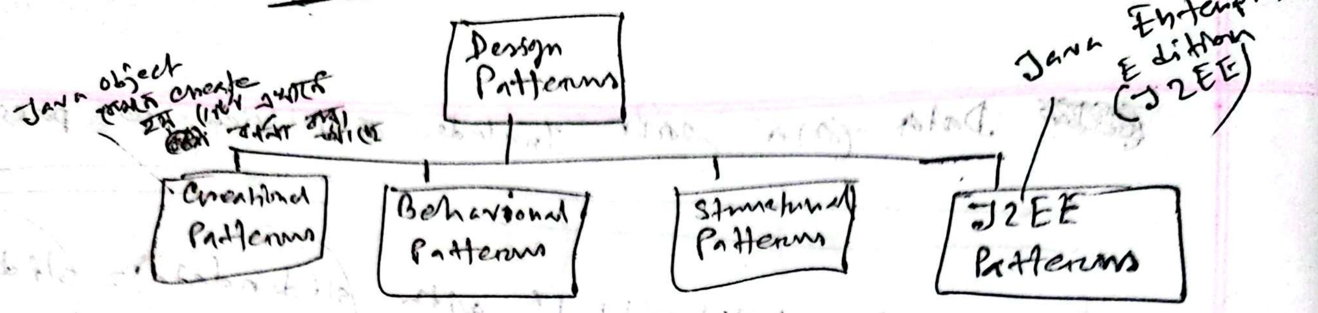
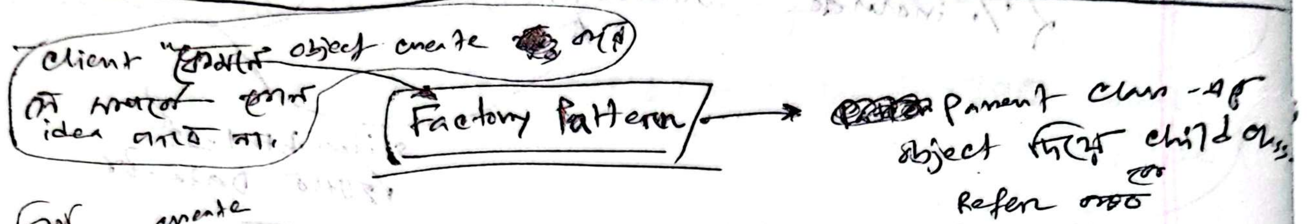
