

Nesne Yönelimli Analiz ve Tasarım

Nesne Tasarımı

Tasarım Desenleri
Design Patterns

Tasarım Desenleri

- * Yazılımlar geliştirilirken karşılaşılan genel tasarım problemlerini tanımlayarak, bu problemin en uygun nasıl çözülebileceğini (kod tekrar kullanımını artırmak ve değişikliği kolaylaştırmak için) ve çözümün ortaya çıkaracağı sonuçları anlatır.
- * Programlama dillerinden bağımsızdır.
- * Yazılım geliştiricilerin tasarımlarla ilgili tartışma yapmasını kolaylaştırır.
- * Tasarım desenleri dört bölümden oluşur
 - * Desenin adı
 - * problemin tanımı: desenin ne zaman/herede kullanılabileceği
 - * çözüm: problemin nasıl çözüleceği (kullanılması gereken bileşenler, aralarındaki bağıntı vb.)
 - * sonuç: deseni uygulamanın sonuçları?

Tasarım Desenleri

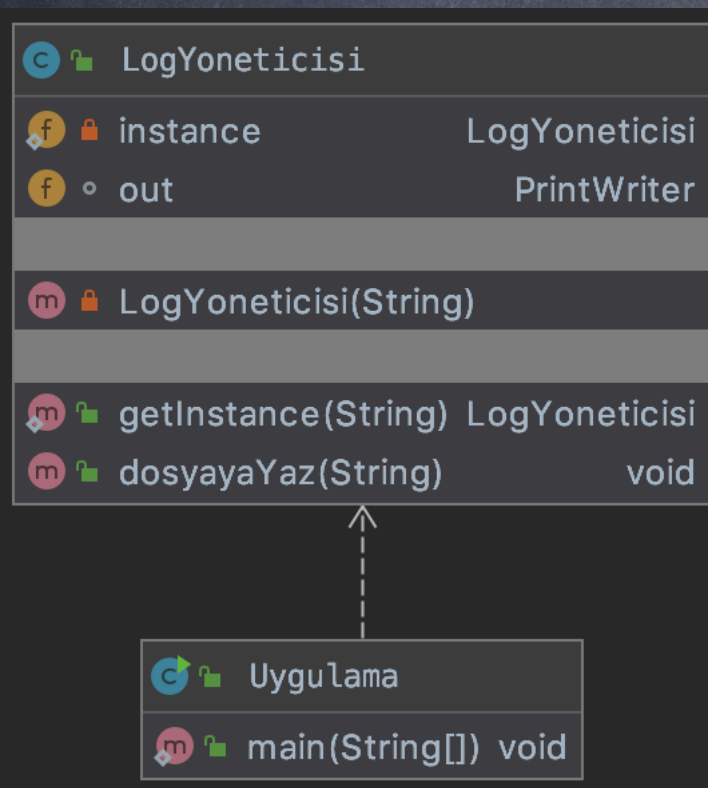
- * Creational(Nesne oluşturma): Nesnelerin uygun bir şekilde oluşturulmasını sağlayacak mekanizmalar içerir.
- * Structural(Yapısal): Nesnelerin sistemler içerisinde uygun olarak yerleştirilmesini sağlayacak desenlerdir.
- * Behavioral(Davranışsal): Nesnelerin birbirleriyle uygun olarak etkileşiminini düzenleyen mekanizmalar.

Creational (Nesne oluşturma)	Structural (Yapısal)	Behavioral (Davranışsal)
Singleton Pattern	Adapter Pattern	Template Method Pattern
Factory Pattern	Composite Pattern	Mediator Pattern
Abstract Factory Pattern	Proxy Pattern	Chain of Responsibility Pattern
Builder Pattern	Flyweight Pattern	Observer Pattern
Prototype Pattern	Facade Pattern	Strategy Pattern
	Bridge Pattern	Command Pattern
	Decorator Pattern	State Pattern
		Visitor Pattern
		Interpreter Pattern
		Iterator Pattern
		Memento Pattern

Tasarım Desenleri : Singleton

- * Amaç: Bir sınıfın yalnızca tek nesnesi olmasını ve bu nesneye global olarak erişilmesini sağlamak.
- * Nesne yoksa oluşturulur döndürülür, varsa olan döndürülür.
- * Nesnenin new komutu ile dışarıdan oluşturulabilmesini engellemek için yapıcı yöntem "private" yapılır.
- * Yapıcı gibi geçen "static" bir yöntem tanımlanır. Nesne oluşturma görevi bu yöntemindir. Oluşturulan nesne "static" bir üyede saklanır.

