LLD - Factory Design Patterns

Agenda

Factory

- Simple Factory
- Factory Method
- Abstract Factory

}

Repository (Github)

Factory

↳ to produce | create

Creational DP

— create objects

Factory

— Create multiple type of objects

Database

- MySql — MySql Database
- Postgres — Postgres Database
- Mongo — Mongo Database

MySql Database, Postgres Database, Mongo Database → DB

Given the type of DB, how can I create/get the object of the DB.

Email Service

sendEmail(Config config) {
  if (config.db == "mySql") {
    & db. - Type }

```
{
    db = new MySqlDb()
} else if (config.db == "postgres")
    { ... } else
        {
            db = new MongoDb()
        }

db.getUsers().sendEmail()

}
```

if-else hell

- maintaible X
- SRP X
- OC X
- DRY * not reusable

Database Factory ✓

↳ CreateDbForType(Type type){
switch (type) {

case MYSQL:

return new MySqlDB()

case POSTGRES:

return new Postgres()

};

return Interface

Simple Factory

- DRY ✓
- maintaible ✓
- SRP ✓

- OC

Email Service {

sendEmail() {

Db db = DatabaseFactory.createDb()

db.getUsers()

}

return new Database() x

return new MysqlDb()

Database

Program to interface

Database Factory

└ createDb

→ argument → type

→ return type → DB (interface)

Email Service

sendEmail

- Config & db: mysqlly
- Database Factory
  . create DBC
    config.ab

what?

- Create an object from

diff implementations on the

basis of a parameter

Db - MySql
   - Postgres

Tax calculation

- new
- old

why?

- Maintable
- Reusable

SOLID

How!

1. Create a common interface

2. Create sub classes of the

interface

3. Create a common factory class
   - ↳ static method
   - → accepts the parameter
   - → returns the subclass (Interface)

Downsides of Simple factory

→ OC → Factory

→ maintainable ✗

cyclomatic complexity

Factory Method

abstract

Database {

String host;

Int port;

MySqlDb

{

Type type;

connect();
terminate();

createQuery();

createQuery();

Query

Query

new Query()

Select * from Users;

mysql

postgres

Query

MySqlQuery

Database

Create Our

Query

Mongo
ca

MySql

co

Postgy

co

MySql

Ours

Postgres


```
class MysqlDb extend Database{

    Query createQuery() {
        return new MySqlQuery();
    }

}

class PostgresQuery {

    createQuery() {
```

return new

S

MySql

Database db = new PostgresDb();

Query query = db.createQuery();

