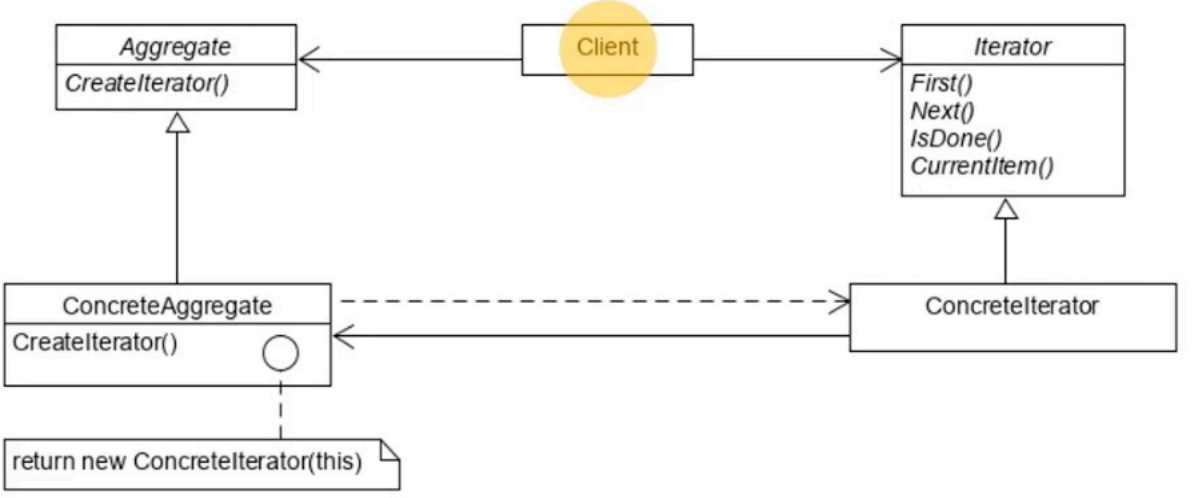
Iterator Deseni

- Iterator: Yineleyici, tekrarlayıcı
- Diğer bilinen adı: Cursor; imleç
- Amaç: koleksiyonun alt yapısını teşhir etmeden elemanlarına sıralı erişim sağlamak
- Koleksiyonda farklı şekilde gezinmek mümkün
- Koleksiyona erişim ve gezinme sorumluluğu koleksiyondan iterator'a geçiyor.
- Nesne Davranışsal

Iterator Deseni



Iterator Deseni

Aggregate.java

Listeleme.java

Iterator.java

Dizi.java

DiziIterator.java

Agac.java

AgacIterator.java

Aggregate.java | Iterator.java

```
1 public abstract class Iterator {
2     public abstract void First();       /* imleci ilk elemana getir */
3     public abstract void Next();        /* imleci bir sonraki elemana
4     public abstract boolean IsDone();   /* koleksiyonun sonuna geldik
5     public abstract int CurrentItem();  /* imlecin şimdi gösterdiği de
6 }
7
```

```java
public class DiziIterator extends Iterator {
    private Dizi dizi;
    private int idx;

    public DiziIterator(Dizi d) {
        dizi = d;
        idx = 0;
    }

    @Override
    public void First() { idx = 0; }

    @Override
    public void Next() { idx++; }

    @Override
    public boolean IsDone() { return idx == dizi.liste.size(); }

    @Override
    public int CurrentItem() { return dizi.liste.get(idx); }
}
```

