NESNE YÖNELİMLİ PROGRAMLAMA 2(Object Oriented Programming 2/OOP)

Öğr. Gör. Celil ÖZTÜRK
Marmara Üniversitesi
Teknik Bilimler Meslek Yüksekokulu

İçerik

- Null Object Pattern
- Chain of Responsibility Pattern
- Anti Patterns

Tasarım Kalıpları

Behavioral Patterns(Davranışsal Kalıplar)

- Chain of Responsibility Pattern
- Command Pattern
- Interpreter Pattern
- Iterator Pattern
- Mediator Pattern
- Memento Pattern
- Observer Pattern
- Null Object Pattern
- **Strategy Pattern**
- **State Pattern**
- Visitor Pattern

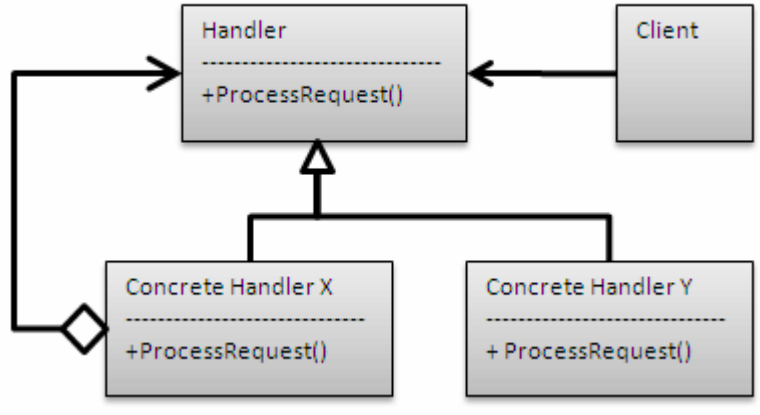
Sorumluluk Zinciri Kalıbı (Chain of responsibility Pattern)

- Sorumluluk zinciri kalıbı sisteme gönderilen bir isteğin (komut) hangi nesne tarafından cevaplanması gerektiğini bilmediğimiz durumlarda ya da isteği yapan nesne ve servis sağlayan nesne arasında sıkı bir bağ oluşmasını engellememiz gerektiğinde kullanılmaktadır.
- Bu tasarım kalıbında servis sağlayan ilgili tüm nesneler birbirleriyle ilişkili hale getirilir.
- Bir nesne zincirdeki kendinden sonraki nesneyi tanır ve isteğe cevap veremediği durumda, kendinden sonraki nesneye iletir. Bu işlem, zincirde bulunan doğru servis sağlayıcı nesneyi bulana kadar devam eder.(bidb.itu.edu.tr)

https://bidb.itu.edu.tr/seyir-defteri/blog/2013/09/08/davran%C4%B1%C5%9Fsal-kal%C4%B1plar-(behavioral-patterns)

Sorumluluk Zinciri Kalıbı (Chain of responsibility Pattern)

