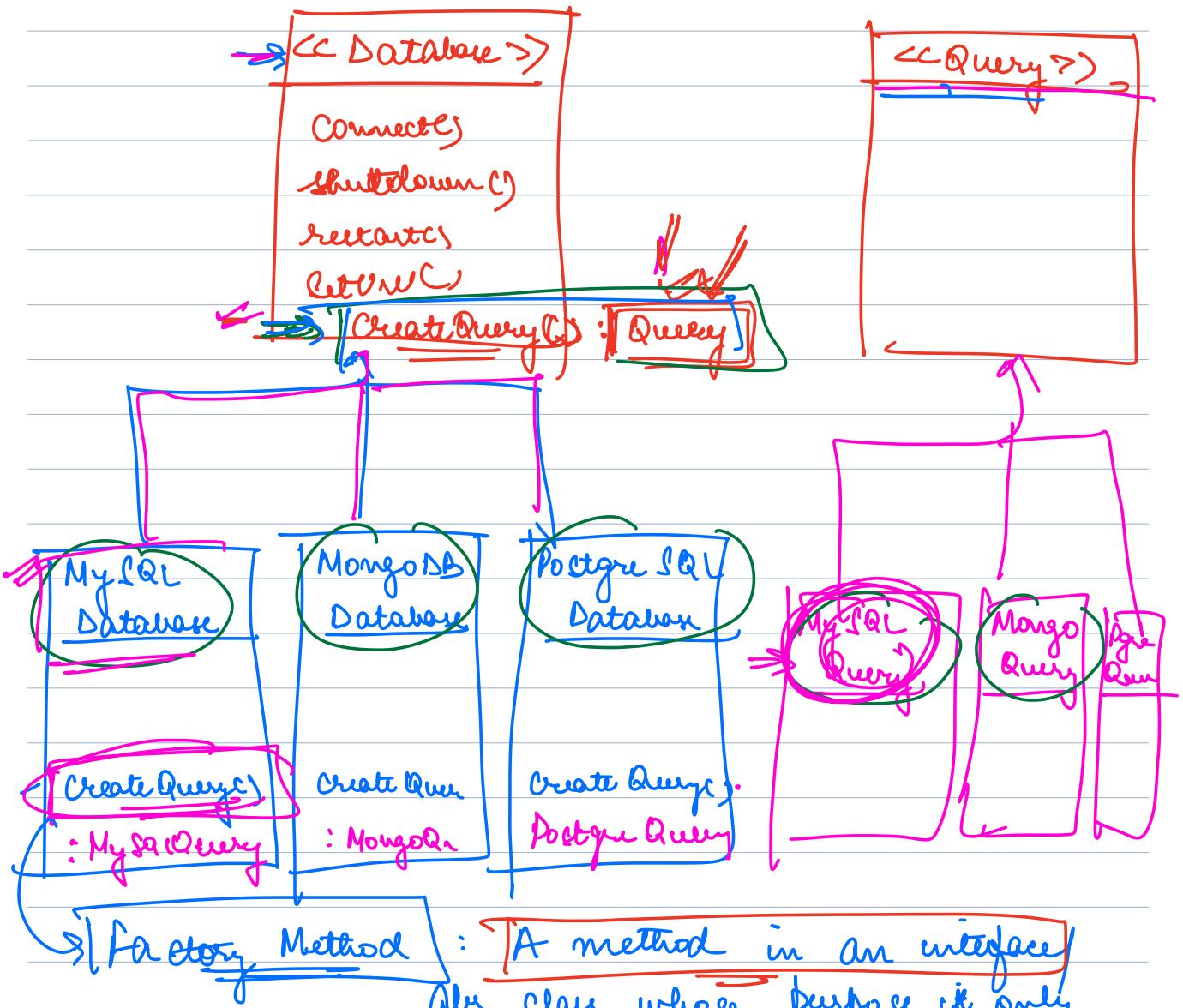
↓

An object of some class.



⇒ how factory is actually used in day to day lives.