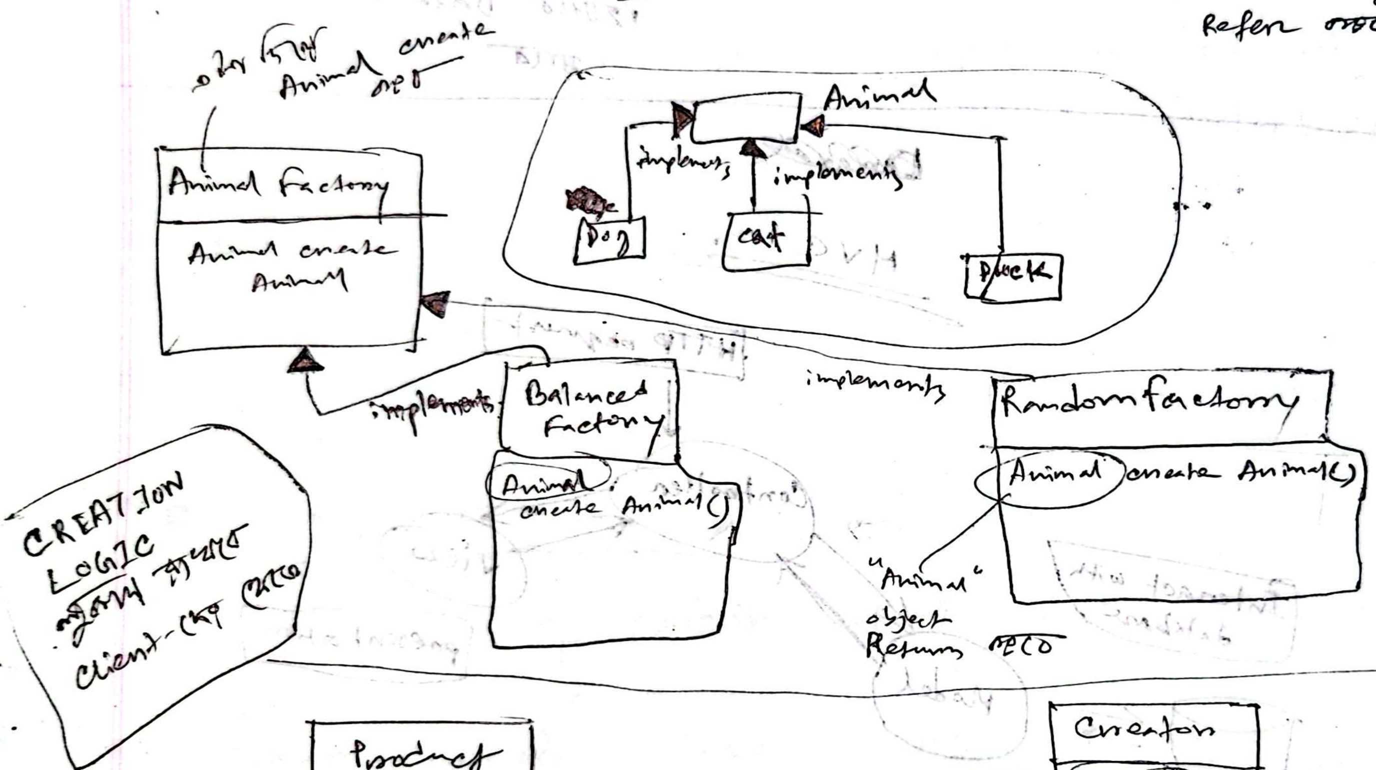
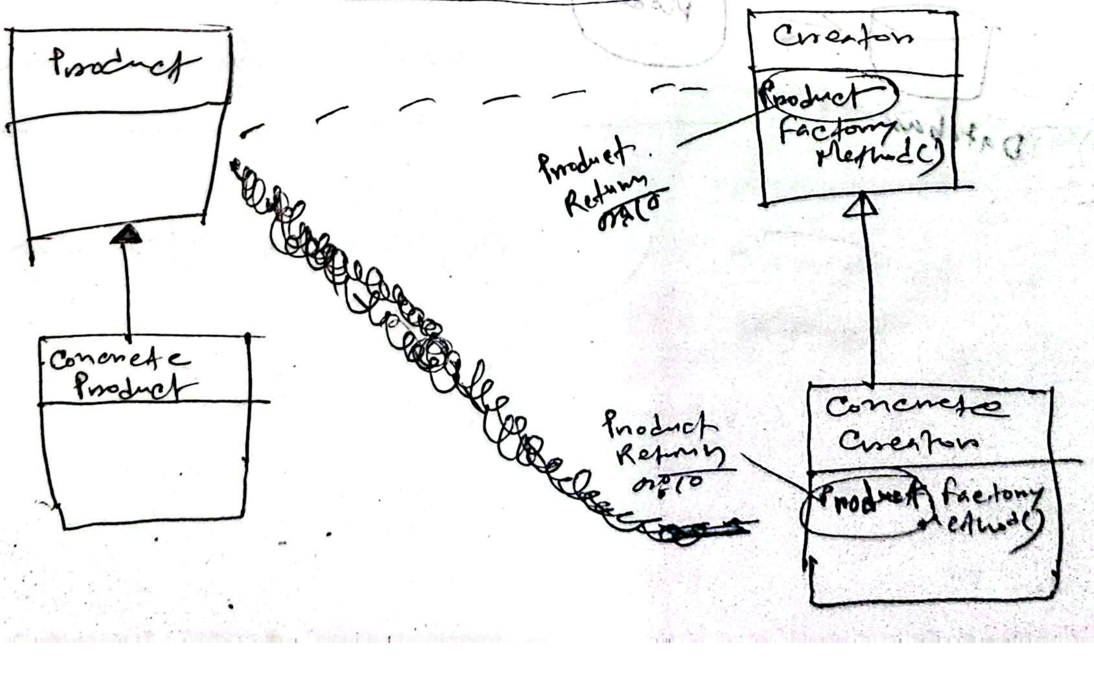
Java Design Patterns