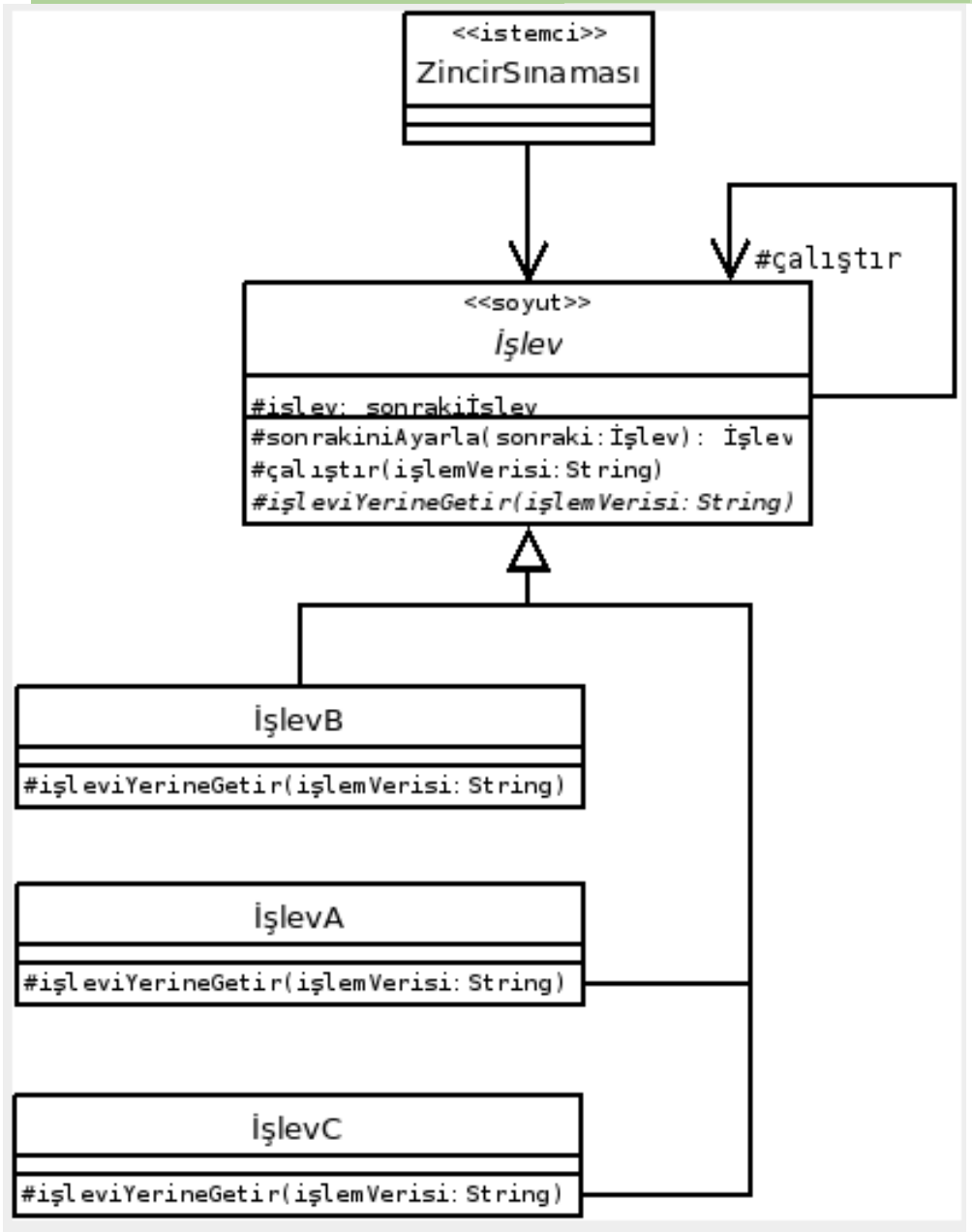
- **Handler:** Kendisinden türeyen ConcreteHandler' ların, talebi ele alması için gerekli arayüzü tanımlar. Abstract class veya Interface olarak tasarlanır.
- **ConcreteHandler :** Sorumlu olduğu talebi değerlendirir ve işler. Gerekirse talebi zincir içerisinde arkasından gelen nesneye iletir. Sonraki nesnenin ne olacağı genellikle istemci tarafında belirlenir.
- **Client :** Talebi veya mesajı gönderir.



https://www.buraksenyurt.com/post/Tasarc4b1m-Desenleri-Chain-of-Responsibility

Sorumluluk Zinciri Kalıbı (Chain of responsibility Pattern)

- Bir işlemin, belli miktarda işlevlerden sırayla geçmesi gerektiğinde bu tasarım deseni kullanılabilir.



http://www.tasarimdesenleri.com/jsp/tasdesincele/chainOfResponsibility.jsp

Sorumluluk Zinciri Kalıbı (Chain of responsibility Pattern)

İslev.java

```
public abstract class İşlev {

  protected İşlev  sonrakiİşlev;

  protected İşlev sonrakiniAyarla( final İşlev sonraki ) {
    sonrakiİşlev = sonraki;
    return sonraki;
  }

  protected void çalıştır( final String işlemVerisi ) {

    // İşlevi yerine getir.
    işleviYerineGetir( işlemVerisi );

    // Sonraki işlev boş değilse, ona geç.
    if ( sonrakiİşlev != null ) {
      sonrakiİşlev.çalıştır( işlemVerisi );
    }
  }

  protected abstract void işleviYerineGetir( String işlemVerisi );
}
```

http://www.tasarimdesenleri.com/jsp/tasdesincele/chainOfResponsibility.jsp

Sorumluluk Zinciri Kalıbı (Chain of responsibility Pattern)

