Tasarım Örüntüleri kullanarak "Fiber Fiyat Hesaplama" Projesi

Tayfun Erkorkmaz <tayfunerkorkmaz@gmail.com>

Revision History

Revision 0.9	2019-06-11	TE

- Bu dokümanın hazırlanmasında asciidoc kullanılmıştır.
- Dokümanın hazırlanması için AsciidocFX programından faydalanılmıştır.
- UML diyagramları için plantuml'den faydalanılmıştır.
- Program kodları java ile yazılmıştır.
- Uygulama NetBeans ile hazırlanmıştır. Klasör içerisinde uygulamanın çalışabilmesi için ihtiyaç duyulan kütüphaneler mevcuttur.
- Uygulama ilk açılışında bir mysql bağlantısı kurmaya çalışacaktır. Mysql'e localhost üzerinden bağlanmaya çalışacaktır. Kaynak kodlarından bağlantı ayarlarını değiştirebilirsiniz.
- Uygulama bir ana demo ekranından oluşur. Uygulama ilk çalışmada döviz kurlarını çektiği için internet bağlantısına ihtiyaç duyar.

1. Tasarım Örüntüleri

1.1. Nesne Yaratılmasına (Creational) İlişkin Patternler

Singleton
Factory
Prototype
Object Pool
Builder
Abstract Factory

Singleton

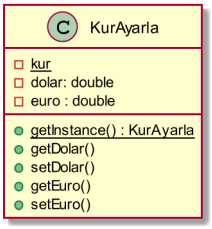
Hazırlayacağımız sınıftan sadece bir örneğinin oluşturulmasını sağlar. Bu sayede nesnenin kopyalanmasını yada yeni bir tane oluşturmasını engeller ve nesneye ihtiyaç duyulduğunda o nesnenin daha önceden oluşturulan örneği çağırır.

Veritabanı bağlantılarında, port bağlantılarında, ya da dosya işlemleri gibi tek bir nesneye ihtiyaç duyduğumuz zamanlarda kullanırız.

Neden Kullanıldı?

Ürün fiyat hesaplamaları için gereken kur verisinin Merkez Bankasından çekilip tutulması için bu tasarım örüntüsü kullanılmıştır. Kur bilgileri ihtiyaç duyulduğunda ya da güncellenmesi gerektiğinde programın başladığında oluşturulan nesnenin örneğinden istenecektir.

Singleton



plantuml code.

```
@startuml
title Singleton

class KurAyarla{
 {static} - kur
 - dolar: double
 - euro : double
 {static} + getInstance() : KurAyarla
 +getDolar()
 +setDolar()
```

```
 +getEuro()
 +setEuro()
}
@enduml
```

KurAyarla.java.

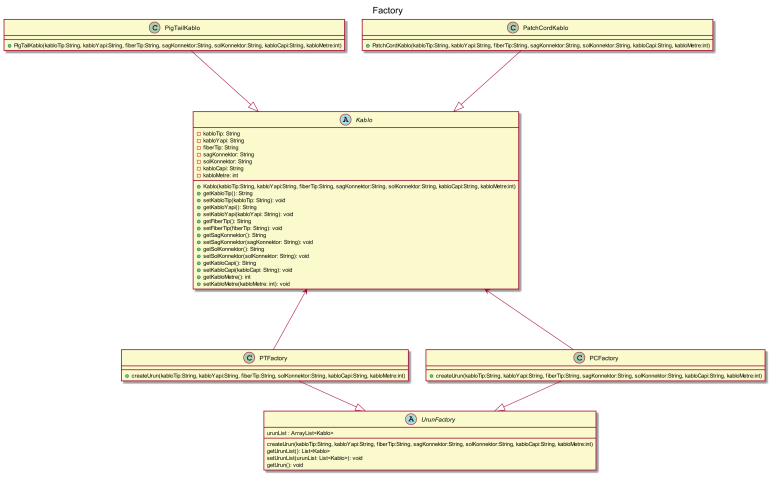
```
public class KurAyarla {

    private static KurAyarla kur;
    private double dolar;
    private double euro;
    private static Object lock_obj = new Object();

    private KurAyarla() {
        System.out.println("Singleton Pattern Çalıştı!");
    }

    public static KurAyarla CreateObject() {

        synchronized(lock_obj){
            if (kur == null) {
                kur = new KurAyarla();
            }
        }
        return kur;
    }

    public double getDolar() {
        return dolar;
    }

    public void setDolar(double dolar) {
        this.dolar = dolar;
    }

    public double getEuro() {
        return euro;
    }

    public void setEuro(double euro) {
        this.euro = euro;
    }

}
```

Factory

Birbirinden yapısal olarak farklı ancak aynı zamanda birçok karakteristik özelliği ortak olan nesnelerin yönetimi, oluşturma kıstaslarının belirlenmesi ve yaratılması için Factory Metodu kullanılır. Factory Method deseninin ana amacı, “genişletilebilirlik” tir.

Neden Kullanıldı?

İki tipteki herhangi bir kablo türü oluşturulurken karakteristik olarak birbirine benzeyen farklı türdeki Factory sınıflarına ulaşılacaktır. Yeni bir kablo türü eklenmek istediğinde yeni bir Factory sınıfı eklenebilecektir.



plantuml code.

```
@startuml
title Factory
abstract class Kablo {

- kabloTip: String
- kabloYapi: String
- fiberTip: String
- sagKonnektor: String
- solKonnektor: String
```

```
- kabloCapi: String
- kabloMetre: int

+ Kablo(kabloTip:String, kabloYapi:String, fiberTip:String,
 sagKonnektor:String, solKonnektor:String, kabloCapi:String,
 kabloMetre:int)
+ getKabloTip(): String
+ setKabloTip(kabloTip: String): void
+ getKabloYapi(): String
+ setKabloYapi(kabloYapi: String): void
+ getFiberTip(): String
+ setFiberTip(fiberTip: String): void
+ getSagKonnektor(): String
+ setSagKonnektor(sagKonnektor: String): void
+ getSolKonnektor(): String
+ setSolKonnektor(solKonnektor: String): void
+ getKabloCapi(): String
+ setKabloCapi(kabloCapi: String): void
+ getKabloMetre(): int
+ setKabloMetre(kabloMetre: int): void
}

abstract class UrunFactory {
urunList : ArrayList<Kablo>
createUrun(kabloTip:String, kabloYapi:String, fiberTip:String,
 sagKonnektor:String, solKonnektor:String, kabloCapi:String,
 kabloMetre:int)
getUrunList(): List<Kablo>
setUrunList(urunList: List<Kablo>): void
getUrun(): void
}

class PigTailKablo{
+ PigTailKablo(kabloTip:String, kabloYapi:String, fiberTip:String,
 sagKonnektor:String, solKonnektor:String, kabloCapi:String,
 kabloMetre:int)
}

class PatchCordKablo{
+ PatchCordKablo(kabloTip:String, kabloYapi:String, fiberTip:String,
 sagKonnektor:String, solKonnektor:String, kabloCapi:String,
 kabloMetre:int)
}

class PTFactory{
```

```
+ createUrun(kabloTip:String, kabloYapi:String, fiberTip:String,
 solKonnektor:String, kabloCapi:String, kabloMetre:int)
}

class PCFactory{
+ createUrun(kabloTip:String, kabloYapi:String, fiberTip:String,
 sagKonnektor:String, solKonnektor:String, kabloCapi:String,
 kabloMetre:int)
}

UrunFactory <|-up- PCFactory
UrunFactory <|-up- PTFactory
PigTailKablo ---|> Kablo
PatchCordKablo ---|> Kablo
Kablo <--- PCFactory
Kablo <--- PTFactory

@enduml
```

UrunFactory.java.

```
public abstract class UrunFactory {

 public List<Kablo> urunList=new ArrayList<Kablo>();

 public abstract void createUrun(String kabloTip, String kabloYapi,
 String fiberTip, String sagKonnektor, String solKonnektor, String
 kabloCapi, int kabloMetre);

 public List<Kablo> getUrunList() {
  return urunList;
 }

 public void setUrunList(List<Kablo> arabaListesi) {
  this.urunList = urunList;
 }

}
```

Kablo.java.

```
public abstract class Kablo {

    private String kabloTip;
    private String kabloYapi;
```

```
    private String fiberTip;
    private String sagKonnektor;
    private String solKonnektor;
    private String kabloCapi;
    private int kabloMetre;

    public Kablo(String kabloTip, String kabloYapi, String fiberTip,
String sagKonnektor, String solKonnektor, String kabloCapi, int
kabloMetre) {
        this.kabloTip = kabloTip;
        this.kabloYapi = kabloYapi;
        this.fiberTip = fiberTip;
        this.sagKonnektor = sagKonnektor;
        this.solKonnektor = solKonnektor;
        this.kabloCapi = kabloCapi;
        this.kabloMetre = kabloMetre;
    }

    public String getKabloTip() {
        return kabloTip;
    }

    public String getKabloYapi() {
        return kabloYapi;
    }

    public String getFiberTip() {
        return fiberTip;
    }

    public String getSagKonnektor() {
        return sagKonnektor;
    }

    public String getSolKonnektor() {
        return solKonnektor;
    }

    public String getKabloCapi() {
        return kabloCapi;
    }

    public int getKabloMetre() {
        return kabloMetre;
    }
```

```
    public void getUrun(){
    System.out.println("Kablo Tipi: " +  kabloTip);
    System.out.println("Kablo Yapısı: " + kabloYapi );
    System.out.println("Fiber Tipi: " + fiberTip);
    System.out.println("Sağ Konnektor: " + sagKonnektor);
    System.out.println("Sol Konnektor: " + solKonnektor);
    System.out.println("Kablo Çapı: " + kabloCapi);
    System.out.println("Kablo Uzunluğu: " + kabloMetre );
    }


}
```