Design Patterns
Java object কিভাবে create হয় সেটা এখানে বর্ণনা করা থাকে।
Creational Patterns
Behavioral Patterns
Structural Patterns
J2EE Patterns
Java Enterprise Edition (J2EE)


Client কিভাবে object create হচ্ছে সে ব্যাপারে কোনো idea পাবে না।
Factory Pattern
Parent class-এর object দিয়ে child class কে Refer করে

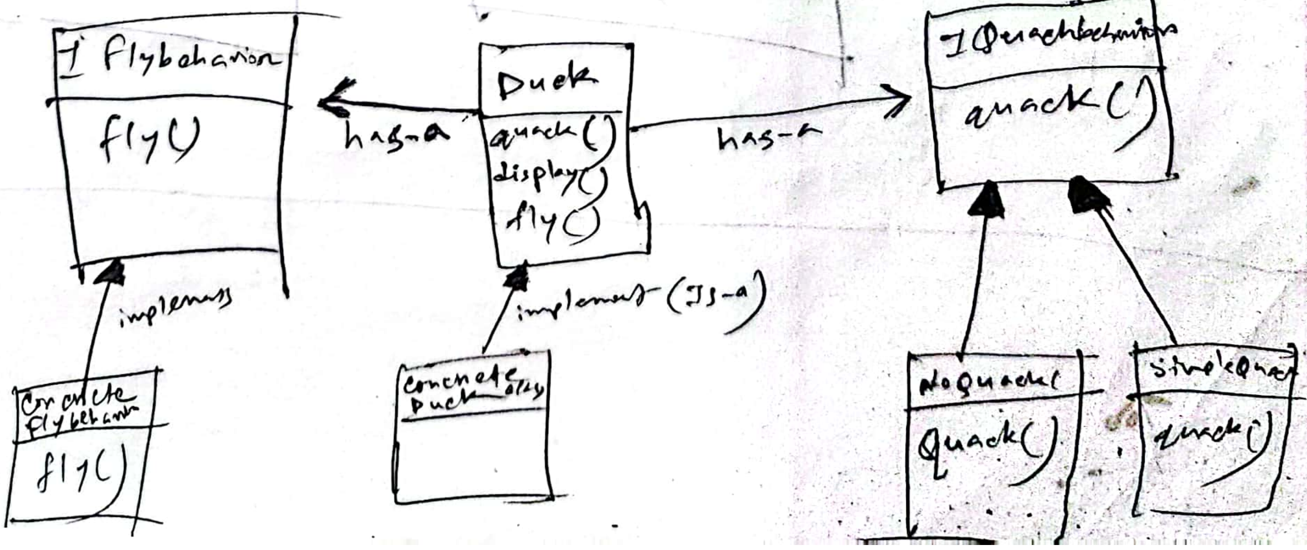




Singleton Pattern

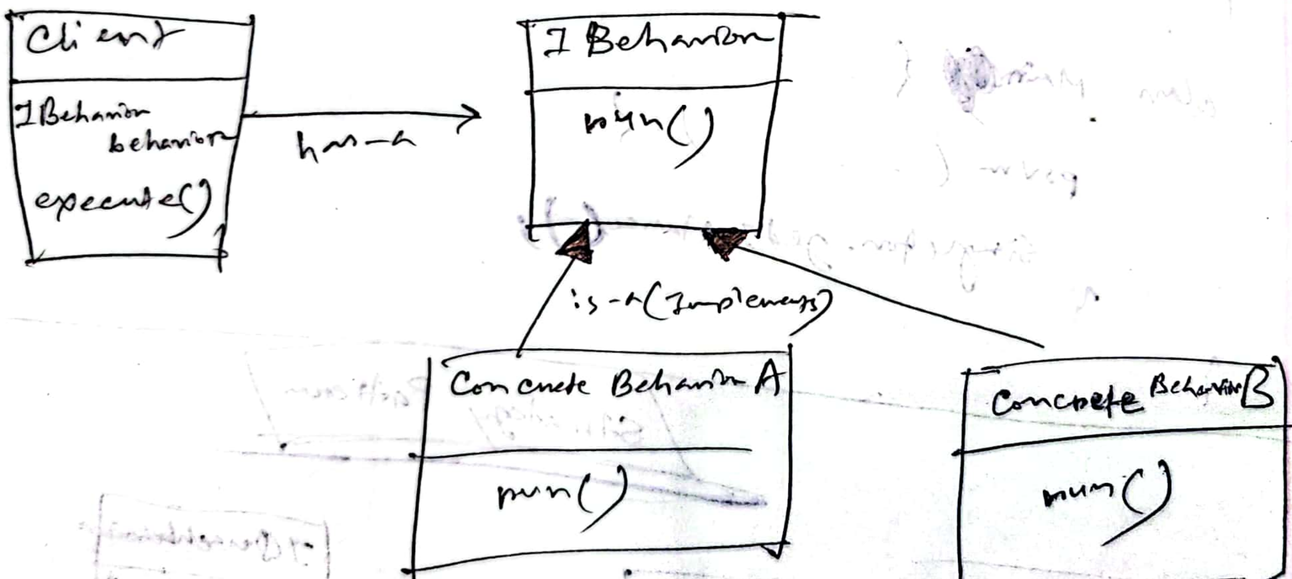
```
class Singleton{
    static private Singleton instance;
    private Singleton()
    {
        .......
    }
    public static Singleton getInstance()
    {
        if(instance==null){
            instance = new Singleton();
        }
        return instance;
    }
}

class Main {
    psvm ( ...
        Singleton.getInstance();
    }
}
```