İslevA.java

```
public class İşlevA extends İşlev {

  @Override
  protected void işleviYerineGetir( final String işlemVerisi ) {
    System.out.println( "İşlev A yapıldı. Veri: " + işlemVerisi );
  }
}
```

İslevB.java

```
public class İşlevB extends İşlev {

  @Override
  protected void işleviYerineGetir( final String işlemVerisi ) {
    System.out.println( "İşlev B yapıldı. Veri: " + işlemVerisi );
  }
}
```

http://www.tasarimdesenleri.com/jsp/tasdesincele/chainOfResponsibility.jsp

Sorumluluk Zinciri Kalıbı (Chain of responsibility Pattern)

İslevC.java

```
public class İşlevC extends İşlev {

  @Override
  protected void işleviYerineGetir( final String işlemVerisi ) {
    System.out.println( "İşlev C yapıldı. Veri: " + işlemVerisi );
  }
}
```

http://www.tasarimdesenleri.com/jsp/tasdesincele/chainOfResponsibility.jsp

Sorumluluk Zinciri Kalıbı (Chain of responsibility Pattern)

ZincirSınaması.java

```
public class ZincirSınaması {

  public static void main( final String[] args ) {

    // Zincir nesneleri.
    İşlev zincir1, zincir2, zincir;

    // Zincir 1 i oluştur. A -> C -> B
    zincir1 = new İşlevA();
    zincir = zincir1.sonrakiniAyarla( new İşlevC() );
    zincir = zincir.sonrakiniAyarla( new İşlevB() );

    // Zincir 2 yi oluştur. C -> A
    zincir2 = new İşlevC();
    zincir2.sonrakiniAyarla( new İşlevA() );

    // Zincir 1 i çalıştır.
    zincir1.çalıştır( "veri1" );

    // Zincir 2 yi çalıştır.
    zincir2.çalıştır( "veri2" );
  }
}
```

Ekran Çıktısı

```
İşlev A yapıldı. Veri: veri1
İşlev C yapıldı. Veri: veri1
İşlev B yapıldı. Veri: veri1
İşlev C yapıldı. Veri: veri2
İşlev A yapıldı. Veri: veri2
```

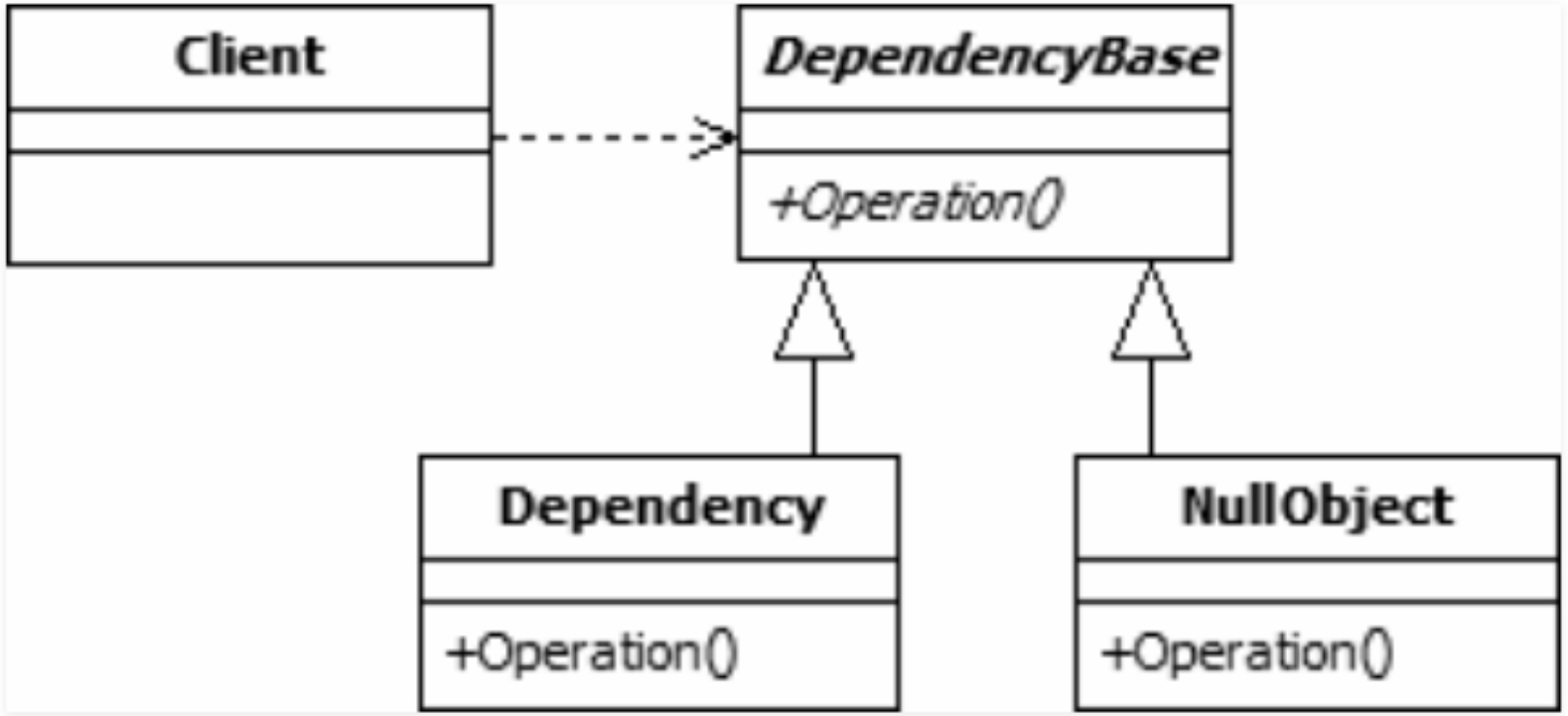
http://www.tasarimdesenleri.com/jsp/tasdesincele/chainOfResponsibility.jsp