Iterator Deseni

- Kütüphanelerde iterator kullanımı
- Dikkatli kullanım
- Ek Iterator yöntemleri

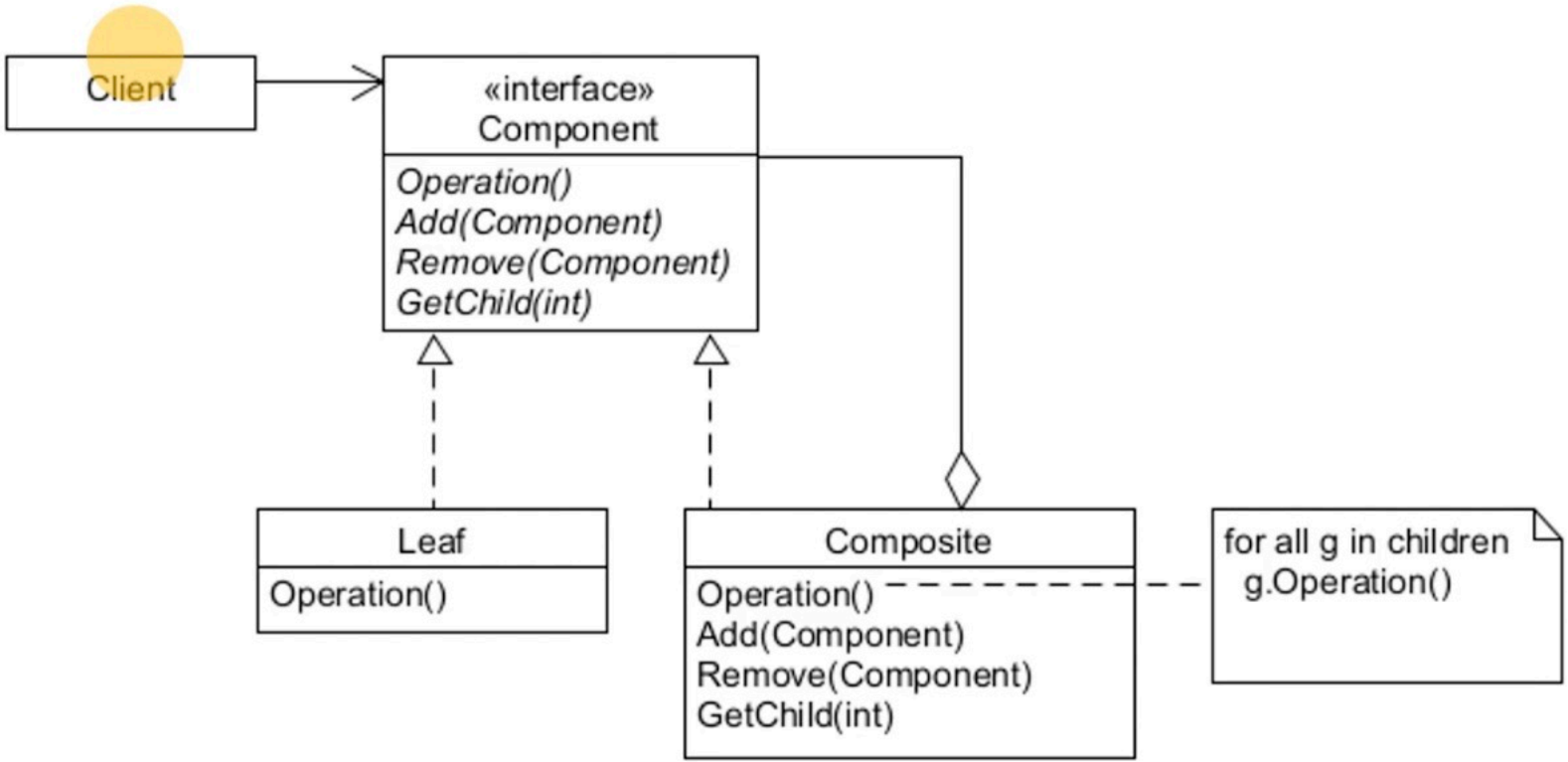
İlgili Desenler

- Composite
- Factory

Composite Deseni

- Composite: Bileşim, birçok parçadan oluşan
- Amaç: Bütüne ve bütünün parçalarına aynı şekilde erişim sağlamak
- Nesne Yapısal
- Pencere
  - Etiket
  - Metin satırı
  - Açılır menü
    - Düğme
    - Metin satırı

Composite Deseni



```
1 public interface Icerik {
2     void Operation();
3     void Add(Icerik i);
4     void Remove(Icerik i);
5     Icerik GetChild(int i);
6 }
7
```

```java
import java.util.ArrayList;

public class Bolum implements Icerik {

    private String baslik;
    private ArrayList<Icerik> icerikler;
    public Bolum(String b) {
        baslik = b;
        icerikler = new ArrayList<Icerik>();
    }

    @Override
    public void Operation() {
        System.out.println("<b>" + baslik + "</b>");
        for(Icerik i:icerikler) {
            i.Operation();
        }
    }

    @Override
    public void Add(Icerik i) {
        icerikler.add(i);
    }

    @Override
    public void Remove(Icerik i) {
        icerikler.remove(i);
    }

    @Override
    public Icerik GetChild(int i) {
        return icerikler.get(i);
    }

}
```

```java
public class Metin implements Icerik {

    private String icerik;
    public Metin(String i) {
        icerik = i;
    }

    @Override
    public void Operation() {
        System.out.println("<p>" + icerik + "</p>");
    }

    @Override
    public void Add(Icerik i) {
        return;
    }

    @Override
    public void Remove(Icerik i) {
        return;
    }

    @Override
    public Icerik GetChild(int i) {
        return null;
    }

}
```

