Template Metod Tasarım Deseni

Nedir?

Template Metod tasarım deseni, **behavioral** tasarım desenlerinden biridir. Bir algoritmanın **adımlarını** tanımlamayı sağlar ve alt sınıfların bir veya daha fazla adımların implementasyonunu yapmasını olanak tanır. **Örneğin**, bir ev inşa etmek istiyoruz. Ev inşa etmek için şu adımlar gereklidir: bina temel atılması, bina sütunları, bina duvarları ve pencereler. Burada **dikkat** edilecek olan husus, adımların **sırayla** yapılmasının **zorunlu** olmasıdır. Pencereler takıldıktan sonra temel atılamaz, önce temel atılır sonra bina sütunları yapılır, daha sonra duvarlar yapılır en son ise pencereler yapılır. Template metod tasarımı ile bu adımların sırasıyla yapılması zorunlu hale getirilmiştir.

Ne zaman Kullanılır?

Benzer adımlardan oluşan sınıfları tasarlarken, aynı kodları her sınıfta tekrar tekrar kullanmayı engellemek için kullanılır. Örneğin, kahve ve çay hazırlamakla ilgili sınıflar ortak özelliklere sahiptir. Her ikisinde de sırasıyla şu işlemler yapılır: Su kaynatma, demleme, bardağa dökme ve isteğe göre çeşni ekleme.

Nasıl Kullanılır?

Türü **abstract** olan bir sınıf yaratılır. Bu sınıf, içerisinde aynı adımlara sahip olan işlemleri temsil eder. Her işlem için bir metod eklenir. İşlemlerin sırasıyla yapılmasını zorunlu yapabilmek için ise **template metodu** eklenir. Template metodu işlem adımlarının **zorunlu** yapılması için **final** türüne sahip olur. Bu sayede alt sınıflar **template metodu** **override** edemezler. Template metod içerisinde, işlem adımları için oluşturulmuş metodlar sırasıyla çağrılır.

Not: Kısaca ifade etmek gerekirse, template metod algoritma, diğer metodlar ise algoritma adımlarıdır.

Not: Template metod dışındaki diğer metodlardan bazıları **abstract** tanımlanabilir. Kahve ve çay hazırlama örneğinde bazı adımları(çeşni ekleme adımı gibi) alt sınıflar kendi amaçlarına göre farklı şekilde implement ederler.

Not: Template metod dışındaki diğer metodlardan bazıları **final** tanımlanabilir. Final tanımlanan metod alt sınıflar tarafından **override** edilmeyeceği için tüm alt sınıflar o adımı değiştirmeden kullanırlar.

Hook Kavramı

Hook(Kanca), **abstract** sınıf içinde tanımlanmış, içi boş veya **default** implementasyona sahip bir metodtur. Bu metod alt sınıflar tarafından **override** edilebilir. Hook kullanmanın faydaları şunlardır:
1. Algoritmanın işleyişini alt sınıfın değiştirebilmesi sağlanır.
2. Algoritmanın opsiyonel parçasının olması sağlanır.
3. Alt sınıfa, gerçekleşmiş veya gerçekleşek bazı adımlara tepki verme şansı verir.

Faydaları Nedir?

1. Kod tekrarından kurtarır.
2. **final** keyword sayesinde standart bir tasarım sağlar.

Gerekenler

Türü **abstract** olan, template metodu **içeren** süper sınıf
En az iki tane **somut** Template sınıfı
Client sınıfı

Örnek Kullanım Alanları

1. **java.io.InputStream, java.io.OutputStream, java.io.Reader** ve **java.io.Writer** sınıflarının **abstract** olmayan metodları
2. **java.util.AbstractList, java.util.AbstractSet** ve **java.util.AbstractMap** sınıflarının **abstract** olmayan metodları

Örnek Uygulama

Soyut Sınıf

Template metodunu içinde barındıran sınıftır.

```
1  /**
2   * Soyut sinifi
3   */
4  public abstract class Education {
5      /**
6       * template metodu
7       * final tanimlandi bu sayede alt siniflar override edemeyecek.
8       */
9      final void template(){
10         elementary();
11         secondary();
12         highSchool();
13         collage();
14         //Hook kullanimi.
```

```
        //Goruldugu gibi algoritmayi alt sinifin degistirebilmesi saglandi
        if(hasMaster()){
            master();
        }
    }
    public abstract void elementary();
    //final tanimlanarak alt siniflarin override edebilmesi engellendi
    //Bu sayede alt siniflar metodu degistiremeyecek
    public final void secondary(){
        System.out.println("İkisi de Niksar Anadolu Ortaokulu'na gitti");
    }
    protected abstract void collage();
    protected abstract void highSchool();
    //Hook metodu. Default implementasyona sahip
    protected boolean hasMaster(){
        return true;
    }
    protected abstract void master();
}
```

Somut Sınıflar

Template metod içerisinde bulunan algoritma adımlarını implement eden alt sınıflardır.

```
/**
 * Somut sinif
 */
public class Ahmet extends Education {
    @Override
    public void elementary() {
        System.out.println("Ahmet Büyük Ata İlkokulu'na gitti");
    }

    @Override
    protected void collage() {
        System.out.println("Ahmet Gazi Üniversitesi'nde okudu");
    }

    @Override
    protected void highSchool() {
        System.out.println("Ahmet Niksar Anadolu Lisesi'nde okudu");
    }

    @Override
    protected boolean hasMaster() {
        return false; //Ahmet yüksek lisans yapmadığı için false donderdik
    }

    @Override
    protected void master() {
        System.out.println("Ahmet yüksek lisans yaptı");
    }
}
```