Null Object Pattern

- NULL Object Pattern Gang of Four’s Design Patterns kitabında anlatılmış olup behavioral design pattern'ler den biridir.
- Bu pattern'in amacı uygulama içeresinde null objeler return etmek yerine ilgili tipin yerine geçen ve expected value'nun null objesi olarak kabul edilen tipi geriye dönmektir diğer bir değişle null yerine daha tutarlı nesneler dönmektir.
- . NULL Object Pattern , süreli olarak null kontrolü yaparak hem server-side hemde client-side için boilerplate code yazmaya engel olmak amacıyla ortaya çıkmış bir pattern dir.

http://www.canertosuner.com/post/null-object-pattern-nedir

Null Object Pattern



https://www.geeksforgeeks.org/null-object-design-pattern/

Null Object Pattern

- **İstemci :** Bu sınıfın gerekli olabilecek veya gerekmeyebilecek bir bağımlılığı vardır. Bağımlılıkta işlevsellik gerekmediğinde, boş bir nesnenin yöntemlerini çalıştıracaktır.
- **DependencyBase:** Bu soyut sınıf, İstemcinin kullanabileceği çeşitli mevcut bağımlılıklar için temel sınıftır. Bu aynı zamanda boş nesne sınıfının temel sınıfıdır. Temel sınıfın paylaşılan bir işlevsellik sağlamadığı durumlarda, bir arabirim ile değiştirilebilir.
- **Bağımlılık :** Bu sınıf, İstemci tarafından kullanılabilecek işlevsel bir bağımlılıktır.
- **NullObject :** Bu, İstemci tarafından bağımlılık olarak kullanılabilecek boş nesne sınıfıdır. Hiçbir işlevsellik içermez, ancak DependencyBase soyut sınıfı tarafından tanımlanan tüm üyeleri uygular.

https://www.geeksforgeeks.org/null-object-design-pattern/

Null Object Pattern

```
abstract class Emp
{
    protected String name;
    public abstract boolean isNull();
    public abstract String getName();
}
```

```
class Coder extends Emp
{
    public Coder(String name)
    {
        this.name = name;
    }
    @Override
    public String getName()
    {
        return name;
    }
    @Override
    public boolean isNull()
    {
        return false;
    }
}
```

https://www.geeksforgeeks.org/null-object-design-pattern/

Null Object Pattern

```
class NoClient extends Emp
{
    @Override
    public String getName()
    {
        return "Not Available";
    }

    @Override
    public boolean isNull()
    {
        return true;
    }
}
```

```
class EmpData
{

    public static final String[] names = {"Lokesh", "Kushagra", "Vikram"};
    public static Emp getClient(String name)
    {
        for (int i = 0; i < names.length; i++)
        {
            if (names[i].equalsIgnoreCase(name))
            {
                return new Coder(name);
            }
        }
        return new NoClient();
    }
}
```

