Nesneye Yönelik Yazılım Mühendisliği (376)

Yrd. Doç. Dr. Ahmet Arif AYDIN

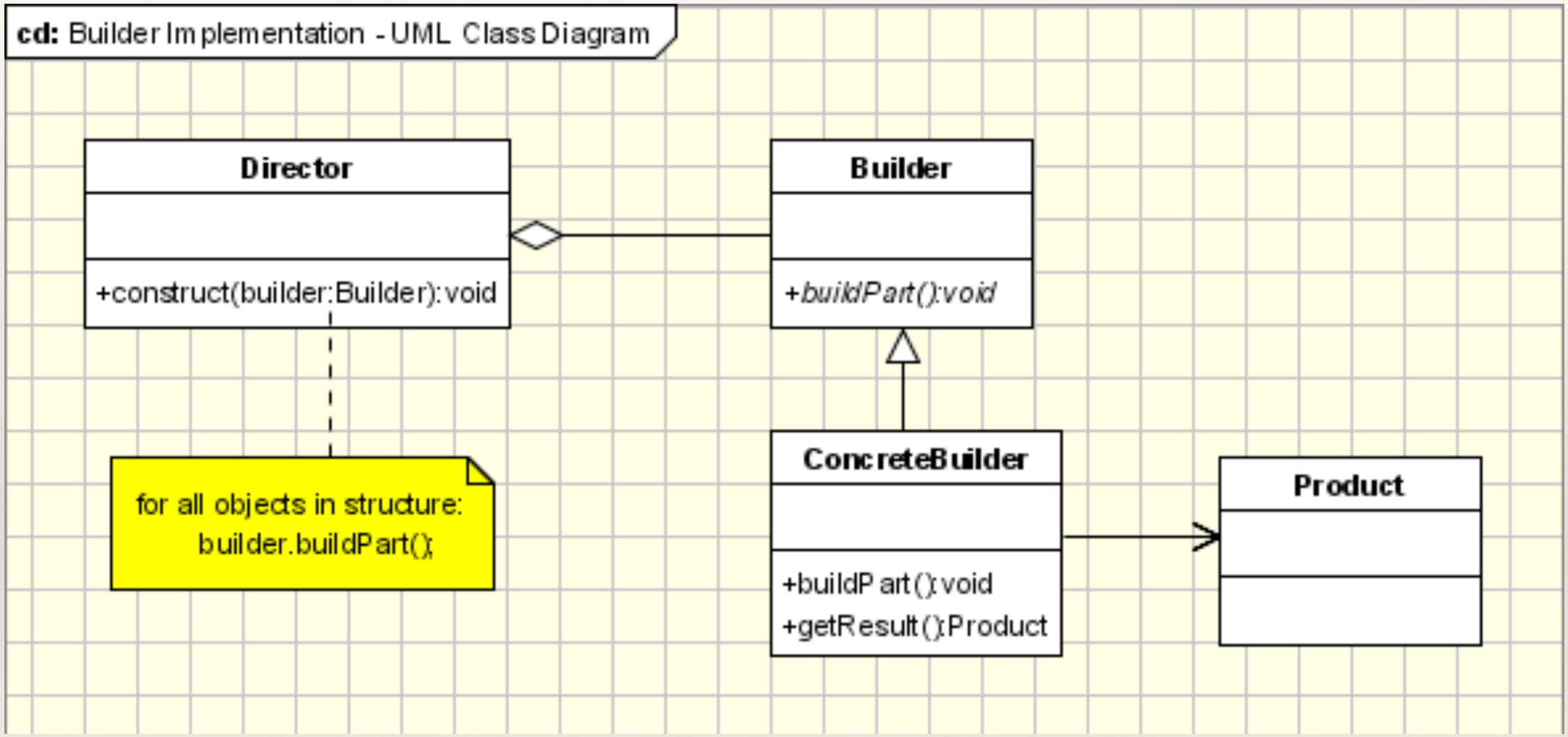
Creational Design Patterns : Builder

- Bir nesnenin bir birinden farklı özellikleri içeren ürünlerini alt sınıflar olarak tanımlanan somut oluşturucu (concrete builder) lar kullanılarak oluşturulmasını saglayan yazılım kalıbına builder denir. (*Separates object construction from its representation*)
- Karmaşık nesnenin oluşturulması birbirinden farklı parçaların bir araya gelmesiyle oluşturulmalıdır. *(The algorithm for creating a complex object should be independent of the parts that make up the object and how they're assembled.)*
- Oluşturma surecinde özellikleri biribirinden farklı ürünlerin oluşturulmasını sağlamalıdır *(The construction process must allow different representations for the object that's constructed)*