PigTailKablo.java.

```
public class PigTailKablo extends Kablo {

    public PigTailKablo(String kabloTip, String kabloYapi, String
 fiberTip, String sagKonnektor, String solKonnektor, String
 kabloCapi, int kabloMetre) {
        super(kabloTip, kabloYapi, fiberTip, sagKonnektor, solKonnektor,
 kabloCapi, kabloMetre);
    }

}
```

PatchCordKablo.java.

```
public class PatchCordKablo extends Kablo {

    public PatchCordKablo(String kabloTip, String kabloYapi,
 String fiberTip, String sagKonnektor, String solKonnektor, String
 kabloCapi, int kabloMetre) {
        super(kabloTip, kabloYapi, fiberTip, sagKonnektor, solKonnektor,
 kabloCapi, kabloMetre);
    }

}
```

PTFactory.java.

```
public class PTFactory extends UrunFactory{

    @Override
```

```
    public void createUrun(String kabloTip, String kabloYapi,
 String fiberTip, String sagKonnektor, String solKonnektor, String
 kabloCapi, int kabloMetre) {
        getUrunList().add(new PigTailKablo(kabloTip, kabloYapi,
 fiberTip, "X", solKonnektor, kabloCapi, kabloMetre));

    }

}
```

PCFactory.java.

```
public class PCFactory extends UrunFactory {

    @Override
    public void createUrun(String kabloTip, String kabloYapi,
 String fiberTip, String sagKonnektor, String solKonnektor, String
 kabloCapi, int kabloMetre) {
        getUrunList().add(new PatchCordKablo(kabloTip, kabloYapi,
 fiberTip, sagKonnektor, solKonnektor, kabloCapi, kabloMetre));
    }

}
```

Prototype

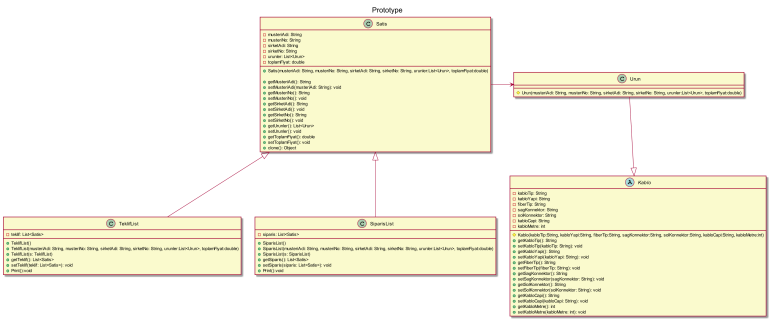
Nesnemizi birden fazla oluşturmamız gerektiğinde normalde “new” olarak oluşturmak yerine daha önce oluşturduğumuz nesnemizin klonunu oluşturmamızı sağlayan bir design pattern’dir.

Shallow Copy: Property referanslarının birbirine eşlenmesi anlamına gelirki String, int, float vb. primitiv structure’larda bir ortaklık yok iken Collection, Sub Class vb. kısımların kopyalana iki sınıf tarafından paylaşılması anlamına gelir.

Deep Copy: Nesne içerisinde yer alan tüm referanslar için yeni bellek alanlarının alındığı ve en içteki primitiflere kadar kopyalamanın yapıldığı kopyalama türüdür.

Neden Kullanıldı?

Oluşturulan bir teklifin siparişe çevrilmesi için teklif nesnesinin klonlanarak sipariş nesnesine dönüştürülmesi amaçlanmıştır.



plantuml code.

```
@startuml
title Prototype
abstract class Kablo {

- kabloTip: String
- kabloYapi: String
- fiberTip: String
- sagKonnektor: String
- solKonnektor: String
- kabloCapi: String
- kabloMetre: int

# Kablo(kabloTip:String, kabloYapi:String, fiberTip:String,
 sagKonnektor:String, solKonnektor:String, kabloCapi:String,
 kabloMetre:int)
+ getKabloTip(): String
+ setKabloTip(kabloTip: String): void
+ getKabloYapi(): String
+ setKabloYapi(kabloYapi: String): void
+ getFiberTip(): String
+ setFiberTip(fiberTip: String): void
+ getSagKonnektor(): String
+ setSagKonnektor(sagKonnektor: String): void
+ getSolKonnektor(): String
+ setSolKonnektor(solKonnektor: String): void
```

```
+ getKabloCapi(): String
+ setKabloCapi(kabloCapi: String): void
+ getKabloMetre(): int
+ setKabloMetre(kabloMetre: int): void
}

class Satis{
- musteriAdi: String
- musteriNo: String
- sirketAdi: String
- sirketNo: String
- urunler: List<Urun>
- toplamFiyat: double

+ Satis(musteriAdi: String, musteriNo: String, sirketAdi: String,
 sirketNo: String, urunler:List<Urun>, toplamFiyat:double)

+ getMusteriAdi(): String
+ setMusteriAdi(musteriAdi: String): void
+ getMusteriNo(): String
+ setMusteriNo(): void
+ getSirketAdi(): String
+ setSirketAdi(): void
+ getSirketNo(): String
+ setSirketNo(): void
+ getUrunler(): List<Urun>
+ setUrunler(): void
+ getToplamFiyat(): double
+ setToplamFiyat(): void
+ clone(): Object
}

class TeklifList{
- teklif: List<Satis>
+ TeklifList()
+ TeklifList(musteriAdi: String, musteriNo: String, sirketAdi: String,
 sirketNo: String, urunler:List<Urun>, toplamFiyat:double)
+ TeklifList(s: TeklifList)
+ getTeklif(): List<Satis>
+ setTeklif(teklif: List<Satis>): void
+ Print():void
}

class SiparisList{
- siparis: List<Satis>
+ SiparisList()
```

```
+ SiparisList(musteriAdi: String, musteriNo: String, sirketAdi: String,
 sirketNo: String, urunler:List<Urun>, toplamFiyat:double)
+ SiparisList(s: SiparisList)
+ getSiparis(): List<Satis>
+ setSiparis(siparis: List<Satis>): void
+ Print():void
}

class Urun {
#Urun(musteriAdi: String, musteriNo: String, sirketAdi: String,
 sirketNo: String, urunler:List<Urun>, toplamFiyat:double)
}

Urun --|> Kablo
Satis <|-- SiparisList
Satis <|-- TeklifList
Satis -right-> Urun

@enduml
```

Kablo.java.

```
public abstract class Kablo {

    private String kabloTip;
    private String kabloYapi;
    private String fiberTip;
    private String sagKonnektor;
    private String solKonnektor;
    private String kabloCapi;
    private int kabloMetre;

    protected Kablo(String kabloTip, String kabloYapi, String fiberTip,
 String sagKonnektor, String solKonnektor, String kabloCapi, int
 kabloMetre) {
        this.kabloTip = kabloTip;
        this.kabloYapi = kabloYapi;
        this.fiberTip = fiberTip;
        this.sagKonnektor = sagKonnektor;
        this.solKonnektor = solKonnektor;
        this.kabloCapi = kabloCapi;
        this.kabloMetre = kabloMetre;
    }

    public String getKabloTip() {
```

```
        return kabloTip;
    }

    public String getKabloYapi() {
        return kabloYapi;
    }

    public String getFiberTip() {
        return fiberTip;
    }

    public String getSagKonnektor() {
        return sagKonnektor;
    }

    public String getSolKonnektor() {
        return solKonnektor;
    }

    public String getKabloCapi() {
        return kabloCapi;
    }

    public int getKabloMetre() {
        return kabloMetre;
    }

    public void getUrun(){
    System.out.println("Kablo Tipi: " +  kabloTip);
    System.out.println("Kablo Yapısı: " + kabloYapi );
    System.out.println("Fiber Tipi: " + fiberTip);
    System.out.println("Sağ Konnektor: " + sagKonnektor);
    System.out.println("Sol Konnektor: " + solKonnektor);
    System.out.println("Kablo Çapı: " + kabloCapi);
    System.out.println("Kablo Uzunluğu: " + kabloMetre );
    }


}
```

Urun.java.

```
public class Urun extends Kablo {
```

```
    protected Urun(String kabloTip, String kabloYapi, String fiberTip,
 String sagKonnektor, String solKonnektor, String kabloCapi, int
 kabloMetre) {
        super(kabloTip, kabloYapi, fiberTip, sagKonnektor, solKonnektor,
 kabloCapi, kabloMetre);
    }
}
```

Satis.java.

```
public class Satis implements Cloneable {

    private String musteriAdi;
    private String musteriNo;
    private String sirketAdi;
    private String sirketNo;
    private List<Urun> urunler = new ArrayList();
    private double toplamFiyat;

    public Satis(String musteriAdi, String musteriNo, String sirketAdi,
 String sirketNo, List<Urun> urunler, double toplamFiyat) {
        this.musteriAdi = musteriAdi;
        this.musteriNo = musteriNo;
        this.sirketAdi = sirketAdi;
        this.sirketNo = sirketNo;
        this.urunler = urunler;
        this.toplamFiyat = toplamFiyat;
    }

    public Satis() {

    }

    public String getMusteriAdi() {
        return musteriAdi;
    }

    public void setMusteriAdi(String musteriAdi) {
        this.musteriAdi = musteriAdi;
    }

    public String getMusteriNo() {
        return musteriNo;
    }
```

```
    public void setMusteriNo(String musteriNo) {
        this.musteriNo = musteriNo;
    }

    public String getSirketAdi() {
        return sirketAdi;
    }

    public void setSirketAdi(String sirketAdi) {
        this.sirketAdi = sirketAdi;
    }

    public String getSirketNo() {
        return sirketNo;
    }

    public void setSirketNo(String sirketNo) {
        this.sirketNo = sirketNo;
    }

    public List<Urun> getUrunler() {
        return urunler;
    }

    public void setUrunler(List<Urun> urunler) {
        this.urunler = urunler;
    }

    public double getToplamFiyat() {
        return toplamFiyat;
    }

    public void setToplamFiyat(double toplamFiyat) {
        this.toplamFiyat = toplamFiyat;
    }

    public void urunEkle(Urun u) {
        urunler.add(u);

    }

    @Override
    public Object clone() throws CloneNotSupportedException {
        Satis satis = null;
        try {
            satis = (Satis) super.clone();
```

```
        } catch (CloneNotSupportedException e) {
            System.out.println("Problem when cloning the object: " +
 e.getMessage());
        }
        return satis;
    }

}
```