https://www.geeksforgeeks.org/null-object-design-pattern/

Null Object Pattern

```
public class Main
{
    public static void main(String[] args)
    {
        Emp emp1 = EmpData.getClient("Lokesh");
        Emp emp2 = EmpData.getClient("Kushagra");
        Emp emp3 = EmpData.getClient("Vikram");
        Emp emp4 = EmpData.getClient("Rishabh");


        System.out.println(emp1.getName());
        System.out.println(emp2.getName());
        System.out.println(emp3.getName());
        System.out.println(emp4.getName());
    }
}
```

https://www.geeksforgeeks.org/null-object-design-pattern/

Anti Patterns

- AntiPatterns terimi **1995' de Andrew Koenig'** in **Journal of Object Oriented Programming**(ki doğrulatamadığım bilgilere göre bunun yerini Journal of Object Technology almıştır)' de yayınlanan **C++** köşesindeki **Patterns and AntiPatterns** makalesinde şu şekilde tanımlanmıştır;
- AntiPattern is just like pattern, except that instead of solution it gives something that looks superficially like a solution, but isn't one. (Koenig, 1995)
- *Şöyle Yorumlayabiliriz :* ***AntiPattern görünüşte(yüzeysel anlamda) çözüm zannedilen bir Pattern gibidir, ama aslında değildir.***

https://www.buraksenyurt.com/post/AntiPatterns-Ders-Notlarc4b1m

Anti Patterns

- **AntiPattern'** ler yazılım geliştirme de kötü çözüm yaklaşımları ve pratikleri olarak da bilinirler.
- Tasarım desenleri, belli başlı problemlerin çözümünde standart pratikleri önerirken, **AntiPattern**' ler tam tersine arzu edilmeyen ve sonrasında daha büyük problemlerin kapısını açan pratiklerin uygulanması olarak düşünülebilir.
- **AntiPattern'** lerin tehlikeli tarafı ürün geliştirme süreçlerinde ve vakalarda en uygun çözüm yolu olarak düşünülmeleridir.

https://www.buraksenyurt.com/post/AntiPatterns-Ders-Notlarc4b1m

Anti Patterns

- **Sihirli Düğme (Magic PushButton)**
- **Spaghetti Kodlama (Spaghetti Coding)**
- **Lazanya Kodlama (Lasagna Coding)**
- **Kopyala -Yapıştır Programlama (Copy-Paste Programming)**
- **Tanrısal Nesne (God Object)**
- **Altın Çekiç (Golden Hammer)**
- **Tekerleği Yeniden Keşfetme Eğilimi (Reinventing the Square Wheel)**
- **Aşırı Mühendislik (Over Engineering)**

https://medium.com/aykiri-yazilimcilar/kaliteli-yaz%C4%B1l%C4%B1m-tasar%C4%B1m%C4%B1-ve-anti-patternler-%C3%BCzerine-notlar-a8f9ccfb6847

Anti Patterns

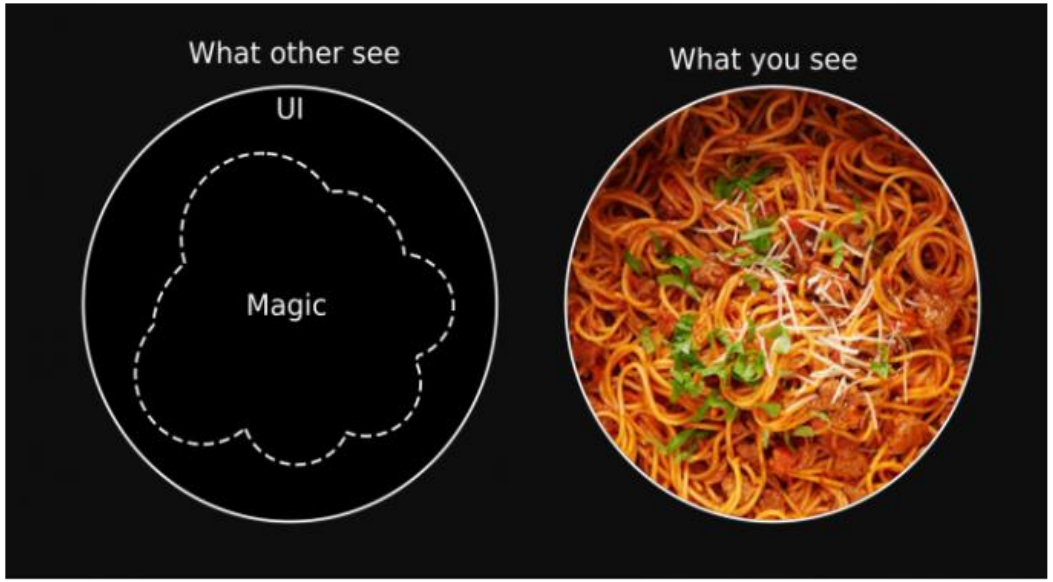
Sihirli Düğme (Magic PushButton)