Strategy Pattern



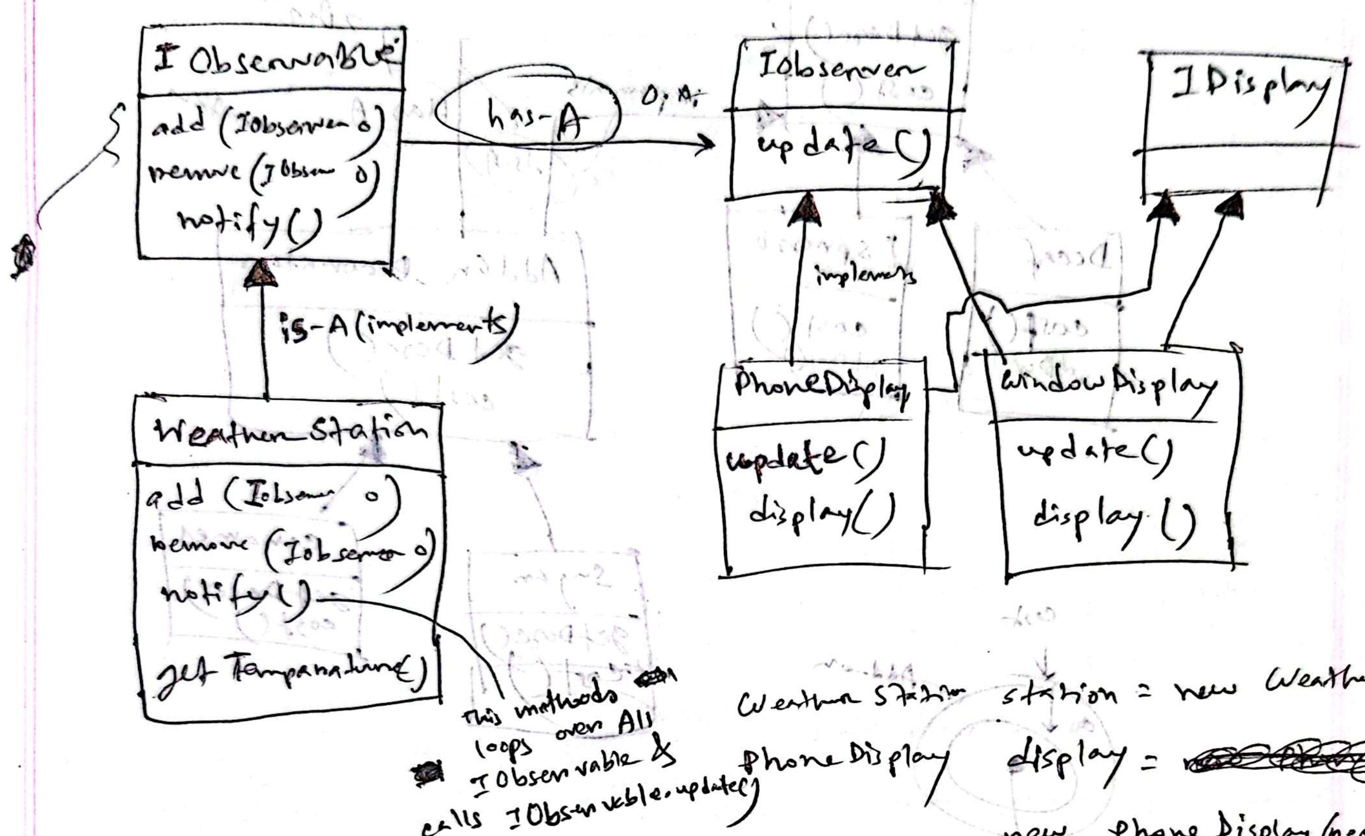
Strategy Pattern

```
abstract class Duck {
        I FlyBehavior fb;
        I QuackBehavior qb;

        public Duck(FlyBehavior fb, QuackBehavior qb)
           {
                this.fb=fb;
                this.qb=qb;
           }

public void fly() {
                this.fb.fly()
      }
}
```