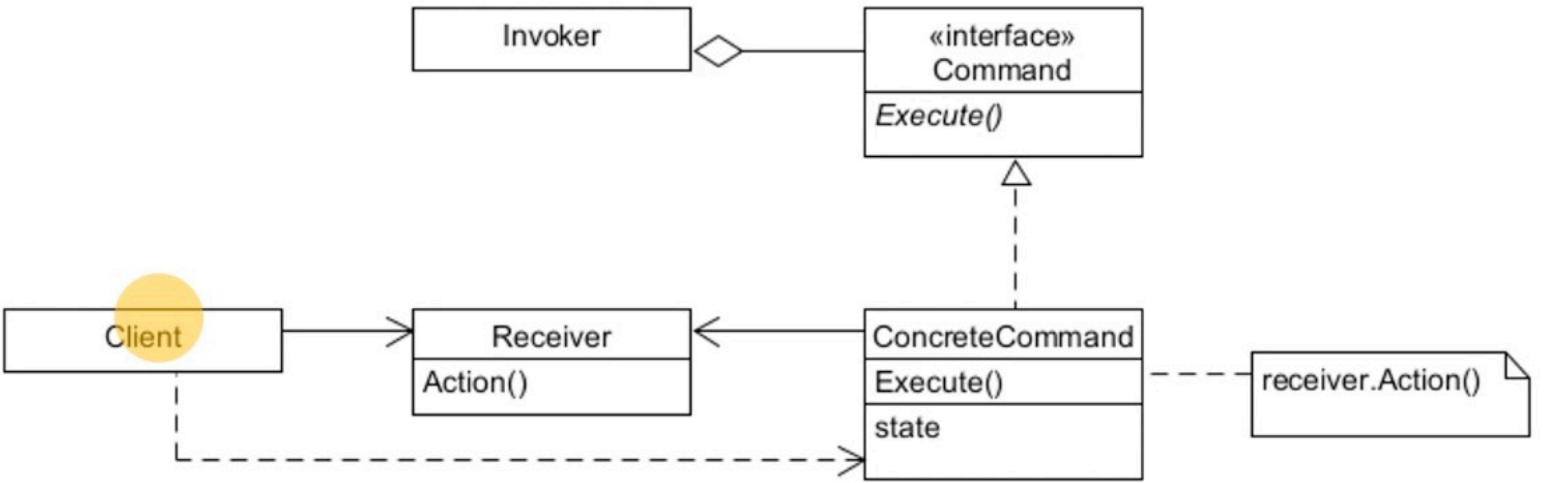
```java

public class Resim implements Icerik {

    private String resimyolu;

    public Resim(String r) {
        resimyolu = r;
    }

    @Override
    public void Operation() {
        System.out.println("<img src='" + resimyolu + "' />");
    }

    @Override
    public void Add(Icerik i) {
        return;
    }

    @Override
    public void Remove(Icerik i) {
        return;
    }

    @Override
    public Icerik GetChild(int i) {
        return null;
    }

}

```

```java
public class Main {

    public static void main(String[] args) {
        Bolum kitap = new Bolum("Programlamaya Giriş");
        Bolum b1 = new Bolum("Giriş");
        b1.Add(new Metin("hello, world"));
        b1.Add(new Metin("diger programlar"));
        Bolum b12 = new Bolum("Temel Programlama");
        b12.Add(new Metin("değişkenler"));
        b12.Add(new Metin("kontrol"));
        b12.Add(new Metin("döngüler"));
        b12.Add(new Metin("fonksiyonlar"));
        b1.Add(b12);

        Bolum b2 = new Bolum("Değişkenler");
        b2.Add(new Resim("degisken.jpg"));
        b2.Add(new Metin("int, double, float, boolean"));

        kitap.Add(b1);
        kitap.Add(b2);

        yazdir(kitap);
        System.out.println();
        yazdir(b12);
    }

    public static void yazdir(Icerik icerik) {
        icerik.Operation();
    }
}
```