→ CassandraDb

→ CassandraQuery

Create Query

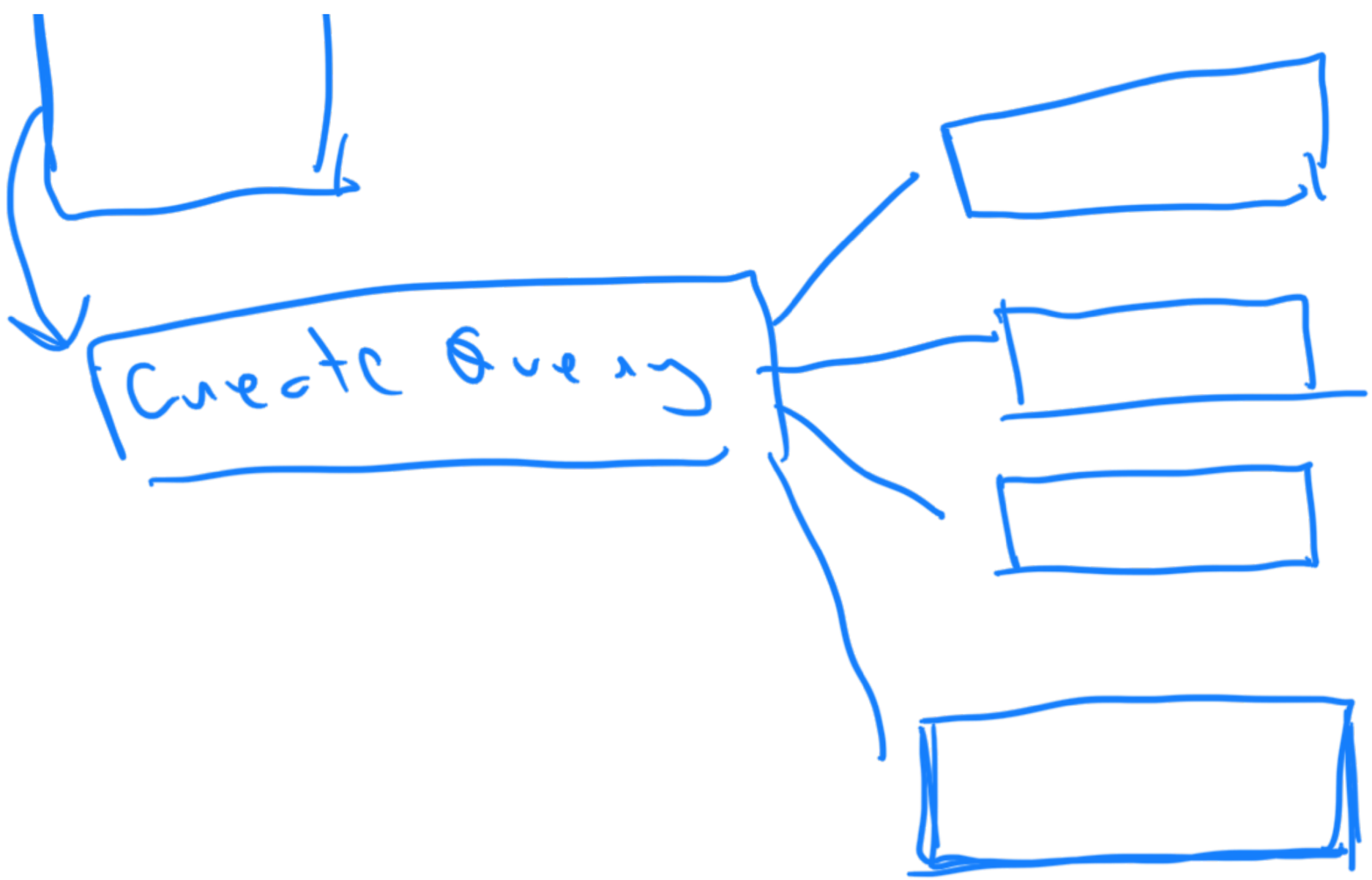
↳ Factory Method

DFS

Create Qu (type){

}

Database



Simple factory vs Fn

→ Database factory

↳ DB

Factory Method

Database

↳ createQuery()

↳ Query

MySqlDb

How?

① Create relevant interface | abstract class (DB, query)

② Create the relevant subclasses

MySqlDB | MySql and Postgresql

③ Add a abstract method to create the object in parent class

DB

Query

↓

create Query()

(4) In all subclasses,

return new object

MySqlDb{

createQuery(){

return new MySqlQuery()