FACTORY METHOD DESIGN PATTERN



Factory : Create an object of a corresponding type.

```
if (database type of MySQLDB)
    return new MySQLQuery()
else if (_____ PostgreSQLDB)
    return new PostgreQuery
```

```
Client {
    psvm() {
        Database db = ____ ;
        Query q = db.createQuery()
    }
}
```

write factory method

```
Query q;
if (db instance of MySQLDB)
    q = new MySQLQuery
else if (db instance of Postgre)
    q = ____
```

Info Factory Method

```
}
}
```

A {

attr1 = ~~new X()~~

attr2 = ~~new A()~~

A(attr1, attr2)

this. attr1 = attr1

this. attr2 = attr2

}

Database

Get URL()

Create Query()

Create Connection()

Get Username()

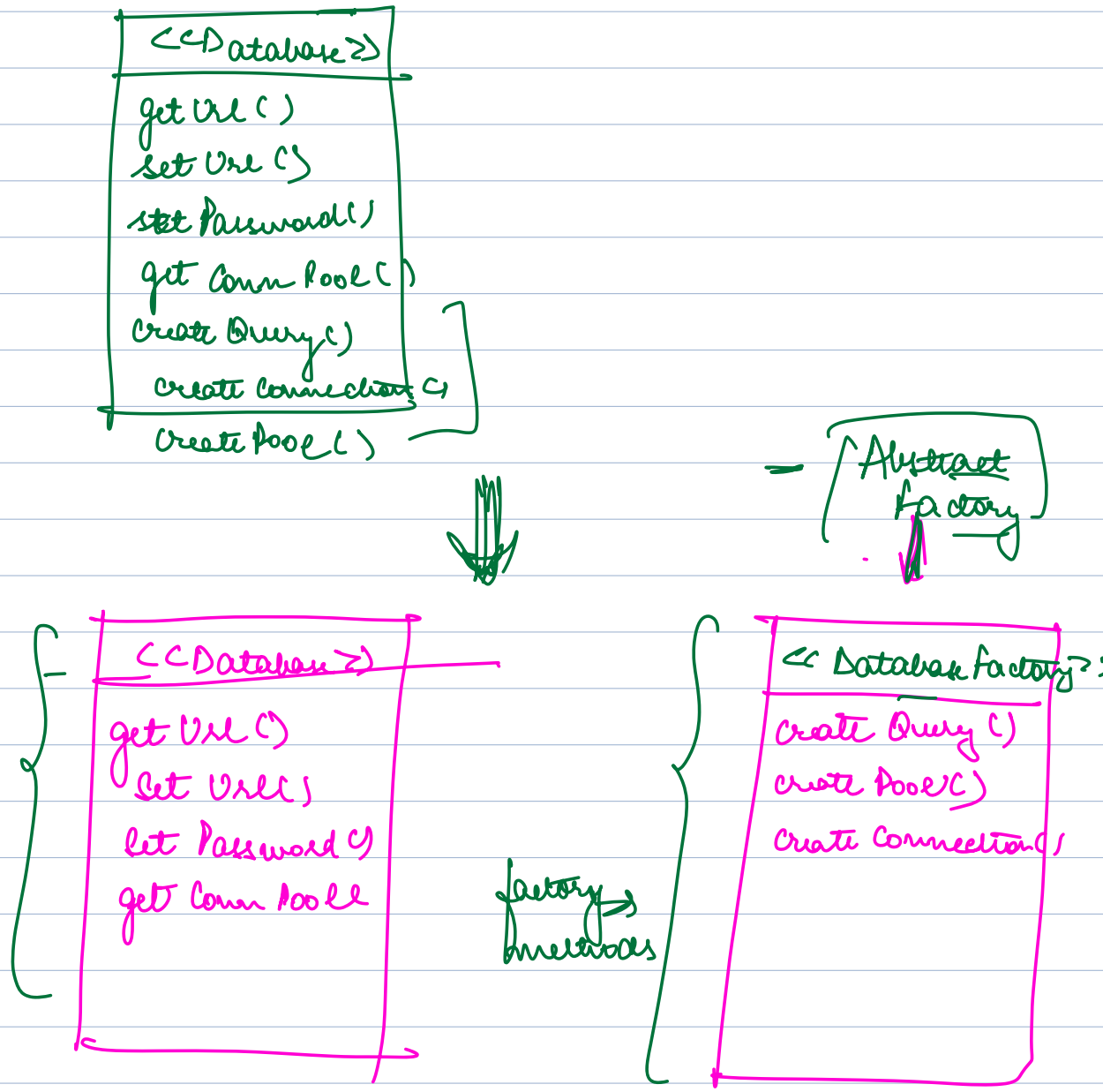
many more factory methods

⇒ Violates LRP

① Have methods needed to perform DB op^n

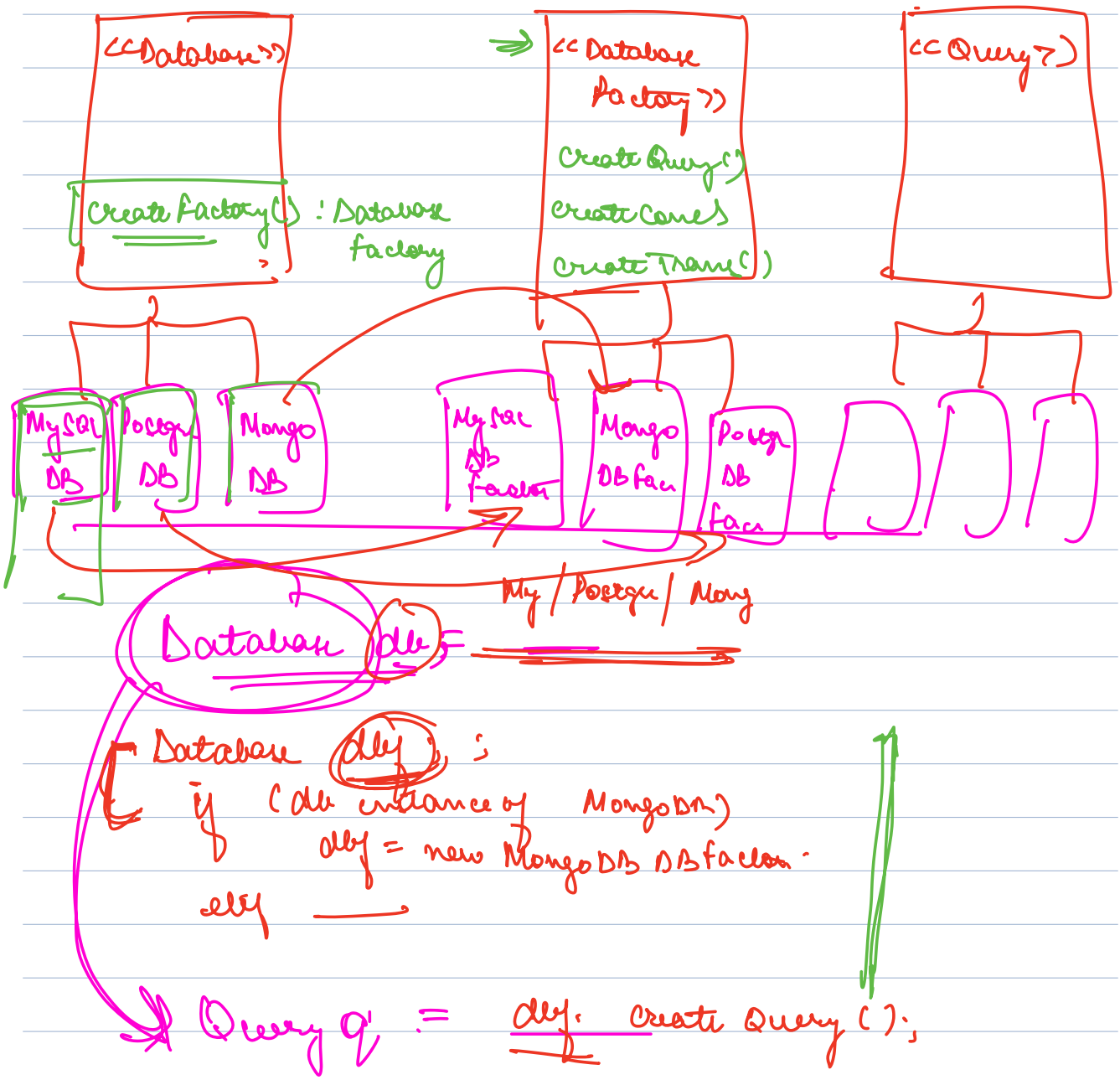
② Lot of factory methods

}




```
Client {
    psvm() {
        Database db = ____ ;
                      -    )

    }
}
```