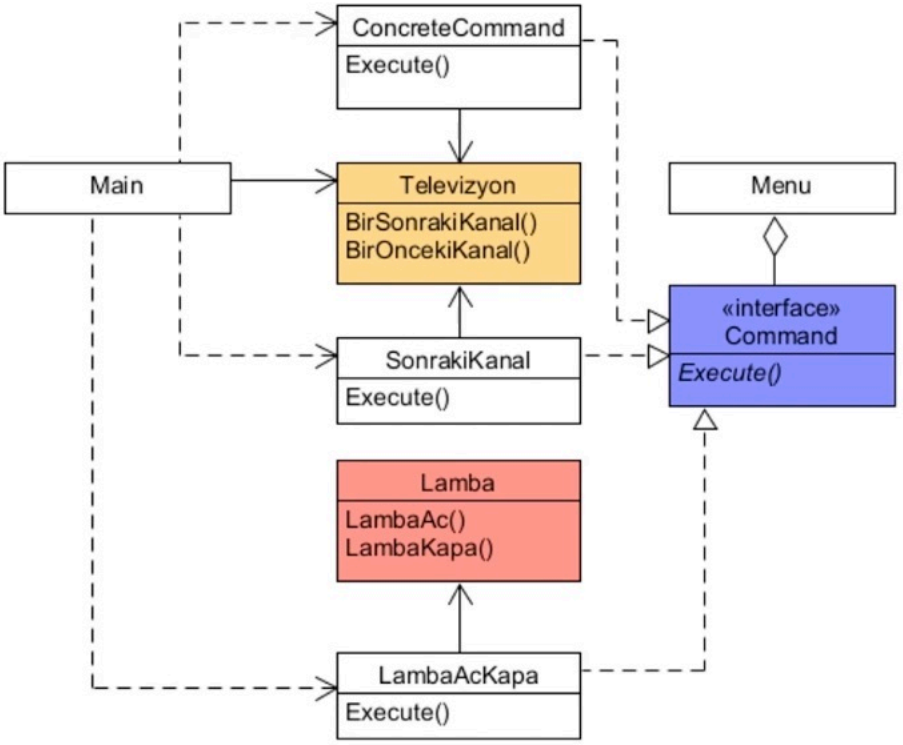
Command Deseni

- Command: komut, emir.
- Diğer bilinen adları: Action (eylem), Transaction (işlem)
- Amaç: Bir isteği, ya da komutu, nesne olarak tanımlamak
- İstek nesnesi
- Nesne Davranışsal

Command Deseni



Command Deseni



Command.java | Menu.java

```java
import java.util.ArrayList;

public class Menu {
    private ArrayList<Command> komutlar;
    public Menu() {
        komutlar = new ArrayList<Command>();
    }

    public void komutEkle(Command c) {
        komutlar.add(c);
    }

    public void menu() {
        Scanner sc = new Scanner(System.in);
        int s = 0;
        while(s != -1) {
            for (int i=0;i<komutlar.size();i++) {
                System.out.println(i + ": " + komutlar.get(i));
            }
            System.out.println("Çıkmak için -1");
            try {
                s = sc.nextInt();
                komutlar.get(s).Execute();
            } catch(Exception e) {}
        }
        sc.close();
    }
}

```

Command.java | Menu.java | Lamba.java

```java

public class Lamba {
    private int lamba;
    public Lamba() {
        lamba = 0;
    }
    public void lambaAc() {
        System.out.println("Lamba açıldı");
        lamba = 1;
    }

    public void lambaKapa() {
        System.out.println("Lamba kapandı");
        lamba = 0;
    }

    public int lambaDegeri() { return lamba; }
}

```

```java

public class Televizyon {
    private int kanal;
    public Televizyon() {
        kanal = 1;
    }

    public void kanalDegistir(int k) {
        kanal = k;
    }

    public void birSonrakiKanal() {
        kanal++;
    }

    public void birOncekiKanal() {
        kanal--;
    }

    public int gecerliKanal() { return kanal; }
}

```

```java
public class LambaAcKapa implements Command {

    Lamba lamba;

    public LambaAcKapa(Lamba l) {
        lamba = l;
    }

    @Override
    public void Execute() {
        // TODO Auto-generated method stub
        if(lamba.lambaDegeri() == 1) lamba.lambaKapa();
        else lamba.lambaAc();
    }

    @Override
    public String toString() {
        if(lamba.lambaDegeri()==1) return "Lambayı Kapa";
        else return "Lambayı Aç";
    }

}
```

```java
public class OncekiKanal implements Command {

private Televizyon tv;

    public OncekiKanal(Televizyon t) {
        tv = t;
    }

    @Override
    public void Execute() {
        tv.birOncekiKanal();
        System.out.println("Geçerli Kanal: " + tv.gecerliKanal());
    }

    @Override
    public String toString() {
        return "Önceki Kanal";
    }

}
```

```java
public class SonrakiKanal implements Command {
    private Televizyon tv;

    public SonrakiKanal(Televizyon t) {
        tv = t;
    }

    @Override
    public void Execute() {
        tv.birSonrakiKanal();
        System.out.println("Geçerli Kanal: " + tv.gecerliKanal());
    }

    @Override
    public String toString() {
        return "Sonraki Kanal";
    }

}
```

```java
public class Main {

    public static void main(String[] args) {
        Menu m = new Menu();
        LambaAcKapa komut_lamba = new LambaAcKapa(new Lamba());
        Televizyon tv = new Televizyon();
        OncekiKanal komut_once = new OncekiKanal(tv);
        SonrakiKanal komut_sonra = new SonrakiKanal(tv);

        m.komutEkle(komut_lamba);
        m.komutEkle(komut_once);
        m.komutEkle(komut_sonra);

        m.menu();

    }

}
```