Herhangi bir soyutlama yapılmaksızın, görsel bileşenlerin (olaylarının / events) arkasında tüm kodlamanın yapılmasıdır. Bu yaklaşım *"buton-click"* programcılığı olarak da adlandırılmaktadır. Özellikle GUI (Graphical User Interface) tabanlı uygulamalarda daha fazla görülür (Örneğin, Windows Forms Programlama). Özetle; ara yüz tarafı ile iş mantıkları genellikle buton gibi bir bileşenlerin arkasına gömülür.

https://medium.com/aykiri-yazilimcilar/kaliteli-yaz%C4%B1l%C4%B1m-tasar%C4%B1m%C4%B1-ve-anti-patternler-%C3%BCzerine-notlar-a8f9ccfb6847

Anti Patterns

Spaghetti Kodlama (Spaghetti Coding)
Bakım ve değişiklik yapılamayacak kadar karmaşık yazılmış kodlama türüne bu ad verilmektedir. Nesne yönelimli olmayan dillerde daha sık rastlanan bir durumdur. Metotlar bir sürecin tamamını gerçekleştirmek için yazılırlar. Çoğu metot parametre almaz ya da tam aksine çok fazla parametre alır. Kodun yeniden kullanılabilirliği zordur. Nesne Yönelimli Programlama (NYP) kullanılıyorsa bile NYP temel özellikleri (kalıtım, çok biçimlilik, soyutlama) doğru kullanılmamıştır.



https://medium.com/aykiri-yazilimcilar/kaliteli-yaz%C4%B1l%C4%B1m-tasar%C4%B1m%C4%B1-ve-anti-patternler-%C3%BCzerine-notlar-a8f9ccfb6847

Anti Patterns

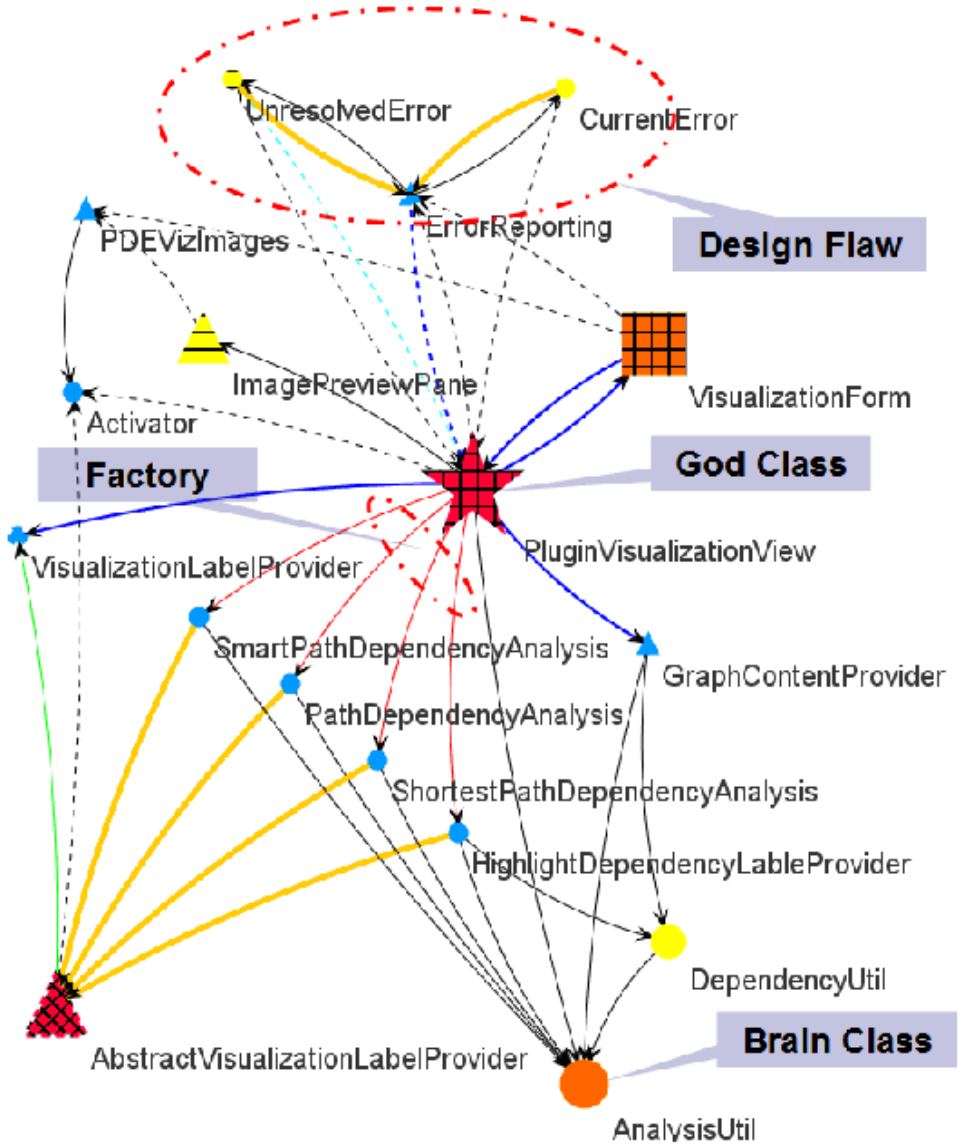
Lazanya Kodlama (Lasagna Coding)
Gereğinden fazla sayıda katmana sahip uygulama geliştirilmesine verilen isimdir (Aşırı çok katmanlı uygulama). Çok katman ve çok sayıda irili ufaklı sınıf kullanılarak tasarlanmış bir yazılım uygulamasının anlaşılması ve değiştirilmesi zordur.