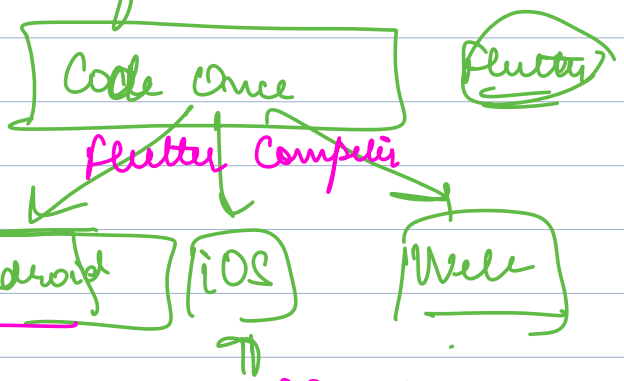



```
Database db;

Database factory dbf = db. createfactory()

Query q= dbf. create Query
```

Flutter ⇒ Cross Platform App^n creation framework



Android Button

Menu

Dropdown

Navbar

iOS Button

Men

Drop

Nav

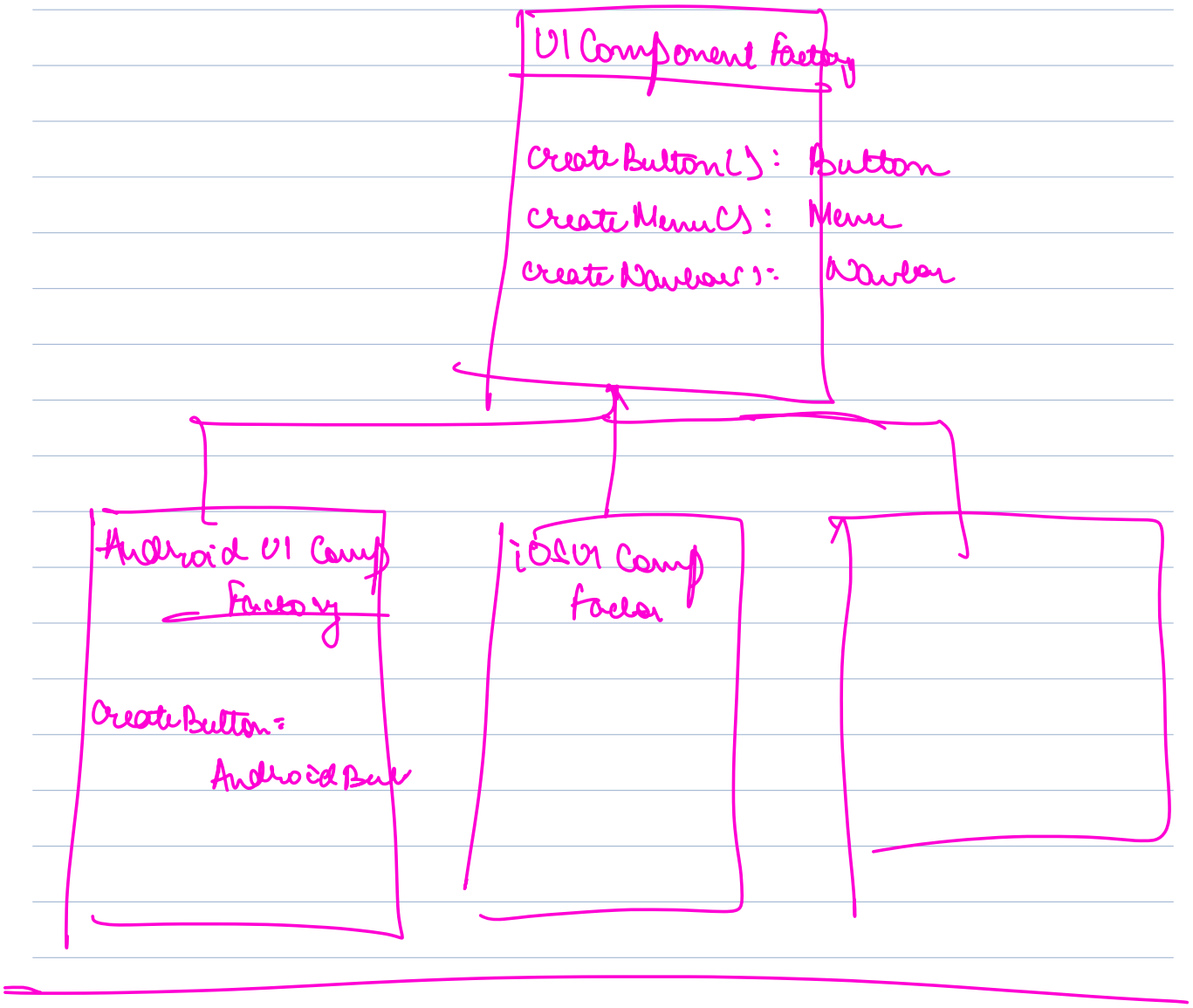
Dark Theme

Dark Button

Menu

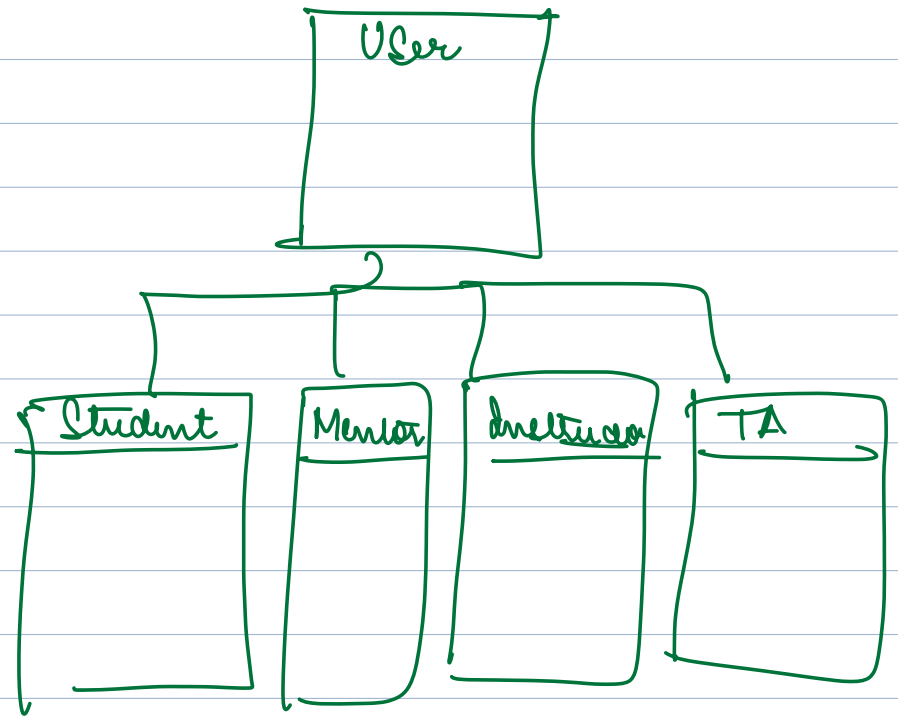
Dropdown

Navbar


UI Component factory
CreateButton(): Button
CreateMenu(): Menu
CreateNavbar(): Navbar
Android UI Comp Factory
CreateButton:
AndroidBut
iOS UI Comp factor

PRACTICAL FACTORY DESIGN PATTERN

→ Create an object of corresponding types (when something has multiple variations)



Client {

psvm() {

SAX

String inp = input();

User u;

if (inp == Student)

u = new Student

elif inp == ___

u =

elif inp == ___

O/C Violate

}

}

Practical Factory ⇒ Class whose purpose is only to give corresponding objects of a class

Class UserFactory {

// Methods that allow getting corresponding
// obj of user

static User getUserForString(String type) {

if (type == ___)

return new Student()

else if (type == ___)