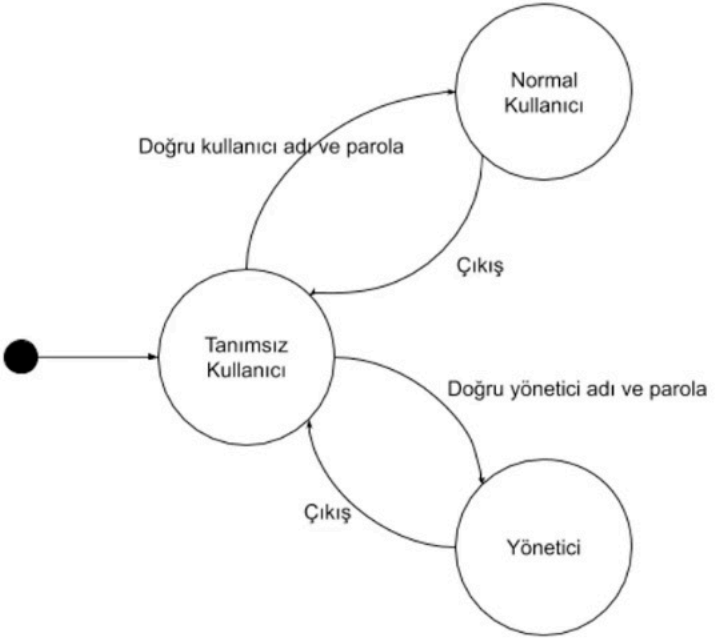
State Deseni

- State: durum, hal.
- Amaç: bir nesnenin iç durumu değişince davranışının da değişmesini sağlamak
- Nesne Davranışsal
- State değişkeni, s

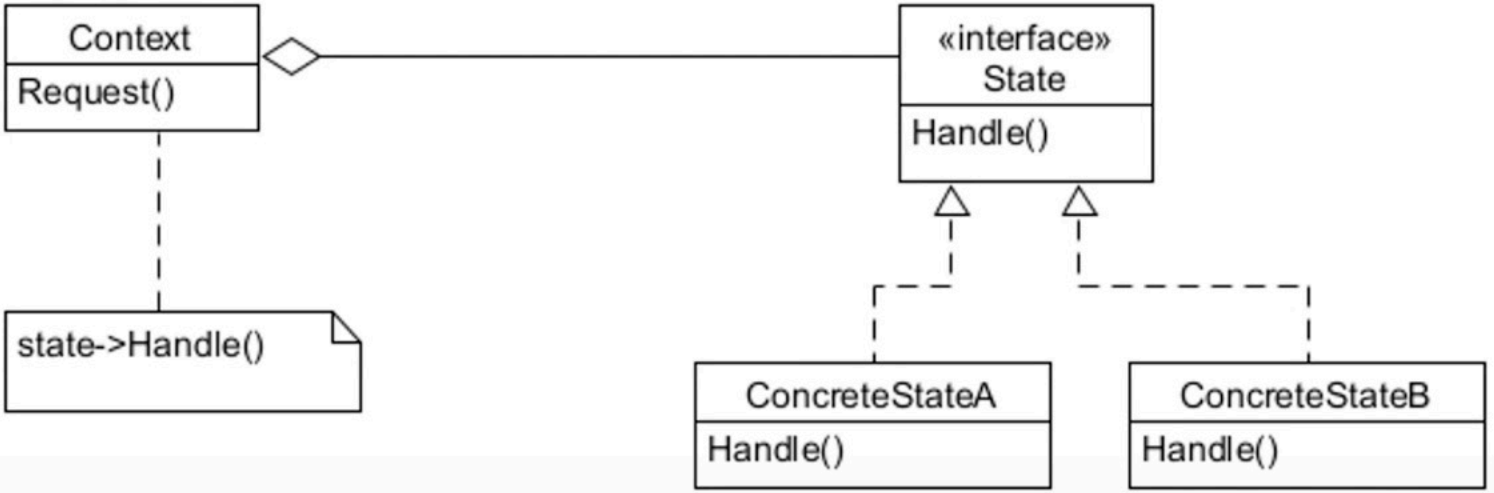
```
if (s==0) { … }
else if (s==1) { … }
else if (s==2) { … }
else { … }
```