Observer Pattern

Observable - যখনই change হবে তখন সে Notify() করবে যে আমি change হয়েছি



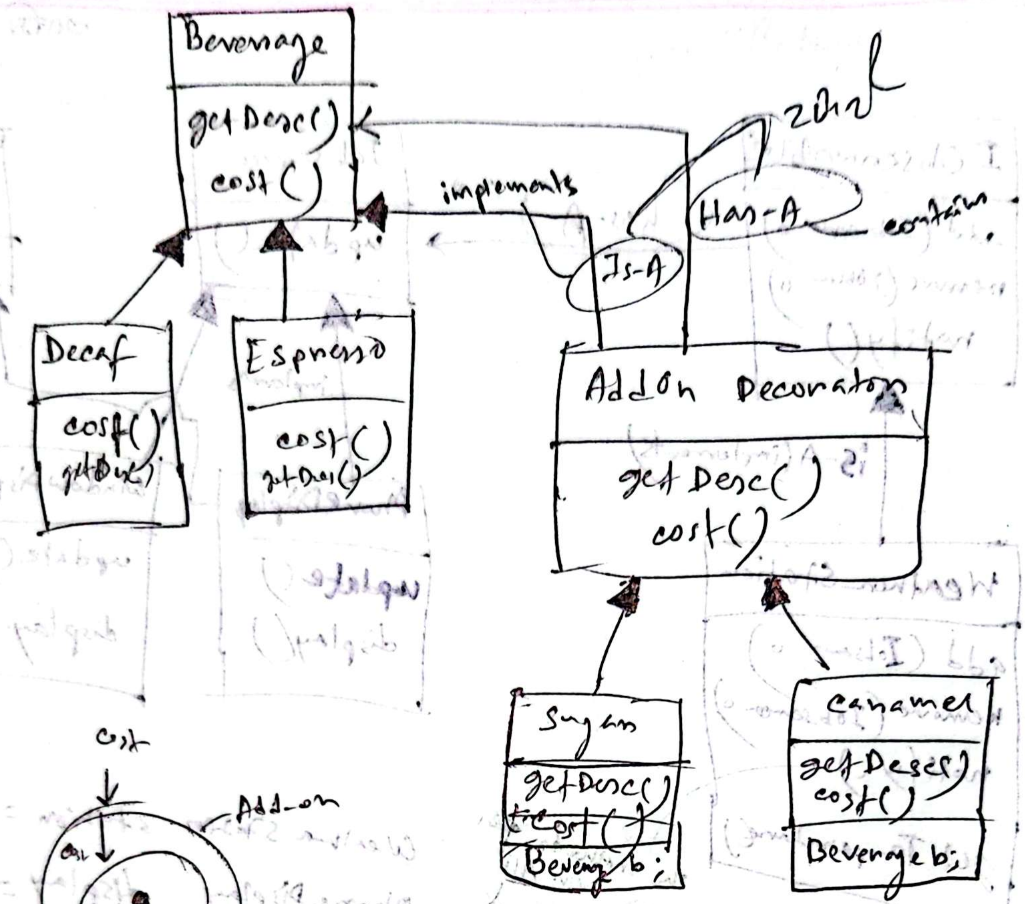
```
WeatherStation station = new WeatherStation()
PhoneDisplay display = ~~[illegible]~~
    new PhoneDisplay(new station)
```

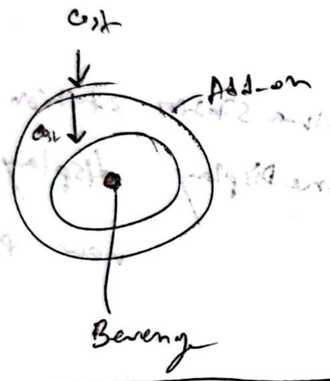
এই station pass করে দিবো যাতে station-এর instance-টা তারা show করতে পারে।