SiparisList.java.

```
public class SiparisList extends Satis {

    List<Satis> siparis;

    public SiparisList() {
        siparis = new ArrayList();
    }

    public SiparisList(String musteriAdi, String musteriNo, String
 sirketAdi, String sirketNo, List<Urun> urunler, double toplamFiyat) {
        super(musteriAdi, musteriNo, sirketAdi, sirketNo, urunler,
 toplamFiyat);
    }

    public SiparisList(SiparisList s) {
        this(s.getMusteriAdi(), s.getMusteriNo(), s.getSirketAdi(),
 s.getSirketNo(), s.getUrunler(), s.getToplamFiyat());
        this.siparis = new ArrayList<>(s.siparis);;
    }

    public List<Satis> getSiparis() {
        return siparis;
    }

    public void setSiparis(List<Satis> siparis) {
        this.siparis = siparis;
    }

    public void Print() {
        for (Satis s : siparis) {
            for (Urun u : s.getUrunler()) {
                System.out.println(s.getMusteriAdi() + " " +
 s.getMusteriNo() + " " + s.getSirketAdi() + "UrunFiberTipi: " +
 u.getFiberTip());
```

```
            }
        }
    }



}
```

TeklifList.java.

```
public class TeklifList extends Satis {

    List<Satis> teklif;

    public TeklifList() {
        teklif = new ArrayList();
    }

     public TeklifList(String musteriAdi, String musteriNo, String
 sirketAdi, String sirketNo, List<Urun> urunler, double toplamFiyat) {
        super(musteriAdi, musteriNo, sirketAdi, sirketNo, urunler,
 toplamFiyat);
    }

    public TeklifList(TeklifList s) {
        this(s.getMusteriAdi(), s.getMusteriNo(), s.getSirketAdi(),
 s.getSirketNo(), s.getUrunler(), s.getToplamFiyat());
        this.teklif = new ArrayList<>(s.teklif);
    }

    public List<Satis> getTeklif() {
        return teklif;
    }

    public void setTeklif(List<Satis> teklif) {
        this.teklif = teklif;
    }

    public void Print() {
        for (Satis s : teklif) {
            for (Urun u : s.getUrunler()) {
                System.out.println(s.getMusteriAdi() + " " +
 s.getMusteriNo() + " " + s.getSirketAdi() + "UrunFiberTipi: " +
 u.getFiberTip());
            }
```

```
        }
    }


}
```

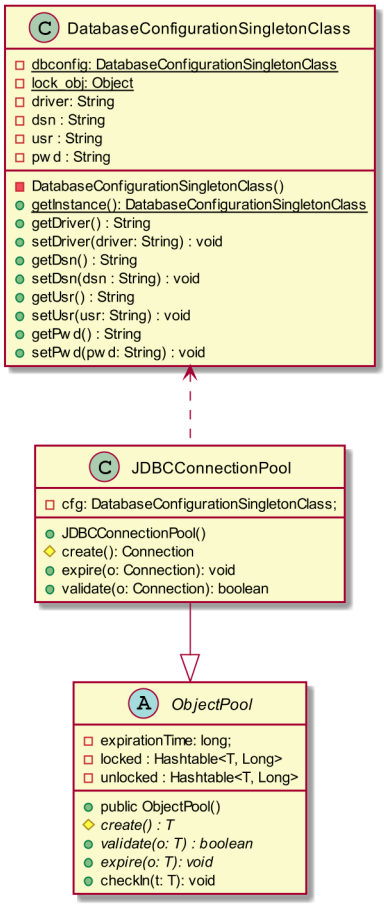
Object Pool

İstenilen nesnelerin sürekli olarak üretilmesi yerine, başlangıçta bir havuzu oluşturulur ve bu havuz nesneler ile doldurulur. İhtiyaç halinde bu nesnelerden biri kullanılmak üzere istenir.

Neden Kullanıldı?

Database bağlantısının yeniden oluşturulması yerine, oluşturulmuş bağlantı nesnesinin havuzdan çekilerek yeniden kullanılması sağlanmıştır.

Object Pool



plantuml code.

```
@startuml
title Object Pool
class DatabaseConfigurationSingletonClass{
{static} - dbconfig: DatabaseConfigurationSingletonClass
{static} - lock_obj: Object
- driver: String
- dsn : String
- usr : String
- pwd : String
- DatabaseConfigurationSingletonClass()
{static} + getInstance(): DatabaseConfigurationSingletonClass
+ getDriver() : String
+ setDriver(driver: String) : void
```

```
+ getDsn() : String
+ setDsn(dsn : String) : void
+ getUsr() : String
+ setUsr(usr: String) : void
+ getPwd() : String
+ setPwd(pwd: String) : void
}

abstract class ObjectPool{
- expirationTime: long;
- locked : Hashtable<T, Long>
- unlocked : Hashtable<T, Long>
+ public ObjectPool()
# {abstract} create() : T
+ {abstract} validate(o: T) : boolean
+ {abstract} expire(o: T): void
+ checkIn(t: T): void
}

class JDBCConnectionPool{
- cfg: DatabaseConfigurationSingletonClass;
+ JDBCConnectionPool()
# create(): Connection
+ expire(o: Connection): void
+ validate(o: Connection): boolean
}

JDBCConnectionPool --|> ObjectPool
JDBCConnectionPool .up.> DatabaseConfigurationSingletonClass
@enduml
```

DatabaseConfigurationSingletonClass.java.

```
public class DatabaseConfigurationSingletonClass {

    private static DatabaseConfigurationSingletonClass dbconfig = null;
    private static Object lock_obj = new Object();

    String driver = "com.mysql.jdbc.Driver";
    private String dsn = "jdbc:mysql://localhost:3306/
fiberfiyathesaplama?createDatabaseIfNotExist=true";
    private String usr = "root";
    private String pwd = "root";

    private DatabaseConfigurationSingletonClass() {
```

```
        System.out.println("DBConfig Singleton Pattern Çalıştı!");
    }

    public static DatabaseConfigurationSingletonClass getInstance() {

        synchronized(lock_obj){
            if ( dbconfig == null) {

                 dbconfig= new DatabaseConfigurationSingletonClass();
            }
        }
        return  dbconfig;
    }

    public String getDriver() {
        return driver;
    }

    public void setDriver(String driver) {
        this.driver = driver;
    }

    public String getDsn() {
        return dsn;
    }

    public void setDsn(String dsn) {
        this.dsn = dsn;
    }

    public String getUsr() {
        return usr;
    }

    public void setUsr(String usr) {
        this.usr = usr;
    }

    public String getPwd() {
        return pwd;
    }

    public void setPwd(String pwd) {
        this.pwd = pwd;
    }
```

```
}
```

ObjectPool.java.

```
public abstract class ObjectPool<T> {

    private long expirationTime;
    private Hashtable<T, Long> locked, unlocked;

    public ObjectPool() {
        expirationTime = 30000; // 30 saniye
        locked = new Hashtable<T, Long>();
        unlocked = new Hashtable<T, Long>();
    }

    protected abstract T create();

    public abstract boolean validate(T o);

    public abstract void expire(T o);

    public synchronized T checkOut() {
        long now = System.currentTimeMillis();
        T t;
        if (unlocked.size() > 0) {
            Enumeration<T> e = unlocked.keys();
            while (e.hasMoreElements()) {
                t = e.nextElement();
                if ((now - unlocked.get(t)) > expirationTime) {
                    unlocked.remove(t);
                    expire(t);
                    t = null;
                } else {
                    if (validate(t)) {
                        unlocked.remove(t);
                        locked.put(t, now);
                        return (t);
                    } else {
                        unlocked.remove(t);
                        expire(t);
                        t = null;
                    }
                }
            }
        }
```

```
// obje yoksa yeni bir tane üret
        t = create();
        locked.put(t, now);
        return (t);
    }

    public synchronized void checkIn(T t) {
        locked.remove(t);
        unlocked.put(t, System.currentTimeMillis());
    }
}
```

JDBCConnectionPool.java.

```
public class JDBCConnectionPool extends ObjectPool<Connection> {

    private DatabaseConfigurationSingletonClass cfg;

    public JDBCConnectionPool() {
        super();
        cfg = DatabaseConfigurationSingletonClass.getInstance();
        try {
            Class.forName(cfg.getDriver()).newInstance();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    @Override
    protected Connection create() {
        try {
            return (Connection)
 (DriverManager.getConnection(cfg.getDsn(), cfg.getUsr(),
 cfg.getPwd()));
        } catch (SQLException e) {
            e.printStackTrace();
            return (null);
        }
    }

    @Override
    public void expire(Connection o) {
        try {
            ((Connection) o).close();
        } catch (SQLException e) {
```

```
            e.printStackTrace();
        }
    }

    @Override
    public boolean validate(Connection o) {
        try {
            return (!((Connection) o).isClosed());
        } catch (SQLException e) {
            e.printStackTrace();
            return (false);
        }
    }
}
```