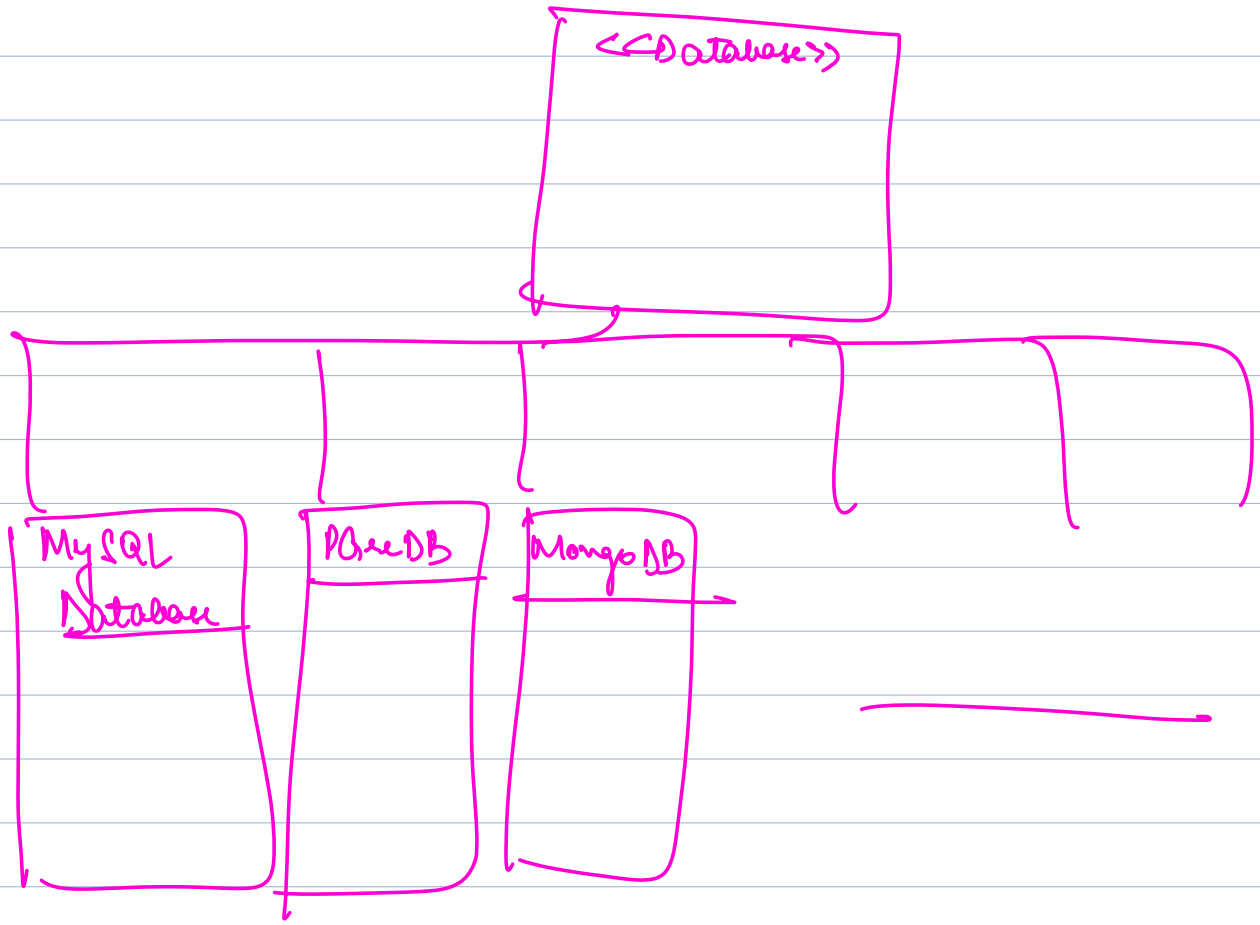
return new TA();

}

getUserFromEnum(UserType type) {

}

⇒ Whenever there is need to get an object of corresponding class



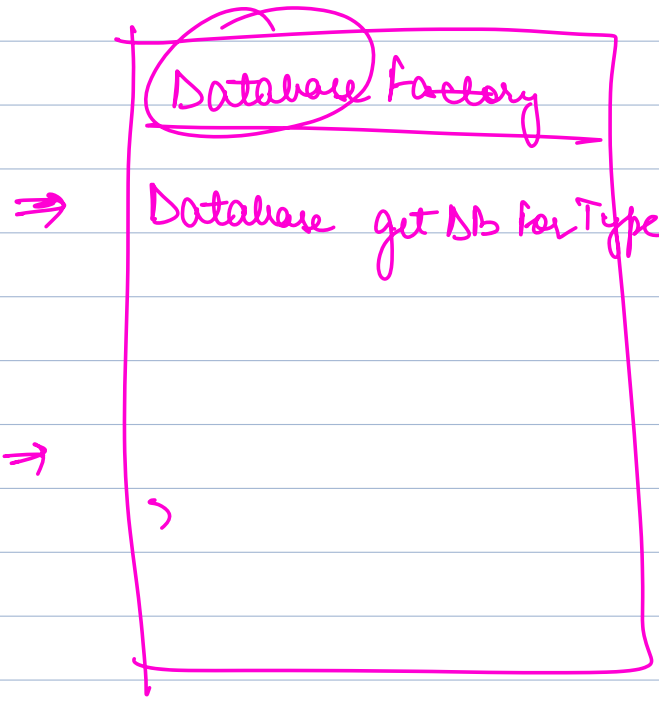
Client {

psvm() {

String url = ——

Database db = Database factory

.getDB of Type(url);



X Factory

Practical → corresponding obj of X

Abstract → Corresponding obj of another class related to X