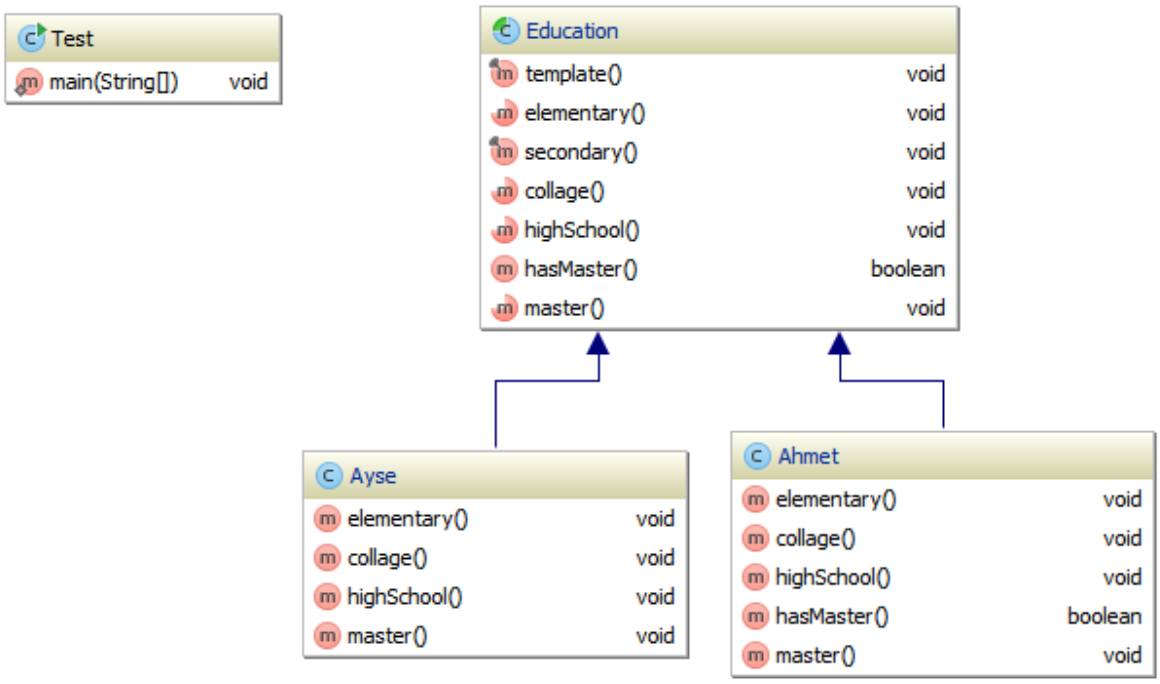
```
/**
 * Somut sinif
 */
public class Ayse extends Education {
    @Override
    public void elementary() {
        System.out.println("Ayşe Danişment Gazi İlkokulu'na gitti");
    }
    @Override
    protected void collage() {
        System.out.println("Ayşe Anadolu Üniversitesi'ne gitti");
    }

    @Override
    protected void highSchool() {
        System.out.println("Ayşe Ankara Fen Lisesi'ni gitti");
    }


    @Override
    protected void master() {
        System.out.println("Ayşe yüksek lisans yaptı");
    }
}
```

Test Sınıfı

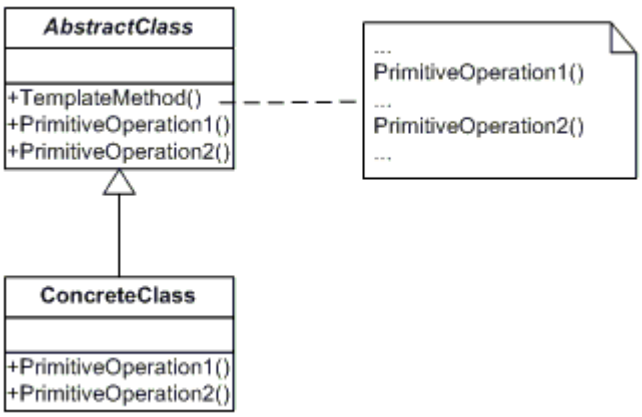
```
/**
 * Client sinifimiz
 */
public class Test {
    public static void main(String[] args) {
        Education ahmet=new Ahmet();
        ahmet.template();
```

```
 8
 9          Education ayse=new Ayse();
10          ayse.template();
11      }
12  }
```

Uygulamamızın UML Diagramı



Template Metod Tasarım Deseni'nin Şematik Gösterimi



Not: Template metod tasarım deseninde, **abstract** sınıfta implement edilen metodlar, alt sınıflar tarafından, hook mantığı yoksa, override edilmemelidir.

Hollywood Prensibi

Çoğu zaman, alt sınıflar süper sınıfın metodlarını çağırırken, template tasarım deseninde, süper sınıfın template metodu alt sınıfların metodunu çağırır. Bu işlem Hollywood Prensibi(Siz bizi aramayın, biz sizi ararız) olarak bilinir.