https://medium.com/aykiri-yazilimcilar/kaliteli-yaz%C4%B1l%C4%B1m-tasar%C4%B1m%C4%B1-ve-anti-patternler-%C3%BCzerine-notlar-a8f9ccfb6847

Anti Patterns

Tanrısal Nesne (God Object)

Gereğinden fazla iş yapabilen sınıflara bu isim verilmektedir. Bu sınıflar çok fazla üyeye ve davranışa sahip olup, uygulamanın ana sınıfı gibi algılanabilmektedirler. Bu kötü bir tasarım yaklaşımıdır ve (neredeyse) tüm iş mantığını barındıran bu sınıflarda değişiklik yapmak imkansızdır. Yazılım karmaşıklığına ek olarak, bu nesnelerin belleğe yüklenmesi zaman alır.



https://medium.com/aykiri-yazilimcilar/kaliteli-yaz%C4%B1l%C4%B1m-tasar%C4%B1m%C4%B1-ve-anti-patternler-%C3%BCzerine-notlar-a8f9ccfb6847

Anti Patterns

Altın Çekiç (Golden Hammer)

Daha önceden bildiğimiz bir tasarım yaklaşımının mükemmel bir çözüm olduğuna inanıp, her sorunu aynı yöntemle çözmeye çalışmaktır.

“Elinde çekiç olan birine tüm sorunlar çivi gibi görünür.”

Farklı problemler farklı çözüm yaklaşımları gerektirebilirler. Fakat işin içerisinde sahiplenme söz konusu olduğu için bu durumu aşmak kolay değildir (alışkanlıklar ve egoyu bir kenara bırakmak zor). Örneğin tüm yazılım problemlerinde aynı mimari desenini kullanmaya çalışmak yanlış bir yaklaşımdır.

https://medium.com/aykiri-yazilimcilar/kaliteli-yaz%C4%B1l%C4%B1m-tasar%C4%B1m%C4%B1-ve-anti-patternler-%C3%BCzerine-notlar-a8f9ccfb6847

Anti Patterns

Tekerleği Yeniden Keşfetme Eğilimi (Reinventing the Square Wheel)

Günlük hayatta hemen her problemin algoritmasının bilindiği bir dünyada, bu çözümleri bilerek ya da (bilmeyerek) görmezden gelerek, yeni çözümler üretme çabasına verilen isimdir. Bu yaklaşımın arkasındaki niyet iyi olsa da sonuçları genellikle hüsrandır. Örneğin veritabanına erişim katmanında kullanabileceğimiz onlarca ORM (Object Relational Mapping) aracı/çatısı varken, kendinizin sıfırdan böyle bir çatıyı yazmaya kalkışmanız mantıklı değildir (2010 yılında olsak neyse).