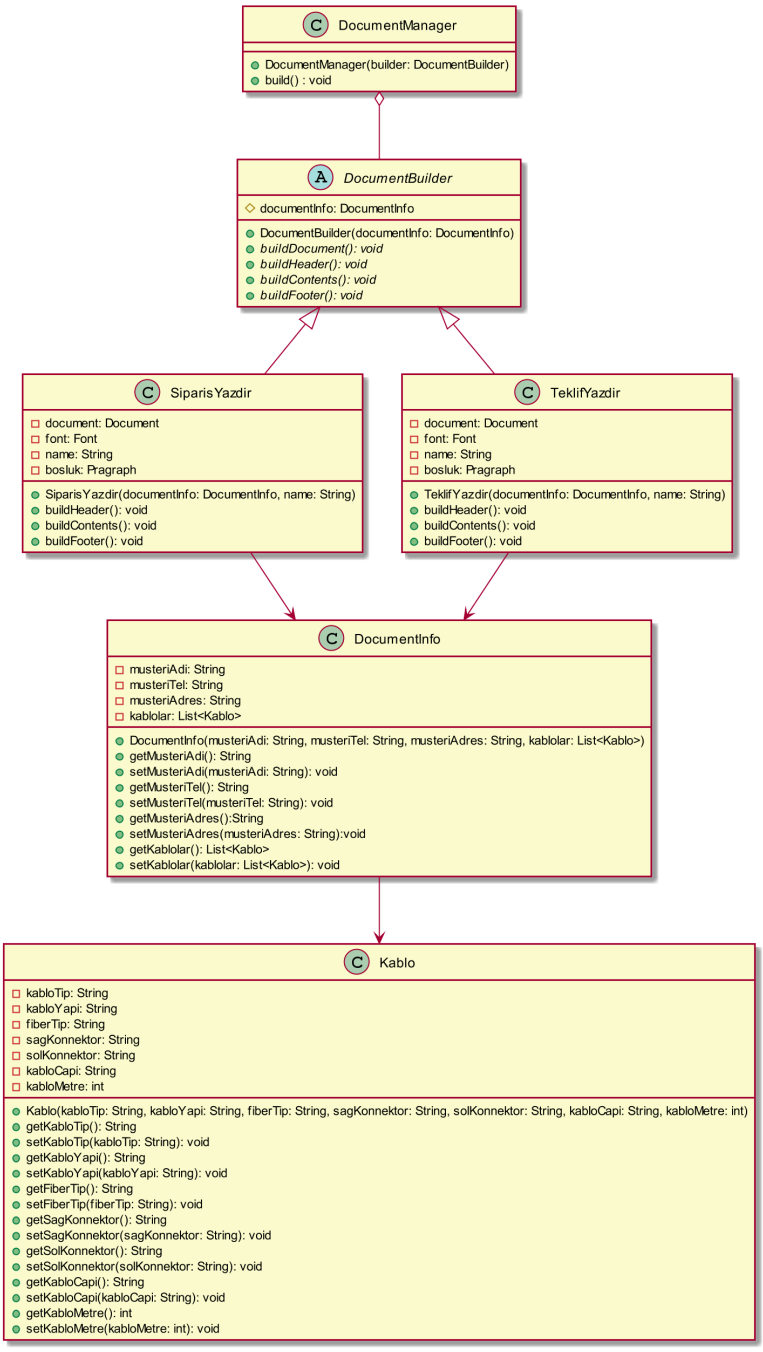
Builder

Bazı nesneler birden fazla nesnenin birleşmesinden oluşabilir. Zamanla bu ana nesneyi oluşturan nesnelerin yapısı değişebilir, bu nesnelerin oluşturulması karışık bir hal alabilir veya bu nesnelere başka nesneler de eklenebilir. Builder tasarım deseni bu gibi durumlarda "genişletilebilirliği" sağlamak ve kod karmaşıklığını engellemek amacıyla kullanılır.

Neden Kullanıldı?

Oluşturulmuş bir Siparişi veya Teklifi PDF formatında bir döküman oluşturmak için tercih edilmiştir. Dökümana için ihtiyaç duyulan çok sayıda paramateyi alarak tercih edilen formata dönüştürür. Döküman içeriklerinin düzenlenmesi bu örüntü sayesinde kolaylaşır. Yeni döküman formatların eklenmeside aynı şekilde kolaylaşır.

Builder



plantuml code.

```
@startuml

title Builder

class DocumentManager{
+ DocumentManager(builder: DocumentBuilder)
+ build() : void
}

abstract class DocumentBuilder {
# documentInfo: DocumentInfo
+ DocumentBuilder(documentInfo: DocumentInfo)
+ {abstract} buildDocument(): void
+ {abstract} buildHeader(): void
+ {abstract} buildContents(): void
+ {abstract} buildFooter(): void
}

class DocumentInfo{
- musteriAdi: String
- musteriTel: String
- musteriAdres: String
- kablolar: List<Kablo>
+ DocumentInfo(musteriAdi: String, musteriTel: String, musteriAdres:
 String, kablolar: List<Kablo>)
+ getMusteriAdi(): String
+ setMusteriAdi(musteriAdi: String): void
+ getMusteriTel(): String
+ setMusteriTel(musteriTel: String): void
+ getMusteriAdres():String
+ setMusteriAdres(musteriAdres: String):void
+ getKablolar(): List<Kablo>
+ setKablolar(kablolar: List<Kablo>): void
}

class TeklifYazdir{
- document: Document
- font: Font
- name: String
- bosluk: Pragraph
+ TeklifYazdir(documentInfo: DocumentInfo, name: String)
+ buildHeader(): void
+ buildContents(): void
+ buildFooter(): void
}
```

```
class Kablo{
- kabloTip: String
- kabloYapi: String
- fiberTip: String
- sagKonnektor: String
- solKonnektor: String
- kabloCapi: String
- kabloMetre: int
+ Kablo(kabloTip: String, kabloYapi: String, fiberTip: String,
 sagKonnektor: String, solKonnektor: String, kabloCapi: String,
 kabloMetre: int)
+ getKabloTip(): String
+ setKabloTip(kabloTip: String): void
+ getKabloYapi(): String
+ setKabloYapi(kabloYapi: String): void
+ getFiberTip(): String
+ setFiberTip(fiberTip: String): void
+ getSagKonnektor(): String
+ setSagKonnektor(sagKonnektor: String): void
+ getSolKonnektor(): String
+ setSolKonnektor(solKonnektor: String): void
+ getKabloCapi(): String
+ setKabloCapi(kabloCapi: String): void
+ getKabloMetre(): int
+ setKabloMetre(kabloMetre: int): void
}

class SiparisYazdir{
- document: Document
- font: Font
- name: String
- bosluk: Pragraph
+ SiparisYazdir(documentInfo: DocumentInfo, name: String)
+ buildHeader(): void
+ buildContents(): void
+ buildFooter(): void
}


DocumentManager o-- DocumentBuilder
DocumentBuilder <|-- TeklifYazdir
DocumentBuilder <|-- SiparisYazdir
DocumentInfo ..> Kablo
DocumentInfo <-down- TeklifYazdir
DocumentInfo <-down- SiparisYazdir
```

```
@enduml
```

Kablo.java.

```
public class Kablo {

    private String kabloTip;
    private String kabloYapi;
    private String fiberTip;
    private String sagKonnektor;
    private String solKonnektor;
    private String kabloCapi;
    private int kabloMetre;

    public Kablo(String kabloTip, String kabloYapi, String fiberTip,
 String sagKonnektor, String solKonnektor, String kabloCapi, int
 kabloMetre) {
        this.kabloTip = kabloTip;
        this.kabloYapi = kabloYapi;
        this.fiberTip = fiberTip;
        this.sagKonnektor = sagKonnektor;
        this.solKonnektor = solKonnektor;
        this.kabloCapi = kabloCapi;
        this.kabloMetre = kabloMetre;
    }

    public String getKabloTip() {
        return kabloTip;
    }

    public String getKabloYapi() {
        return kabloYapi;
    }

    public String getFiberTip() {
        return fiberTip;
    }

    public String getSagKonnektor() {
        return sagKonnektor;
    }

    public String getSolKonnektor() {
        return solKonnektor;
```

```
    }

    public String getKabloCapi() {
        return kabloCapi;
    }

    public int getKabloMetre() {
        return kabloMetre;
    }

    public void getUrun(){
    System.out.println("Kablo Tipi: " +  kabloTip);
    System.out.println("Kablo Yapısı: " + kabloYapi );
    System.out.println("Fiber Tipi: " + fiberTip);
    System.out.println("Sağ Konnektor: " + sagKonnektor);
    System.out.println("Sol Konnektor: " + solKonnektor);
    System.out.println("Kablo Çapı: " + kabloCapi);
    System.out.println("Kablo Uzunluğu: " + kabloMetre );
    }
}
```

DocumentInfo.java.

```
public class DocumentInfo {

    private String musteriAdi;
    private String musteriTel;
    private String musteriAdres;
    private List<Kablo> kablolar;

    public DocumentInfo(String musteriAdi, String musteriTel, String
 musteriAdres, List<Kablo> kablolar) {
        this.musteriAdi = musteriAdi;
        this.musteriTel = musteriTel;
        this.musteriAdres = musteriAdres;
        this.kablolar = kablolar;
    }


    public String getMusteriAdi() {
        return musteriAdi;
    }

    public void setMusteriAdi(String musteriAdi) {
        this.musteriAdi = musteriAdi;
```

```
    }

    public String getMusteriTel() {
        return musteriTel;
    }

    public void setMusteriTel(String musteriTel) {
        this.musteriTel = musteriTel;
    }

    public String getMusteriAdres() {
        return musteriAdres;
    }

    public void setMusteriAdres(String musteriAdres) {
        this.musteriAdres = musteriAdres;
    }

    public List<Kablo> getKablolar() {
        return kablolar;
    }

    public void setKablolar(List<Kablo> kablolar) {
        this.kablolar = kablolar;
    }



}
```

DocumentManager.java.

```
public class DocumentManager {

    private DocumentBuilder builder;

    public DocumentManager(DocumentBuilder builder)
        {
            this.builder = builder;
        }

    public void build()
        {
            builder.buildDocument();
```

```
        }

}
```