}

Abstract

Database → Fig

att. [
utility [
} family objects

→ fn1 [create Query
→ fn2 [Create Index
→ fn3 [Create Views
}

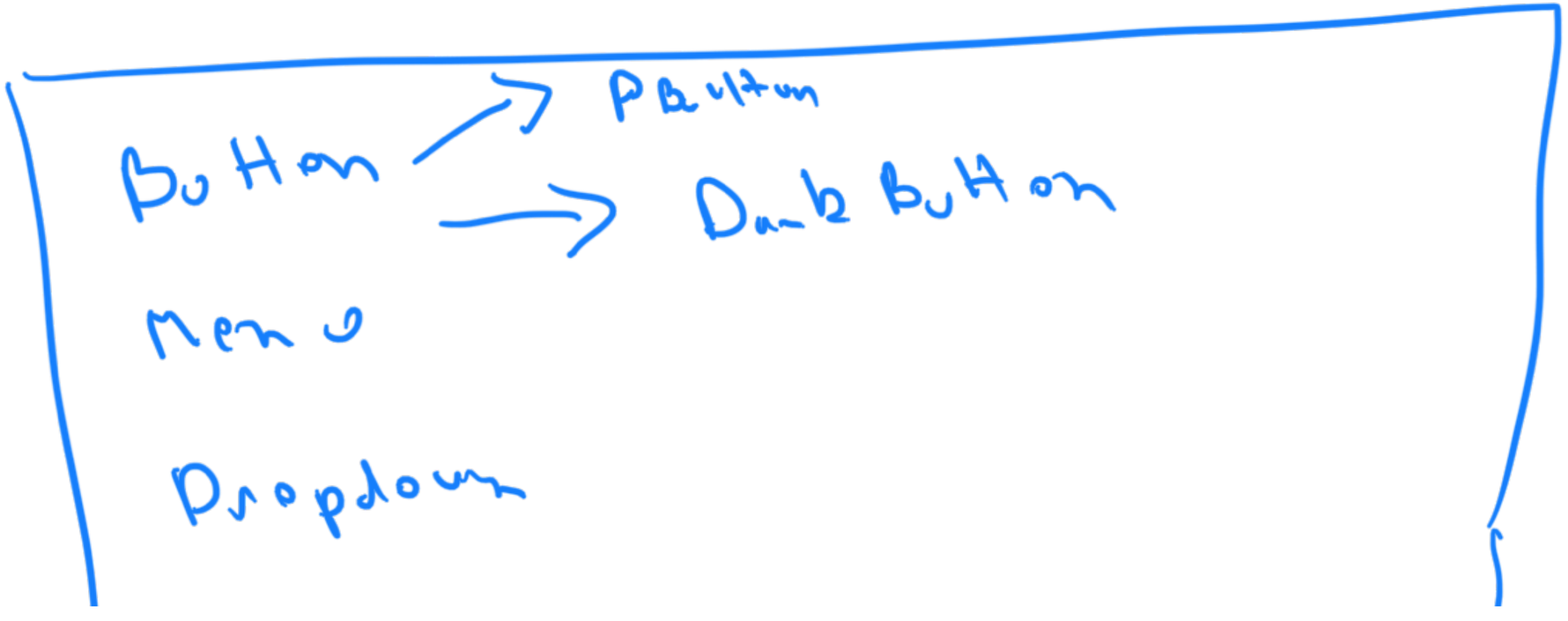
Frontend

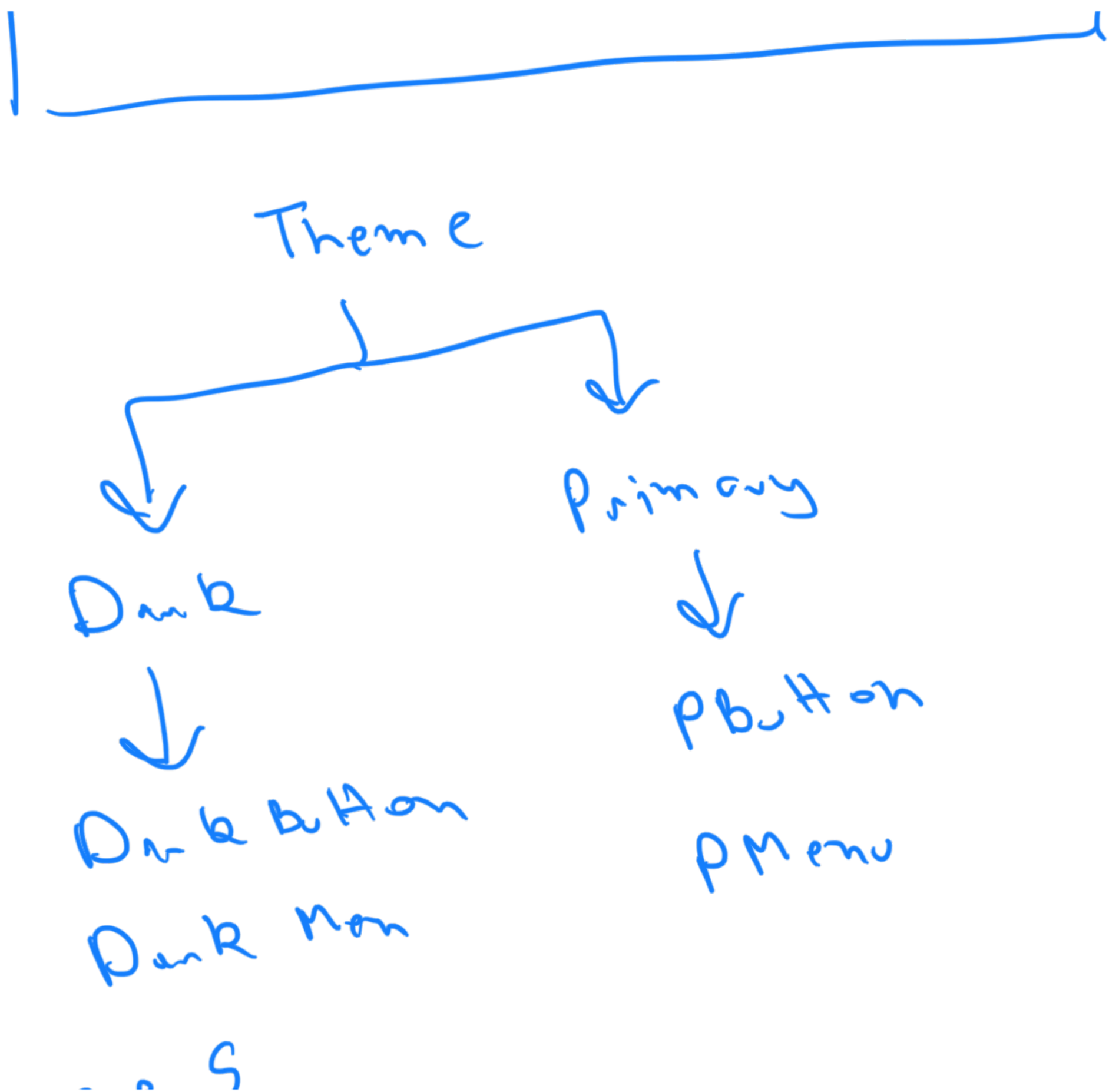
Themes

↳ Dark

↳ Light

Theme




Theme
DmR
DmRButton
DmRMen
S
Primary
PButton
PMenu

The me d

Color primary Color;
Font fontA;
changeColor();
} Attributes

Create Button()
Create Menu(); } FM

Factory Method

- Create Button()
- Create Menu()
- Create

Factory of factories

Theme

Theme

Theme Factory

Details

Theme {

name :

color)
Create Component Factory ( );

FM →

Theme Component Factory {

Create Button ( );

Create Menu ( );

Create Dropdown ( );

}

FM

Theme1 → Primary Theme

Dark theme

```
Theme t = new DarkTheme();

DarkTheme extends Theme {

  Create CF {
      return new DarkComponentF();
  }
```