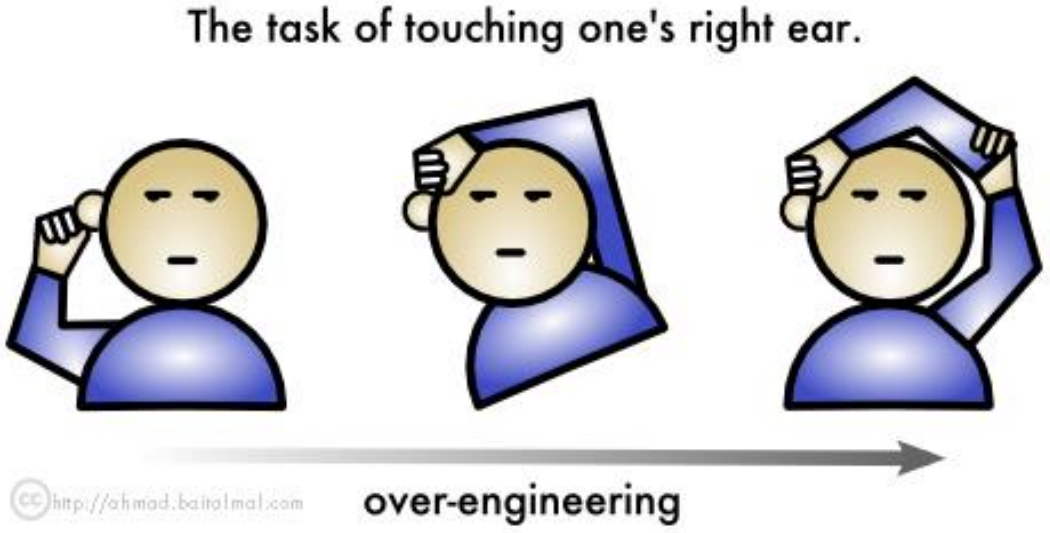
https://medium.com/aykiri-yazilimcilar/kaliteli-yaz%C4%B1l%C4%B1m-tasar%C4%B1m%C4%B1-ve-anti-patternler-%C3%BCzerine-notlar-a8f9ccfb6847

Anti Patterns

Aşırı Mühendislik (Over Engineering)

Özellikle proje yönetimi kategorisinde incelenen bir anti-pattern türüdür. Bir ürünün ihtiyaç olmadığı halde gerekenden daha sağlam ve daha fazla özelliğe sahip olacak şekilde tasarlanması ve haliyle gerekenden daha fazla kaynak harcanması şeklinde tanımlanabilir. Basitlik ilkesi ve startup ürün geliştirme sürecinde kullanılan MVP (Minimum Viable Product) yaklaşımları ile tamamen çelişmektedir.

MVP: Piyasa sürülmek istenilen bir ürünün en önemli ve etkili (pazarlanabilir) temel özelliklerinin, en az maliyetle (zaman, kaynak) yapılması ve mümkün olduğunca ortaya yalın bir ürün çıkartılmasıdır.



https://medium.com/aykiri-yazilimcilar/kaliteli-yaz%C4%B1l%C4%B1m-tasar%C4%B1m%C4%B1-ve-anti-patternler-%C3%BCzerine-notlar-a8f9ccfb6847

Kaynaklar

- http://www.tasarimdesenleri.com/jsp/tasdesincele/chainOfResponsibility.jsp
- https://bidb.itu.edu.tr/seyir-defteri/blog/2013/09/08/davran%C4%B1%C5%9Fsal-kal%C4%B1plar-(behavioral-patterns)
- https://www.buraksenyurt.com/post/Tasarc4b1m-Desenleri-Chain-of-Responsibility
- https://www.geeksforgeeks.org/null-object-design-pattern/
- https://www.buraksenyurt.com/post/AntiPatterns-Ders-Notlarc4b1m
- https://medium.com/aykiri-yazilimcilar/kaliteli-yaz%C4%B1l%C4%B1m-tasar%C4%B1m%C4%B1-ve-anti-patternler-%C3%BCzerine-notlar-a8f9ccfb6847
- https://nevraa.com/2012/06/22/anti-patternbuyuk-sorunla/