Creational Design Patterns : Builder

- Unlike creational patterns that construct products in one shot, the Builder pattern constructs the product step by step under the control of the "director
- Builder pattern aims to fix problems of Factory and Abstract Factory design patterns when the Object contains a lot of attributes.
- There can be more than one such builder classes, each with different implementations for the series of steps to construct the object. Each builder implementation results in a different representation of the object

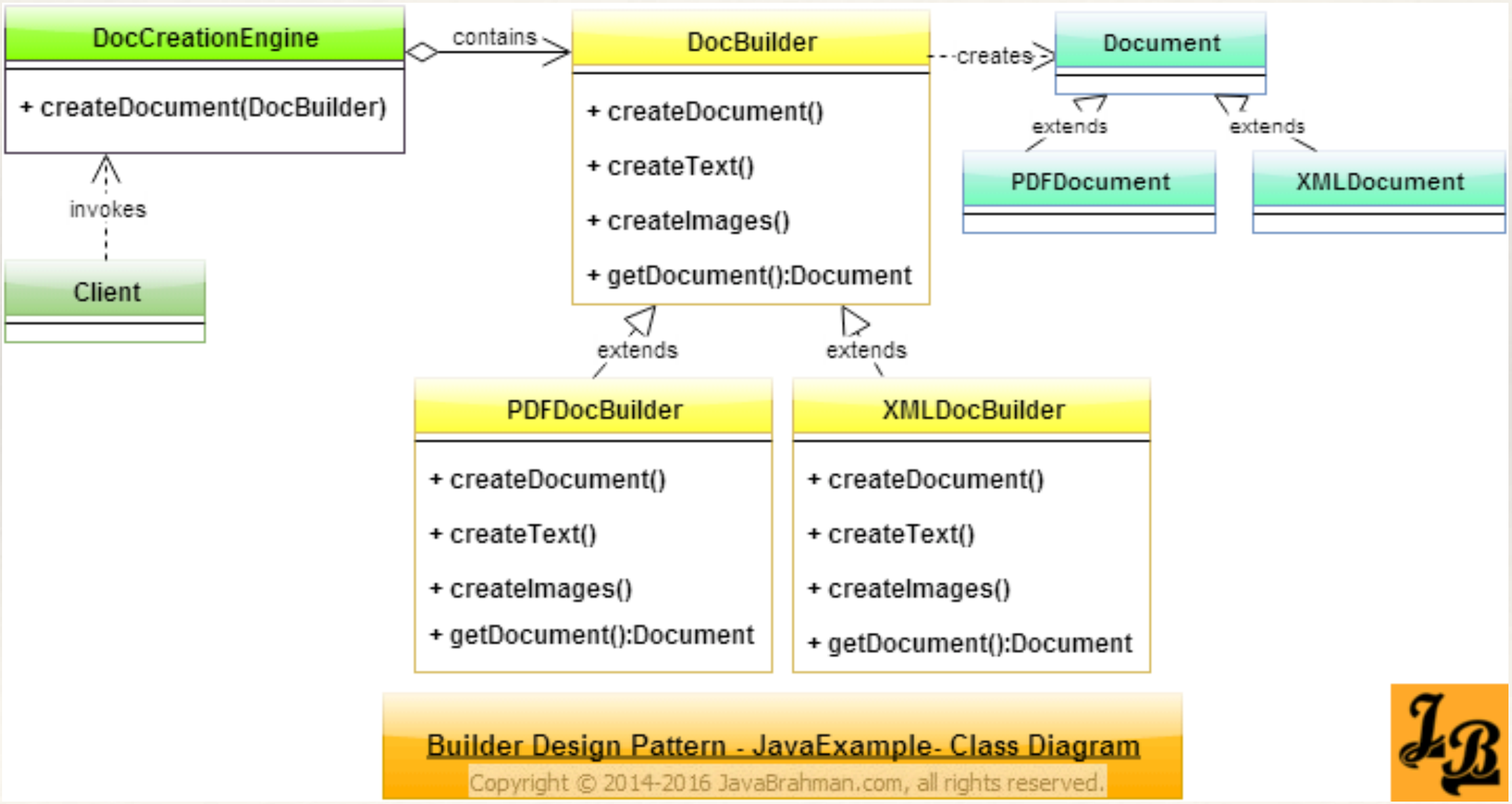
Creational Design Patterns : Builder



http://www.oodesign.com/

Creational Design Patterns : Builder - Örnekler

1. Bir metin belgesini **farklı formatlarda** kayıt işlemini gerçekleştirirken kullanılabilir. (pdf, rtf, doc, docx, jpg , png)