WeatherStation - এর মধ্যে IObserver-এর সব object - রাখবো। যাতে যখনই WeatherStation change হবে তখন সে সব IObserver - কে Notify() করতে পারে।

Design Pattern

Decorator Pattern





```
abstract class Beverage(){
      public abstract int cost();
  }
```

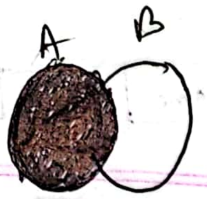
```
class Espresso_coffee extends Beverage{
        public int cost(){
              return 10;
   }    }
```

```
Main class {
    Beverage b =
         new Caramel (new Espresso());
} b.cost();
```

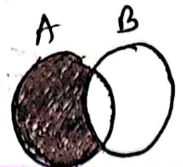
```
abstract class AddOn_Decorator
                     implement Beverage
(Is-A) {
          int cost();
   }
```

```
class Caramel extends Add_on
(Has-A)                      Decorator
   {  Beverage b;
Constructor
   →Caramel (Beverage b)
   {
       this.b = b;
   }
   int cost(){
   return (this.b.cost() + 3
   }
```

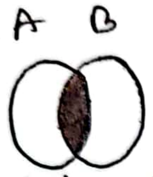
{ coffee
call now the cost



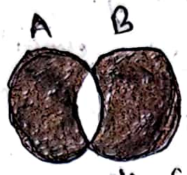
Select * from
A Left Join B
on A.key = B.key



Select * from
A Left Join B
on A.key = B.key
where, $\boxed{\text{B.key IS Null}}$ ***

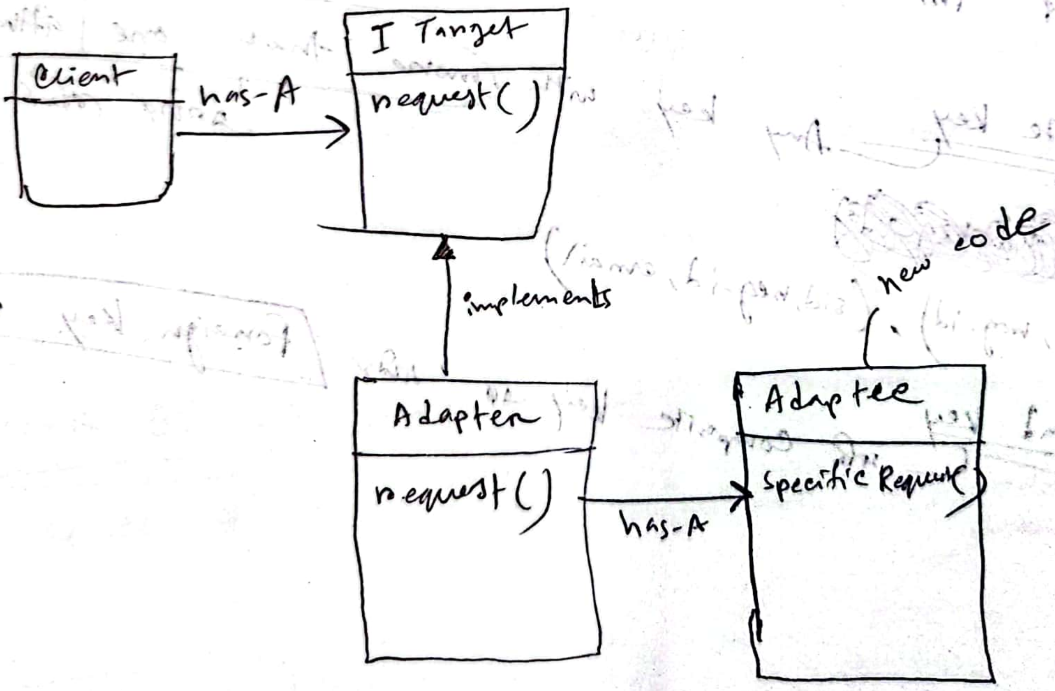


Select * from
A Inner Join B
on A.key = B.key
শুধু Join দিলেও same কাজ করে



Select * from
A Full Outer join B
on A.key = B.key, ***
where A.key IS Null
OR B.key IS Null

Adapter Design Pattern


I Target
request()
Client
has-A
implements
Adapter
request()
has-A
Adaptee
SpecificRequest()
new code