State Deseni

- Finite State Machines, Sonlu Durum Makineleri



State Deseni



```
1 public interface State {
2     void Handle();
3 }
4
```

```
1
2  public class Program {
3      private State s;
4
5      public Program() {
6          s = new Tanimsiz(this);
7      }
8
9      public void calis() {
10         s.Handle();
11     }
12
13     public void GirisYap() {
14         s = new Kullanicdi(this);
15         calis();
16     }
17
18     public void Yonetici() {
19         s = new Yonetici(this);
20         calis();
21     }
22
23     public void Cikis() {
24         s = new Tanimsiz(this);
25     }
26 }
27
```

```java
public class Tanimsiz implements State {
    private Program program;

    public Tanimsiz(Program p) {
        program = p;
    }

    @Override
    public void Handle() {
        System.out.println("1) Kayıt Olun");
        System.out.println("2) Kullanıcı Girişi");
        System.out.println("3) Yönetici Girişi");

        program.GirisYap();
    }

}
```

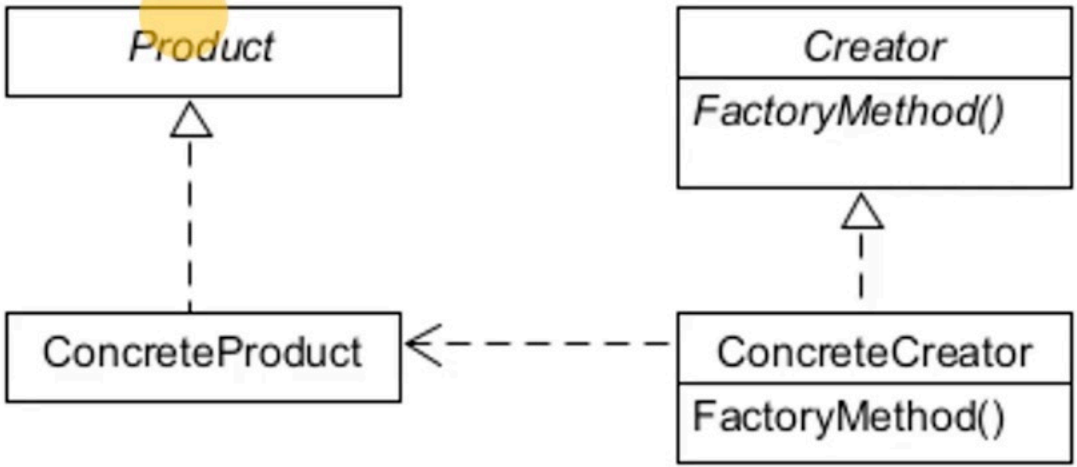
```java
public class Kullanici implements State {
    private Program program;

    public Kullanici(Program p) {
        program = p;
    }

    @Override
    public void Handle() {
        System.out.println("1) Kullanıcı Ayarları");
        System.out.println("2) Hizmet 1");
        System.out.println("3) Hizmet 2");
        System.out.println("4) Çıkış");

        program.Cikis();
    }

}
```

```java
public class Yonetici implements State {

    private Program program;

    public Yonetici(Program p) {
        program = p;
    }

    @Override
    public void Handle() {
        System.out.println("1) Kullanıcı Ekle");
        System.out.println("2) Kullanıcı Güncelle");
        System.out.println("3) Kullanıcı Sil");
        System.out.println("4) Çıkış");

        program.Cikis();
    }

}
```

Factory Method Deseni

- Factory: fabrika, imalathane, yapımevi, Method: yöntem
- Diğer bilinen adı: Virtual Constructor; sanal yapılandırıcı
- Sınıf Yaratımsal
- Amacı: Bir nesnenin yaratılması için tanımlanan ara yüzün altsınıflarına bu nesnenin hangi sınıftan olacağına karar verme hakkı sağlamak

Factory Method Deseni



```java
public abstract class Araba {
    public int maksHiz;
    public int anlikHiz;

    public abstract int maksHizAl();
    public abstract String marka();
    public abstract void hiziBelirle(int s);
}
```

Araba.java | Sahin.java

```java
public class Sahin extends Araba {
    public Sahin() {
        maksHiz = 180;
        anlikHiz = 0;
    }

    @Override
    public int maksHizAl() {
        return maksHiz;
    }

    @Override
    public String marka() {
        return "Şahin";
    }

    @Override
    public void hiziBelirle(int s) {
        anlikHiz = s;
    }
}
```

```java
public class Mercedes extends Araba {

    public Mercedes() {
        maksHiz = 260;
        anlikHiz = 0;
    }

    @Override
    public int maksHizAl() {
        return maksHiz;
    }

    @Override
    public String marka() {
        return "Mercedes";
    }

    @Override
    public void hiziBelirle(int s) {
        anlikHiz = s;
    }

}
```

```
1
2 public abstract class ArabaFactory {
3     public abstract Araba ArabaUret();
4 }
5
```

```
public class SahinFactory extends ArabaFactory {

    @Override
    public Araba ArabaUret() {
        return new Sahin();
    }

}
```