SiparisYazdir.java.

```
public class SiparisYazdir extends DocumentBuilder {

    Document document = new Document();
    Font font = FontFactory.getFont(FontFactory.COURIER, 16,
 BaseColor.BLACK);
    String name;
    Paragraph bosluk = new Paragraph(" ");

    public SiparisYazdir(DocumentInfo documentInfo, String name) throws
 FileNotFoundException, DocumentException {
        super(documentInfo);
        this.name = name;

        PdfWriter.getInstance(document, new
 FileOutputStream("Siparis.pdf"));

        document.open();
        Chunk chunk = new Chunk("Siparis", font);
        Paragraph para = new Paragraph(chunk);
        document.add(para);

    }

    @Override
    public void BuildHeader() {

        Date date = new Date();
        String musteriBilgileri = super.documentInfo.getMusteriAdi()
+ " " + super.documentInfo.getMusteriAdres() + " "
+ super.documentInfo.getMusteriTel();
        Chunk header = new Chunk("Siparis Hazirlayan: " + name, font);
        Paragraph para = new Paragraph(header);
        Chunk header2 = new Chunk(" Tarih:" + date, font);
        Paragraph para2 = new Paragraph(header2);
        Chunk header3 = new Chunk("Musteri Bilgileri: " +
 musteriBilgileri, font);
        Paragraph para3 = new Paragraph(header3);

        try {
```

```
            document.add(bosluk);
            document.add(para);
            document.add(bosluk);
            document.add(para2);
            document.add(bosluk);
            document.add(para3);
            document.add(bosluk);

        } catch (DocumentException ex) {

Logger.getLogger(SiparisYazdir.class.getName()).log(Level.SEVERE, null,
ex);
        }

    }

    @Override
    public void BuildFooter() {

        Chunk footer = new Chunk("Siparisiniz en gec 10 is günü icinde
elinize ulasacaktir. Fiyata %18 KDV dahil degildir.", font);
        Paragraph para4 = new Paragraph(footer);
        try {
            document.add(para4);
            document.add(bosluk);
            document.close();
        } catch (DocumentException ex) {

Logger.getLogger(SiparisYazdir.class.getName()).log(Level.SEVERE, null,
ex);
        }

    }

    @Override
    public void BuildContents() {

        String urun_ = "";
        List<Kablo> list = super.documentInfo.getKablolar();
        for (Kablo a : list) {
            Chunk contents = new Chunk(a.getFiberTip() + " " +
a.getKabloCapi() + " " + a.getKabloTip() + " " + a.getKabloYapi()
+ " " + a.getSagKonnektor() + " " + a.getSolKonnektor() + " " +
String.valueOf(a.getKabloMetre()));
            Paragraph para3 = new Paragraph(contents);
            try {
```

```
                document.add(para3);
            } catch (DocumentException ex) {

 Logger.getLogger(SiparisYazdir.class.getName()).log(Level.SEVERE, null,
 ex);
            }
        }

    }

}
```

TeklifYazdir.java.

```
public class TeklifYazdir extends DocumentBuilder {
    Document document = new Document();
    Font font = FontFactory.getFont(FontFactory.COURIER, 16,
 BaseColor.BLACK);
    String name;
    Paragraph bosluk = new Paragraph(" ");

    public TeklifYazdir(DocumentInfo documentInfo, String name) throws
 FileNotFoundException, DocumentException {
        super(documentInfo);
        this.name = name;

        PdfWriter.getInstance(document, new
 FileOutputStream("Teklif.pdf"));

        document.open();
        Chunk chunk = new Chunk("Teklif", font);
        Paragraph para = new Paragraph(chunk);
        document.add(para);

    }

    @Override
    public void BuildHeader() {

        Date date = new Date();
        String musteriBilgileri = super.documentInfo.getMusteriAdi()
 + " " + super.documentInfo.getMusteriAdres() + " "
 + super.documentInfo.getMusteriTel();
        Chunk header = new Chunk("Teklif Hazirlayan: " + name, font);
        Paragraph para = new Paragraph(header);
```

```
        Chunk header2 = new Chunk(" Tarih:" + date, font);
        Paragraph para2 = new Paragraph(header2);
        Chunk header3 = new Chunk("Musteri Bilgileri: " +
musteriBilgileri, font);
        Paragraph para3 = new Paragraph(header3);

        try {
            document.add(bosluk);
            document.add(para);
            document.add(bosluk);
            document.add(para2);
            document.add(bosluk);
            document.add(para3);
            document.add(bosluk);

        } catch (DocumentException ex) {

Logger.getLogger(SiparisYazdir.class.getName()).log(Level.SEVERE, null,
ex);
        }

    }

    @Override
    public void BuildFooter() {

        Chunk footer = new Chunk("Teklifimiz 1 hafta süre geçerlidir.
Fiyata %18 KDV dahil degildir.", font);
        Paragraph para4 = new Paragraph(footer);
        try {
            document.add(para4);
            document.add(bosluk);
            document.close();
        } catch (DocumentException ex) {

Logger.getLogger(SiparisYazdir.class.getName()).log(Level.SEVERE, null,
ex);
        }

    }

    @Override
    public void BuildContents() {

        String urun_ = "";
        List<Kablo> list = super.documentInfo.getKablolar();
```

```
        for (Kablo a : list) {
            Chunk contents = new Chunk(a.getFiberTip() + " " +
 a.getKabloCapi() + " " + a.getKabloTip() + " " + a.getKabloYapi()
 + " " + a.getSagKonnektor() + " " + a.getSolKonnektor() + " " +
 String.valueOf(a.getKabloMetre()));
            Paragraph para3 = new Paragraph(contents);
            try {
                document.add(para3);
            } catch (DocumentException ex) {

 Logger.getLogger(SiparisYazdir.class.getName()).log(Level.SEVERE, null,
 ex);
            }
        }

    }


}
```

DocumentBuilder.java.

```
public abstract class DocumentBuilder {

    protected DocumentInfo documentInfo;


      public DocumentBuilder(DocumentInfo documentInfo)
        {
            this.documentInfo = documentInfo;
        }

       public void buildDocument() {

         BuildHeader();
         BuildContents();
         BuildFooter();

    }

    public abstract void BuildHeader();

    public abstract void BuildFooter();

    public abstract void BuildContents();
```

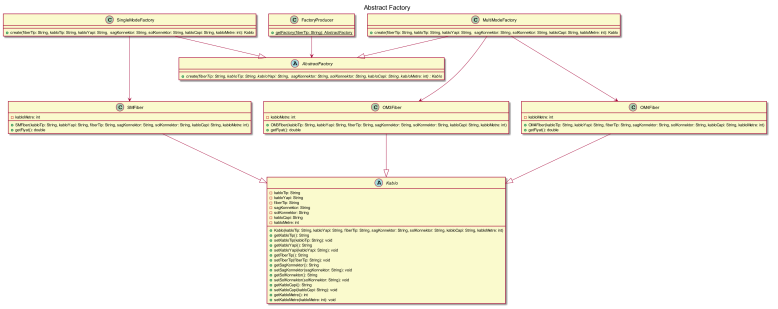
```
}
```

Abstract Factory

Bu tasarım deseni birbiriyle alakalı veya bağımlı nesnelerin somut sınıflarını belirtmeden, yaratılması için gereken bir arayüz sağlar. Ayrıca bu desene fabrikaların fabrikası(factories of the factory) da denir.

Neden Kullanıldı?

İki tip bulunan fiber kıllarından bir çeşit üretmek için bu tasarım deseni tercih edilmiştir. Abstract Factory tercih ettiğimiz fabrikaya ulaşarak istediğimiz tipte fiber kablo üretilmesine olanak sağlar. Yeni bir çeşit üretmek istediğimizde koda yeni bir fabrika ve bu fabrika altında bir tür eklememiz yeterli olacaktır.



plantuml code.

```
@startuml
scale 1.5
title Abstract Factory

abstract class Kablo{
- kabloTip: String
- kabloYapi: String
- fiberTip: String
- sagKonnektor: String
- solKonnektor: String
- kabloCapi: String
```

```
- kabloMetre: int
+ Kablo(kabloTip: String, kabloYapi: String, fiberTip: String,
 sagKonnektor: String, solKonnektor: String, kabloCapi: String,
 kabloMetre: int)
+ getKabloTip(): String
+ setKabloTip(kabloTip: String): void
+ getKabloYapi(): String
+ setKabloYapi(kabloYapi: String): void
+ getFiberTip(): String
+ setFiberTip(fiberTip: String): void
+ getSagKonnektor(): String
+ setSagKonnektor(sagKonnektor: String): void
+ getSolKonnektor(): String
+ setSolKonnektor(solKonnektor: String): void
+ getKabloCapi(): String
+ setKabloCapi(kabloCapi: String): void
+ getKabloMetre(): int
+ setKabloMetre(kabloMetre: int): void
}

class SMFiber{
- kabloMetre: int
+ SMFiber(kabloTip: String, kabloYapi: String, fiberTip: String,
 sagKonnektor: String, solKonnektor: String, kabloCapi: String,
 kabloMetre: int)
+ getFiyat(): double

}

class OM3Fiber{
- kabloMetre: int
+ OM3Fiber(kabloTip: String, kabloYapi: String, fiberTip: String,
 sagKonnektor: String, solKonnektor: String, kabloCapi: String,
 kabloMetre: int)
+ getFiyat(): double

}

class OM4Fiber{
- kabloMetre: int
+ OM4Fiber(kabloTip: String, kabloYapi: String, fiberTip: String,
 sagKonnektor: String, solKonnektor: String, kabloCapi: String,
 kabloMetre: int)
+ getFiyat(): double

}
```

```
abstract class AbstractFactory{

{abstract} + create(fiberTip: String, kabloTip: String, kabloYapi:
 String,  sagKonnektor: String, solKonnektor: String, kabloCapi: String,
 kabloMetre: int) : Kablo

}

class MultiModeFactory {
+ create(fiberTip: String, kabloTip: String, kabloYapi: String,
  sagKonnektor: String, solKonnektor: String, kabloCapi: String,
 kabloMetre: int): Kablo
}

class SingleModeFactory {
+ create(fiberTip: String, kabloTip: String, kabloYapi: String,
  sagKonnektor: String, solKonnektor: String, kabloCapi: String,
 kabloMetre: int): Kablo
}

class FactoryProducer{
{static} + getFactory(fiberTip: String): AbstractFactory
}

OM3Fiber ---|> Kablo
OM4Fiber ---|> Kablo
SMFiber ---|> Kablo
FactoryProducer --> AbstractFactory
MultiModeFactory --|> AbstractFactory
SingleModeFactory --|> AbstractFactory
MultiModeFactory --->  OM3Fiber
MultiModeFactory ---> OM4Fiber
SingleModeFactory ---> SMFiber
@enduml
```