Tasarım Desenleri : Singleton



```
public class LogYoneticisi {

    private static LogYoneticisi instance;
    PrintWriter out;

    private LogYoneticisi(String logDosyasi){
        try {
            out = new PrintWriter(new FileWriter(logDosyasi,true), true);
        } catch (IOException e) {e.printStackTrace();}
    }

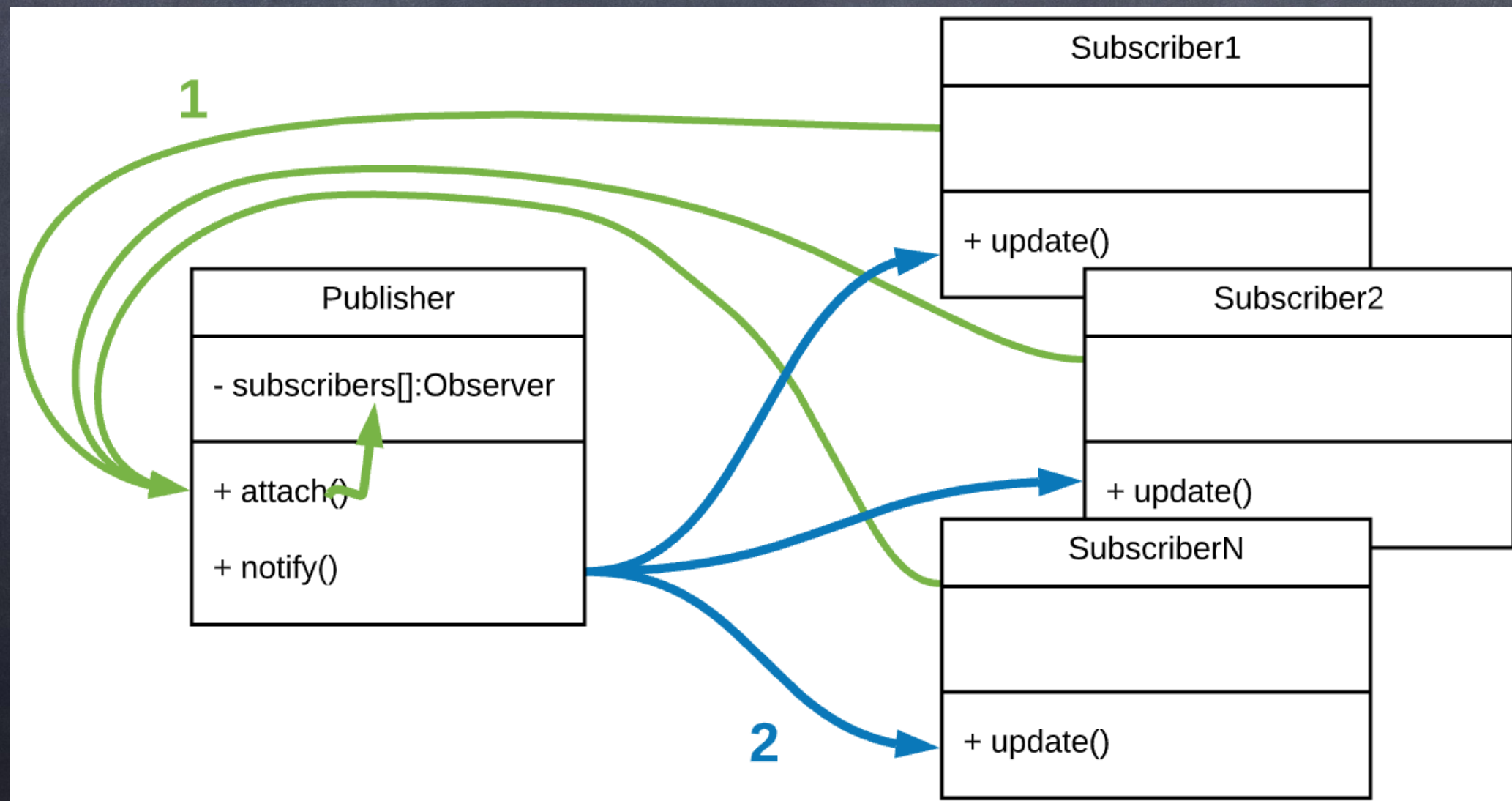
    public static synchronized LogYoneticisi getInstance(String logDosyasi){
        if(instance==null)
            instance = new LogYoneticisi(logDosyasi);
        return instance;
    }

    public void dosyayaYaz(String mesaj) {
        out.println(LocalDate.now()+":"+mesaj);
    }
}

public class Uygulama {
    public static void main(String [] args){
        LogYoneticisi.getInstance("Log.txt").dosyayaYaz("[WARNING]:uyari mesajı 1");
    }
}
```


Tasarım Desenleri : Observer

- * Amaç: Çok sayıda nesneye, gözlemledikleri nesnede meydana gelen olayı bildirmek.
- * Böyle bir etkileşim Publish(Yayın)-Subscribe(abone) olarak da adlandırılır.



Tasarım Desenleri : Observer