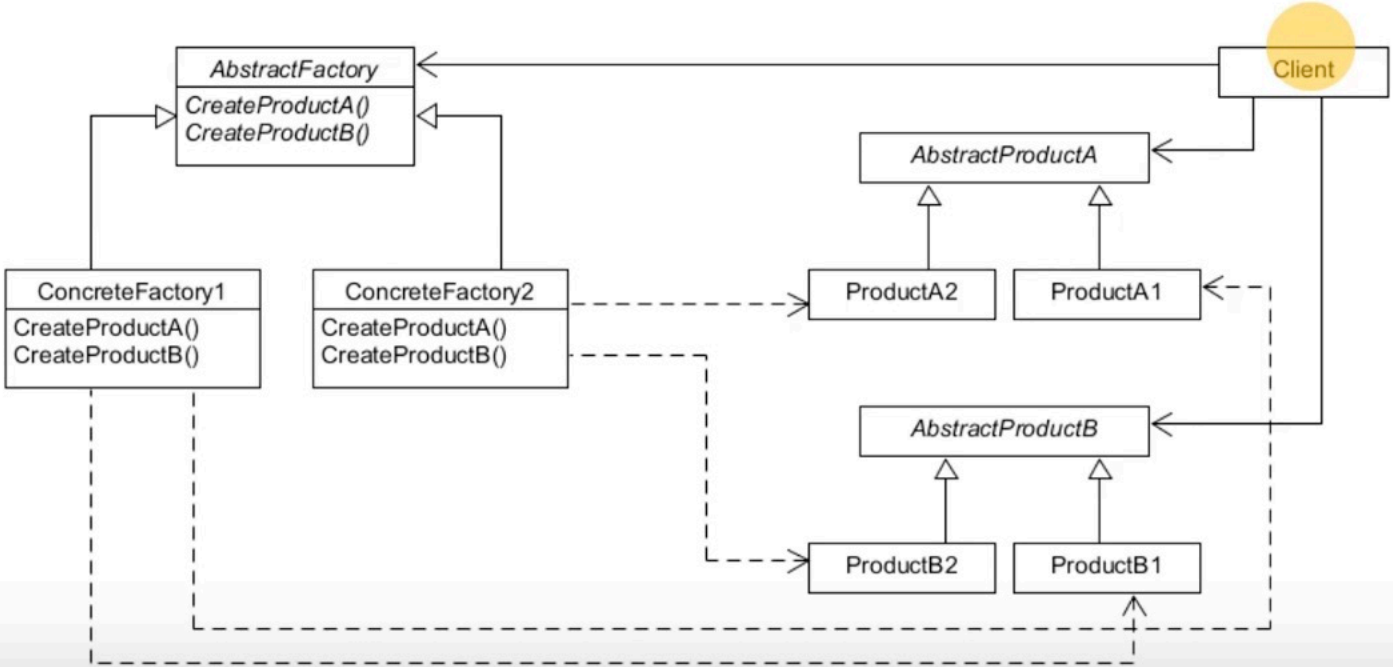
```java

public class MercedesFactory extends ArabaFactory {

    @Override
    public Araba ArabaUret() {
        return new Mercedes();
    }

}

```

```java
public class Main {
    public static void araba(ArabaFactory f) {
        Araba a = f.ArabaUret();
        System.out.println(a.marka() + " marka araba üretildi.
    }

    public static void main(String[] args) {
        araba(new SahinFactory());
        araba(new MercedesFactory());
    }
}
```

Abstract Factory Deseni

- Abstract: soyut, Factory: fabrika, imalathane, yapımevi…
- Diğer bilinen adı: Kit (takım çantası, alet takımı)
- Factory Method
- Nesne Yaratımsal
- Amacı: somut sınıflarını belirtmeden bağlantılı ya da bağımlı olan nesneler grubunu yaratacak bir arayüz sunmak.

Abstract Factory Deseni



```java
public class NormalKapi extends SoyutKapi {

    private boolean acik = false;

    @Override
    public boolean kapiAc() {
        acik = true;
        return acik;
    }

    @Override
    public void kapiKapat() {
        acik = false;
    }

}
```

```
1 public abstract class OyunFactory {
2     public abstract SoyutOyun yeniOyun();
3     public abstract SoyutKapi kapiYarat();
4     public abstract SoyutOda odaYarat();
5 }
6
```

OyunFactory.java | SoyutKapi.java

```java
public abstract class SoyutKapi {
    public abstract boolean kapiAc();
    public abstract void kapiKapat();
}