Kablo.java.

```
public abstract class Kablo {

    private String kabloTip;
    private String kabloYapi;
    private String fiberTip;
    private String sagKonnektor;
    private String solKonnektor;
```

```
    private String kabloCapi;
    private int kabloMetre;

    protected Kablo(String kabloTip, String kabloYapi, String fiberTip,
String sagKonnektor, String solKonnektor, String kabloCapi, int
kabloMetre) {
        this.kabloTip = kabloTip;
        this.kabloYapi = kabloYapi;
        this.fiberTip = fiberTip;
        this.sagKonnektor = sagKonnektor;
        this.solKonnektor = solKonnektor;
        this.kabloCapi = kabloCapi;
        this.kabloMetre = kabloMetre;
    }

    public String getKabloTip() {
        return kabloTip;
    }

    public String getKabloYapi() {
        return kabloYapi;
    }

    public String getFiberTip() {
        return fiberTip;
    }

    public String getSagKonnektor() {
        return sagKonnektor;
    }

    public String getSolKonnektor() {
        return solKonnektor;
    }

    public String getKabloCapi() {
        return kabloCapi;
    }

    public int getKabloMetre() {
        return kabloMetre;
    }

    public void getUrun(){
    System.out.println("Kablo Tipi: " +  kabloTip);
    System.out.println("Kablo Yapısı: " + kabloYapi );
```

```
    System.out.println("Fiber Tipi: " + fiberTip);
    System.out.println("Sağ Konnektor: " + sagKonnektor);
    System.out.println("Sol Konnektor: " + solKonnektor);
    System.out.println("Kablo Çapı: " + kabloCapi);
    System.out.println("Kablo Uzunluğu: " + kabloMetre );
    }


}
```

OM3Fiber.java.

```
public class OM3Fiber extends Kablo {

    public OM3Fiber(String kabloTip, String kabloYapi, String
 sagKonnektor, String solKonnektor, String kabloCapi, int kabloMetre) {
        super(kabloTip, kabloYapi, "OM3", sagKonnektor, solKonnektor,
 kabloCapi, kabloMetre);
    }

    private int kabloMetre;

    public double getFiyat(){
        return 1.7 * kabloMetre;
    }
}
```

OM4Fiber.java.

```
public class OM4Fiber extends Kablo {

    public OM4Fiber(String kabloTip, String kabloYapi, String
 sagKonnektor, String solKonnektor, String kabloCapi, int kabloMetre) {
        super(kabloTip, kabloYapi, "OM4", sagKonnektor, solKonnektor,
 kabloCapi, kabloMetre);
    }

    private int kabloMetre;

    public double getFiyat(){
        return 1.9 * kabloMetre;
    }
}
```

SMFiber.java.

```
public class SMFiber extends Kablo {

    public SMFiber(String kabloTip, String kabloYapi, String
 sagKonnektor, String solKonnektor, String kabloCapi, int kabloMetre) {
        super(kabloTip, kabloYapi, "Single Mode", sagKonnektor,
 solKonnektor, kabloCapi, kabloMetre);
        this.kabloMetre = kabloMetre;
    }

    private int kabloMetre;

    public double getFiyat(){
        return 1.5 * kabloMetre;
    }
}
```

AbstractFactory.java.

```
public abstract class AbstractFactory {
    abstract Kablo create(String fiberTip, String kabloTip, String
 kabloYapi, String sagKonnektor, String solKonnektor, String
 kabloCapi, int kabloMetre) ;
}
```

SingleModeFactory.java.

```
public class SingleModeFactory extends AbstractFactory {

    @Override
    Kablo create(String fiberTip, String kabloTip, String kabloYapi,
 String sagKonnektor, String solKonnektor, String kabloCapi, int
 kabloMetre) {
        if(fiberTip.equalsIgnoreCase("SM")){
         return new SMFiber(kabloTip, kabloYapi, sagKonnektor,
 solKonnektor, kabloCapi, kabloMetre);
      }
      return null;
    }

}
```

MultiModeFactory.java.

```
public class MultiModeFactory extends AbstractFactory{
```

```
    @Override
    Kablo create(String fiberTip, String kabloTip, String kabloYapi,
 String sagKonnektor, String solKonnektor, String kabloCapi, int
 kabloMetre) {
        if(fiberTip.equalsIgnoreCase("OM3")){
         return new OM3Fiber(kabloTip, kabloYapi, sagKonnektor,
 solKonnektor, kabloCapi, kabloMetre);
      }else if(fiberTip.equalsIgnoreCase("OM4")){
         return new OM4Fiber(kabloTip, kabloYapi, sagKonnektor,
 solKonnektor, kabloCapi, kabloMetre);
      }
      return null;
    }

}
```

FactoryProducer.java.

```
public class FactoryProducer {

    public static AbstractFactory getFactory(String fiberTip){
      if(fiberTip.equals("SM")){
         return new SingleModeFactory();
      }else{
         return new MultiModeFactory();
      }
   }
}
```

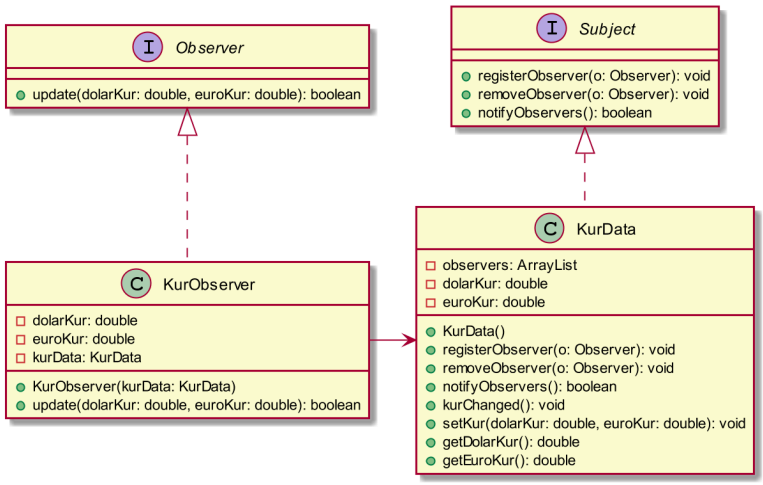
1.2. Davranışsal (Behavioral) Tasarım Kalıpları

- Iterator
- Observer
- Memento
- Chain Of Responsibility
- Command
- Visitor
- Mediator

Observer

Bir nesnenin durumunun değişmesi ile o nesneye bağlı diğer nesnelerin bu değişimi bilmesini istiyorsak, böyle durumlarda bu tasarım desenini kullanabiliriz.

Neden Kullanıldı? Kur değişimi gerçekleştiğinde seçilmiş olan kablo maliyetinin güncellenmesi için tercih edilmiştir.



plantuml code.

```
interface Subject{
+ registerObserver(o: Observer): void
+ removeObserver(o: Observer): void
+ notifyObservers(): boolean
}

interface Observer {
+ update(dolarKur: double, euroKur: double): boolean
}

class KurData{
- observers: ArrayList
- dolarKur: double
```

```
- euroKur: double
+ KurData()
+ registerObserver(o: Observer): void
+ removeObserver(o: Observer): void
+ notifyObservers(): boolean
+ kurChanged(): void
+ setKur(dolarKur: double, euroKur: double): void
+ getDolarKur(): double
+ getEuroKur(): double
}

class KurObserver{
- dolarKur: double
- euroKur: double
- kurData: KurData
+ KurObserver(kurData: KurData)
+ update(dolarKur: double, euroKur: double): boolean
}

KurData -up-|> Subject
KurObserver -up-|> Observer
KurObserver -> KurData


@enduml
```

Subject.java.

```
public interface Subject {

    public void registerObserver(Observer o);

    public void removeObserver(Observer o);

    public boolean notifyObservers();
}
```

Observer.java.

```
public interface Observer {

    public boolean update(double dolarKur, double euroKur);
}
```

KurObserver.java.

```
public class KurObserver implements Observer {

    private double dolarKur = 0;
    private double euroKur = 0;

    private KurData kurData;

    public KurObserver(KurData kurData) {
        this.kurData = kurData;
        kurData.registerObserver(this);

    }

    @Override
    public boolean update(double dolarKur, double euroKur) {
        if (dolarKur != this.dolarKur || euroKur != this.euroKur) {
            this.dolarKur = dolarKur;
            this.euroKur = euroKur;
            return true;
        } else {
            return false;
        }
    }

}
```

KurData.java.

```
public class KurData implements Subject {

    private ArrayList observers;
    private double dolarKur;
    private double euroKur;

    public KurData() {
        observers = new ArrayList();
    }

    @Override
    public void registerObserver(Observer o) {
        observers.add(o);
    }

    @Override
    public void removeObserver(Observer o) {
```