```

OyunFactory.java | SoyutKapi.java | NormalKapi.java | BuyuluKapi.java

```java
import java.util.Scanner;

public class BuyuluKapi extends SoyutKapi {

    private String sihirli;
    private boolean acik;

    public BuyuluKapi() {
        sihirli = "Açıl Susam Açıl!";
        acik = false;
    }

    @Override
    public boolean kapiAc() {
        if(acik) return true;
        System.out.println("Kapıyı açmak için sihirli cümleyi söyleyin
        Scanner scanner = new Scanner(System.in);
        String k = scanner.nextLine();
        scanner.close();
        if(k.equalsIgnoreCase(sihirli)) {
            acik = true;
            return acik;
        }
        else return false;
    }

    @Override
    public void kapiKapat() {
        acik = false;
    }

}
```

```java
public abstract class SoyutOda {
    public abstract void kapiEkle(SoyutKapi k);
    public abstract boolean kapiAc();
}

```

```java
public class NormalOda extends SoyutOda {

    SoyutKapi kapi;

    @Override
    public void kapiEkle(SoyutKapi k) {
        kapi = k;
    }

    @Override
    public boolean kapiAc() {
        return kapi.kapiAc();
    }

}
```

```java
public class BuyuluOda extends SoyutOda {

    SoyutKapi kapi;

    @Override
    public void kapiEkle(SoyutKapi k) {
        kapi = k;
    }

    @Override
    public boolean kapiAc() {
        return kapi.kapiAc();
    }

}
```

NormalKapi.java | BuyuluKapi.java | SoyutOda.java | NormalOda.java | BuyuluOda.java

```java
public class NormalOda extends SoyutOda {

    SoyutKapi kapi;

    @Override
    public void kapiEkle(SoyutKapi k) {
        kapi = k;
    }

    @Override
    public boolean kapiAc() {
        return kapi.kapiAc();
    }

}
```

BuyuluKapi.java | SoyutOda.java | NormalOda.java | BuyuluOda.java | SoyutOyun.j

```java
public abstract class SoyutOyun {
    public abstract void odaEkle(SoyutOda a);
    public abstract void oyunCalistir();
}

```

```java
import java.util.ArrayList;

public class NormalOyun extends SoyutOyun {

    ArrayList<SoyutOda> odalar;

    NormalOyun() {
        odalar = new ArrayList<SoyutOda>();
    }

    @Override
    public void odaEkle(SoyutOda a) {
        odalar.add(a);
    }

    @Override
    public void oyunCalistir() {
        System.out.println("Normal oyun çalışıyor...");
        System.out.println("Oda sayısı: " + odalar.size());
    }

}
```

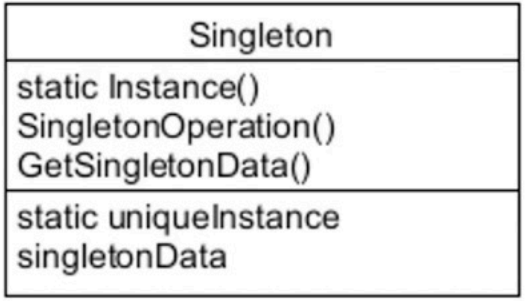
```java
import java.util.ArrayList;

public class BuyuluOyun extends SoyutOyun {

    ArrayList<SoyutOda> odalar;

    public BuyuluOyun() {
        odalar = new ArrayList<SoyutOda>();
    }

    @Override
    public void odaEkle(SoyutOda a) {
        odalar.add(a);
    }

    @Override
    public void oyunCalistir() {
        System.out.println("Buyulu oyun calisiyor...");
        if(odalar.size() > 0) {
            if(odalar.get(0).kapiAc()) {
                System.out.println("kapi acildi!");
            } else System.out.println("kapi acilmadi!");
        }
    }

}
```

Singleton Deseni

- Singleton: tekil, tekil kalıp
- Nesne yaratımsal
- Amacı: Bir sınıftan yalnızca bir nesnenin olması ve bu nesneye global erişim için bir nokta sağlanması
- Neden?
- Nasıl?

Singleton Deseni

- Tek bir sınıf
- Static Instance() { return uniqueInstance; }
- Çok iş parçacıklı (multithreaded)
- Static



TekilSinif.java

```java
public class TekilSinif {

    private int deger;
    private TekilSinif() {
        deger = 12;
    }

    public int degerAl() { return deger; }
    public void degerYaz(int a) {
        deger = a;
    }

    // singleton
    private static TekilSinif uniqueInstance = null;
    public static TekilSinif Instance() {
        if(uniqueInstance == null) {
            uniqueInstance = new TekilSinif();
        }
        return uniqueInstance;
    }

}

```

TekilSinif.java | *Main.java

```java
public class Main {

    public static void main(String[] args) {
        // TekilSinif t = new TekilSinif();
        TekilSinif t = TekilSinif.Instance();
        System.out.println(t.degerAl());
        t.degerYaz(20);
        TekilSinif u = TekilSinif.Instance();
        System.out.println(u.degerAl());
    }

}
```