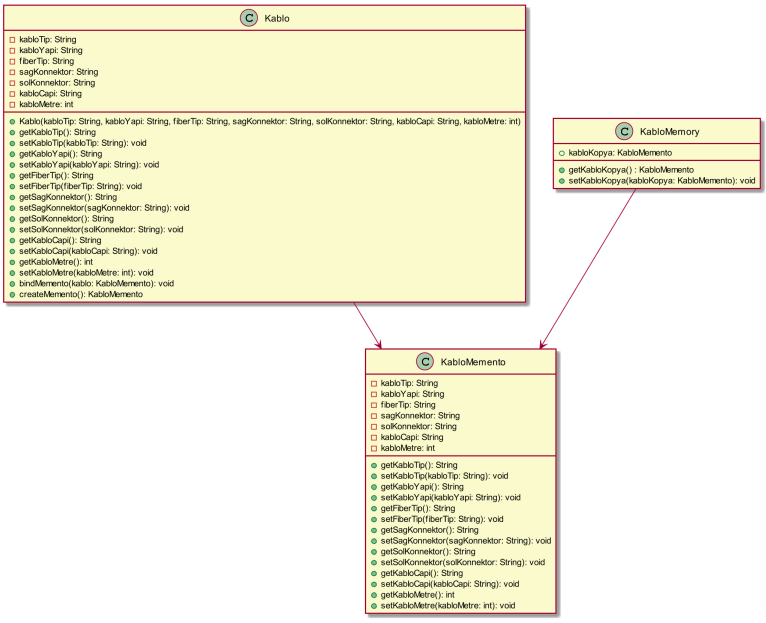
```
        int i = observers.indexOf(o);
        if (i >= 0) {
            observers.remove(i);
        }
    }

    @Override
    public boolean notifyObservers() {
        for (int i = 0; i < observers.size(); i++) {
            Observer observer = (Observer) observers.get(i);
            return observer.update(dolarKur, euroKur);
        }
        return false;
    }

    public void kurChanged() {
        notifyObservers();
    }

    public void setKur(double dolarKur, double euroKur) {
        this.dolarKur = dolarKur;
        this.euroKur = euroKur;
        kurChanged();
    }

    public double getDolarKur() {
        return dolarKur;
    }

    public double getEuroKur() {
        return euroKur;
    }

}
```

Memento

Bir nesnenin tamamının veya bazı özelliklerinin tutularak sonradan tekrar elde edilmesini yöneten tasarım desenidir

Neden Kullanıldı?

Kablo üretiminde bir değişiklik yapıldıktan sonra en son yapılan değişikliği geri almak için tercih edilmiştir.



plantuml code.

```
@startuml

class Kablo{
- kabloTip: String
- kabloYapi: String
- fiberTip: String
- sagKonnektor: String
- solKonnektor: String
- kabloCapi: String
- kabloMetre: int
+ Kablo(kabloTip: String, kabloYapi: String, fiberTip: String,
 sagKonnektor: String, solKonnektor: String, kabloCapi: String,
 kabloMetre: int)
+ getKabloTip(): String
+ setKabloTip(kabloTip: String): void
+ getKabloYapi(): String
```

```
+ setKabloYapi(kabloYapi: String): void
+ getFiberTip(): String
+ setFiberTip(fiberTip: String): void
+ getSagKonnektor(): String
+ setSagKonnektor(sagKonnektor: String): void
+ getSolKonnektor(): String
+ setSolKonnektor(solKonnektor: String): void
+ getKabloCapi(): String
+ setKabloCapi(kabloCapi: String): void
+ getKabloMetre(): int
+ setKabloMetre(kabloMetre: int): void
+ bindMemento(kablo: KabloMemento): void
+ createMemento(): KabloMemento
}

class KabloMemento{
- kabloTip: String
- kabloYapi: String
- fiberTip: String
- sagKonnektor: String
- solKonnektor: String
- kabloCapi: String
- kabloMetre: int
+ getKabloTip(): String
+ setKabloTip(kabloTip: String): void
+ getKabloYapi(): String
+ setKabloYapi(kabloYapi: String): void
+ getFiberTip(): String
+ setFiberTip(fiberTip: String): void
+ getSagKonnektor(): String
+ setSagKonnektor(sagKonnektor: String): void
+ getSolKonnektor(): String
+ setSolKonnektor(solKonnektor: String): void
+ getKabloCapi(): String
+ setKabloCapi(kabloCapi: String): void
+ getKabloMetre(): int
+ setKabloMetre(kabloMetre: int): void
}

class KabloMemory {
+ kabloKopya: KabloMemento
+ getKabloKopya() : KabloMemento
+ setKabloKopya(kabloKopya: KabloMemento): void
}

KabloMemory --> KabloMemento
```

```
Kablo --> KabloMemento


@enduml
```

Kablo.java.

```
public class Kablo {

    private String kabloTip;
    private String kabloYapi;
    private String fiberTip;
    private String sagKonnektor;
    private String solKonnektor;
    private String kabloCapi;
    private int kabloMetre;

    public Kablo(String kabloTip, String kabloYapi, String fiberTip,
 String sagKonnektor, String solKonnektor, String kabloCapi, int
 kabloMetre) {
        this.kabloTip = kabloTip;
        this.kabloYapi = kabloYapi;
        this.fiberTip = fiberTip;
        this.sagKonnektor = sagKonnektor;
        this.solKonnektor = solKonnektor;
        this.kabloCapi = kabloCapi;
        this.kabloMetre = kabloMetre;
    }

    public String getKabloTip() {
        return kabloTip;
    }

    public String getKabloYapi() {
        return kabloYapi;
    }

    public String getFiberTip() {
        return fiberTip;
    }

    public String getSagKonnektor() {
        return sagKonnektor;
    }
```

```
public String getSolKonnektor() {
    return solKonnektor;
}

public String getKabloCapi() {
    return kabloCapi;
}

public int getKabloMetre() {
    return kabloMetre;
}

public void getUrun(){
System.out.println("Kablo Tipi: " +  kabloTip);
System.out.println("Kablo Yapısı: " + kabloYapi );
System.out.println("Fiber Tipi: " + fiberTip);
System.out.println("Sağ Konnektor: " + sagKonnektor);
System.out.println("Sol Konnektor: " + solKonnektor);
System.out.println("Kablo Çapı: " + kabloCapi);
System.out.println("Kablo Uzunluğu: " + kabloMetre );
}

public void bindMemento(KabloMemento kablo){

    this.fiberTip = kablo.getFiberTip();
    this.kabloCapi = kablo.getKabloCapi();
    this.kabloMetre = kablo.getKabloMetre();
    this.kabloTip = kablo.getKabloTip();
    this.kabloYapi = kablo.getKabloTip();
    this.sagKonnektor = kablo.getSagKonnektor();
    this.solKonnektor = kablo.getSolKonnektor();

}

public KabloMemento createMemento(){
    KabloMemento kb= new KabloMemento();
    kb.setFiberTip(this.fiberTip);
    kb.setKabloCapi(this.kabloCapi);
    kb.setKabloMetre(this.kabloMetre);
    kb.setKabloTip(this.kabloTip);
    kb.setKabloYapi(this.kabloYapi);
    kb.setSagKonnektor(this.sagKonnektor);
    kb.setSolKonnektor(this.solKonnektor);

    return kb;
```

```
    }


}
```

KabloMemory.java.

```
public class KabloMemory {
        public KabloMemento kabloKopya;

    public KabloMemento getKabloKopya() {
        return kabloKopya;
    }

    public void setKabloKopya(KabloMemento kabloKopya) {
        this.kabloKopya = kabloKopya;
    }

}
```

KabloMemento.java.

```
public class KabloMemento {
    private String kabloTip;
    private String kabloYapi;
    private String fiberTip;
    private String sagKonnektor;
    private String solKonnektor;
    private String kabloCapi;
    private int kabloMetre;

    public String getKabloTip() {
        return kabloTip;
    }

    public void setKabloTip(String kabloTip) {
        this.kabloTip = kabloTip;
    }

    public String getKabloYapi() {
        return kabloYapi;
    }

    public void setKabloYapi(String kabloYapi) {
```

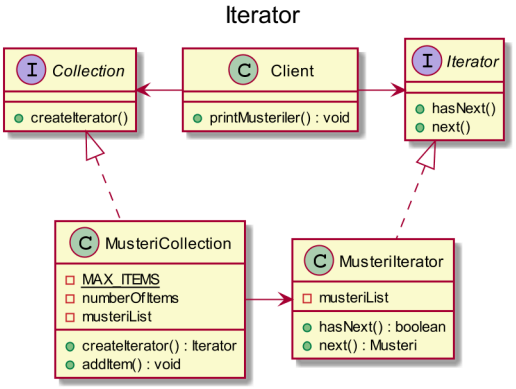
```
        this.kabloYapi = kabloYapi;
    }

    public String getFiberTip() {
        return fiberTip;
    }

    public void setFiberTip(String fiberTip) {
        this.fiberTip = fiberTip;
    }

    public String getSagKonnektor() {
        return sagKonnektor;
    }

    public void setSagKonnektor(String sagKonnektor) {
        this.sagKonnektor = sagKonnektor;
    }

    public String getSolKonnektor() {
        return solKonnektor;
    }

    public void setSolKonnektor(String solKonnektor) {
        this.solKonnektor = solKonnektor;
    }

    public String getKabloCapi() {
        return kabloCapi;
    }

    public void setKabloCapi(String kabloCapi) {
        this.kabloCapi = kabloCapi;
    }

    public int getKabloMetre() {
        return kabloMetre;
    }

    public void setKabloMetre(int kabloMetre) {
        this.kabloMetre = kabloMetre;
    }


}
```

Iterator

Iterator tasarım deseni kullanılarak koleksiyonun array, queue, list olması önemli olmadan, aynı şekilde elemanlarının elde edilmesi sağlanır. Koleksiyon içindeki nesnelerin nasıl elde edileceği tercihe göre belirlenebilir.

Neden Kullanıldı?

Müşteri listesini tutmak ve ekleme yapmak amacıyla kullanıldı. Bu sayede teklif veya sipariş kaydı açılan ürüne müşteri seçilerek ataması yapılabilecek.



plantuml code.

```
@startuml
interface Collection
class Client
interface Iterator
together {
class MusteriCollection
class MusteriIterator
}

Collection  : + createIterator()
```

```
Iterator : + hasNext()
Iterator : + next()

MusteriIterator : + hasNext() : boolean
MusteriIterator : + next() : Musteri
MusteriIterator : - musteriList

MusteriCollection : + createIterator() : Iterator
MusteriCollection : - {static}  MAX_ITEMS
MusteriCollection : - numberOfItems
MusteriCollection : - musteriList
MusteriCollection : + addItem() : void

Client : + printMusteriler() : void
Collection  <- Client
Client -> Iterator
MusteriCollection -> MusteriIterator
Iterator <|.. MusteriIterator
Collection  <|.. MusteriCollection

@enduml
```

Collection.java.

```
interface Collection
{
    public Iterator createIterator();
}
```

Iterator.java.

```
public interface Iterator
{
    boolean hasNext();
    Object next();
}
```

Musteri.java.

```
public class Musteri {
    // To store musteri
    String adSoyad;
    String firma;
    String tel;
    String mail;
```

```
    public Musteri(String adSoyad, String firma, String tel, String
 mail) {
        this.adSoyad = adSoyad;
        this.firma = firma;
        this.tel = tel;
        this.mail = mail;
    }



    public String getAdSoyad() {
        return adSoyad;
    }

    public String getFirma() {
        return firma;
    }

    public String getTel() {
        return tel;
    }

    public String getMail() {
        return mail;
    }


}
```

MusteriCollection.java.

```
public class MusteriCollection implements Collection {

    static final int MAX_ITEMS = 1000;
    int numberOfItems = 0;
    Musteri[] musteriList;

    public MusteriCollection()
    {
        musteriList = new Musteri[MAX_ITEMS];

        // Let us add some dummy notifications
       Musteri m=new Musteri("Tayfun
 Erkorkmaz", "SAMM", "tayfun.erkorkmaz@samm.com", "+905552343978");
```

```
        Musteri m2=new
 Musteri("Test2", "SAMM", "test2@samm.com", "+905551234567");
        Musteri m3=new
 Musteri("Test3", "SAMM", "test3@samm.com", "+905558910111");

        addItem(m);
        addItem(m2);
        addItem(m3);

    }

    public void addItem(Musteri m)
    {
        Musteri notification = new Musteri(m.adSoyad, m.firma, m.mail,
 m.tel);
        if (numberOfItems >= MAX_ITEMS)
            System.err.println("Maksimum Kayıt");
        else
        {
            musteriList[numberOfItems] = notification;
            numberOfItems = numberOfItems + 1;
        }
    }

    @Override
    public Iterator createIterator() {

      return new MusteriIterator(musteriList);

    }

}
```

MusteriIterator.java.

```
public class MusteriIterator implements Iterator {

    Musteri[] musteriList;

    int pos = 0;

    public MusteriIterator(Musteri[] musteriList) {
        this.musteriList = musteriList;
    }
```

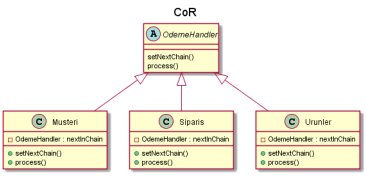
```
    public Object next()
    {
        // return next element in the array and increment pos
        Musteri musteri =  musteriList[pos];
        pos += 1;
        return musteri;
    }

    public boolean hasNext()
    {
        if (pos >= musteriList.length ||
            musteriList[pos] == null)
            return false;
        else
            return true;
    }

}
```

Chain of Responsibility

Bir isteğin duruma göre farklı şekillerde işlem yapılması gereken durumlarda kullanılır. Bu tasarım deseninde isteğe cevap verebilecek sınıflar aynı arayüzü kullanır ve isteğin durumuna göre ya cevap verir ya da isteği zincirdeki sonraki nesneye gönderir.

Neden Kullanıldı?

Önce Müşteri bilgilerini ardından Ürünleri alarak Siparişi oluşturur. Zincir Müşteriden başlayarak Siparişte son bulur. Müşteriden başlanmadığı durumda zincir çalışmaz ve hata alınır.



plantuml code.

```
@startuml
title CoR
skinparam componentStyle uml2

abstract class OdemeHandler
class Musteri
class Siparis
class Urunler

OdemeHandler : setNextChain()
OdemeHandler : process()

Musteri : - OdemeHandler : nextInChain
Musteri : +setNextChain()
Musteri : +process()

Siparis : - OdemeHandler : nextInChain
Siparis : +setNextChain()
Siparis : +process()

Urunler : - OdemeHandler : nextInChain
Urunler : +setNextChain()
Urunler : +process()

OdemeHandler <|-- Musteri
OdemeHandler <|-- Siparis
OdemeHandler <|-- Urunler

@enduml
```

OdemeHandler.java.

```
public interface OdemeHandler {

    void setNextChain(OdemeHandler nextChain);
    void process(Object obj);
}
```

Musteri.java.

```
public class Musteri implements OdemeHandler{

    private OdemeHandler nextInChain;
```

```
    @Override
    public void setNextChain(OdemeHandler nextChain) {
        nextInChain = nextChain;
    }


    @Override
    public void process(Object obj) {
        if (obj instanceof Musteri || obj instanceof Siparis )
        {
            System.out.println("Once Urunleri Girmelisiniz");
        }
        else
        {
            nextInChain.process(new Urunler());

        }
    }

}
```

Siparis.java.

```
public class Siparis implements OdemeHandler {
    private OdemeHandler nextInChain;

    @Override
    public void setNextChain(OdemeHandler nextChain) {
        nextInChain = nextChain;
    }


    @Override
    public void process(Object obj) {
        if (obj instanceof Musteri || obj instanceof Urunler)
        {
            System.out.println("Sipariş Tamamlanamadı");
        }
        else
        {
            System.out.println("Sipariş Tamamlandı");

        }
    }
}
```

Urunler.java.

```
public class Urunler implements OdemeHandler {
    private OdemeHandler nextInChain;

    @Override
    public void setNextChain(OdemeHandler nextChain) {
        nextInChain = nextChain;
    }


    @Override
    public void process(Object obj) {
        if (obj instanceof Musteri || obj instanceof Siparis )
        {
              System.out.println("Önce Müşteri Bilgisi Girmelisiniz");
        }
        else
        {
            nextInChain.process(new Siparis());

        }
    }
}
```

1.3. Yapısal (Structural) Tasarım Kalıpları

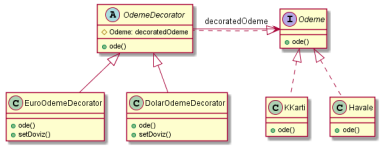
Adapter
Decorator
Component

Decorator

Bir nesneye dinamik olarak yeni özellikler eklemek için kullanılır. Kalıtım kullanmadan da bir nesnenin görevlerini artırabileceğimizi gösterir.

Neden Kullanıldı?

Herhangi bir döviz ödemesinde, ödemeyi havale veya kredi kartı ile gerçekleştirmek için nesneye yeni özellikler tanımlayabiliriz. Yeni bir ödeme çeşiti eklemek istediğimizde "Odeme" arayüzü üzerinden bir ekleme yapmamız yeterli olacaktır.



plantuml code.

```
@startuml
skinparam componentStyle uml2

interface Odeme
class KKarti
class Havale
abstract class OdemeDecorator
class EuroOdemeDecorator
class DolarOdemeDecorator

Odeme : +ode()

KKarti : +ode()

Havale : +ode()

OdemeDecorator : +ode()
OdemeDecorator : #Odeme: decoratedOdeme

EuroOdemeDecorator : +ode()
EuroOdemeDecorator : +setDoviz()

DolarOdemeDecorator : +ode()
DolarOdemeDecorator : +setDoviz()

Odeme <|.. KKarti
Odeme <|.. Havale
Odeme <|.. OdemeDecorator
```

```
OdemeDecorator <|-- EuroOdemeDecorator
OdemeDecorator <|-- DolarOdemeDecorator
OdemeDecorator -> Odeme : decoratedOdeme
@enduml
```

Odeme.java.

```
public interface Odeme {
    void ode();
}
```

OdemeDecorator.java.

```
public abstract class OdemeDecorator implements Odeme {
    protected Odeme decoratedOdeme;

    public OdemeDecorator(Odeme decoratedOdeme){
    this.decoratedOdeme = decoratedOdeme;
    }

    public void ode(){
        decoratedOdeme.ode();
    }
}
```

Havale.java.

```
public class Havale implements Odeme {

    @Override
    public void ode(int miktar) {
        System.out.println("Havale ile");
    }

}
```

KKarti.java.

```
public class KKarti implements Odeme{

    @Override
    public void ode(int miktar) {
        System.out.println("Kredi Kartı ile");
    }
```

```
}
```

EuroOdemeDecorator.java.

```
public class EuroOdemeDecorator extends OdemeDecorator {


    public EuroOdemeDecorator(Odeme decoratedOdeme) {
        super(decoratedOdeme);
    }

    @Override
    public void ode(int miktar){
    decoratedOdeme.ode(miktar);
    setDoviz(decoratedOdeme, miktar);
    }

    private void setDoviz(Odeme decoratedOdeme, int miktar){
        System.out.println(miktar + " Euro");
    }

}
```

DolarOdemeDecorator.java.

```
public class DolarOdemeDecorator extends OdemeDecorator {


    public DolarOdemeDecorator(Odeme decoratedOdeme) {
        super(decoratedOdeme);
    }

    @Override
    public void ode(int miktar){
    decoratedOdeme.ode(miktar);
    setDoviz(decoratedOdeme, miktar);
    }

    private void setDoviz(Odeme decoratedOdeme, int miktar){
        System.out.println(miktar + " Dolar");
    }

}
```