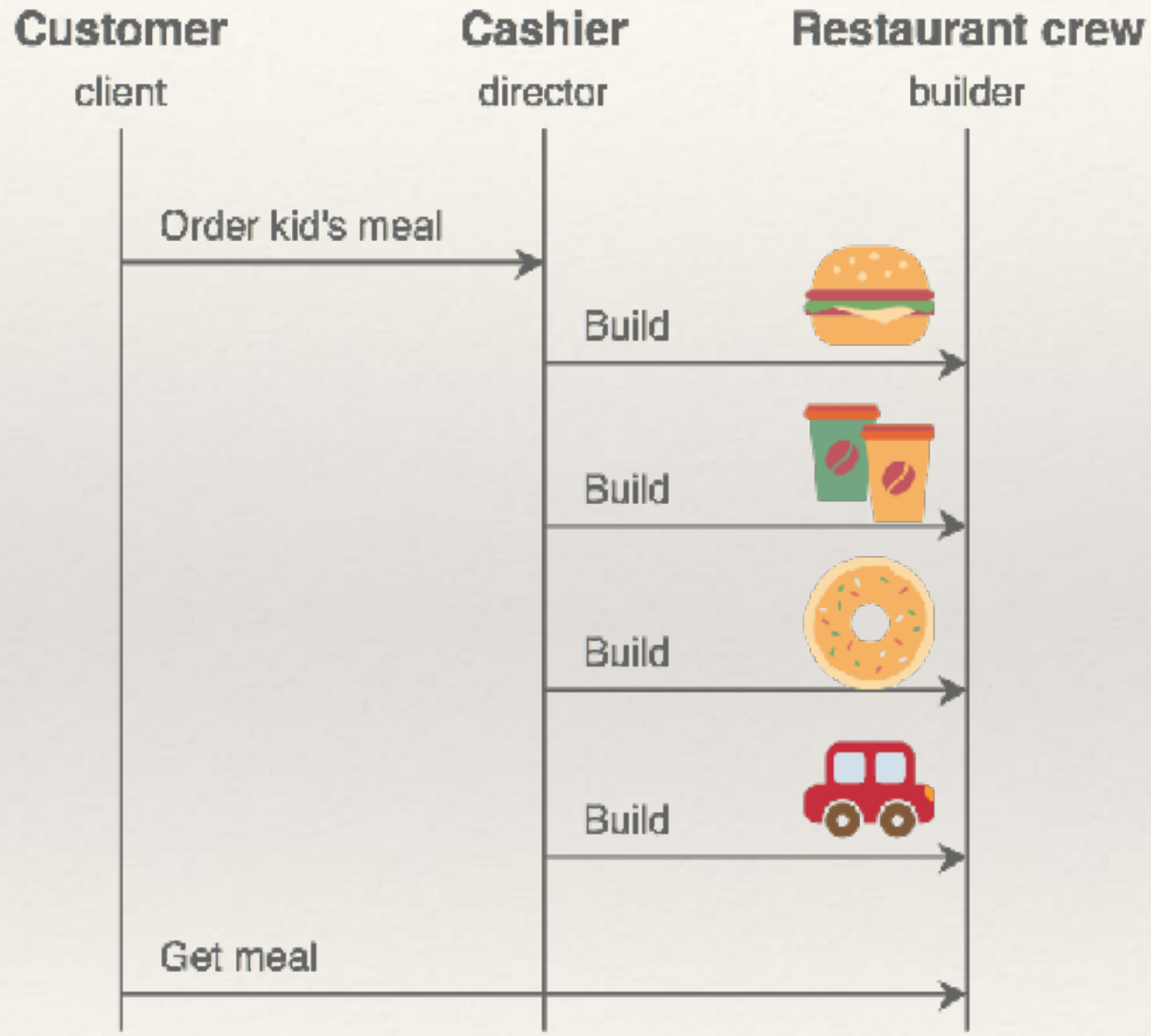


Creational Design Patterns : Builder - Örnekler

2. Bir araç firması kullanıcılarına istedikleri araçların **grafiksel modelini** ekrana çizdiren
    - Araç nesnesi istenilen özellikleri içermelidir
    - Çizim nesneside istenilen özelliklere göre ekrana istenilen modeli çizecektir.
    - Sedan ve Spor araç sınıflarının her biri için özelliklerini gerçekleştirecek builder (oluşturucu) oluşturulur.

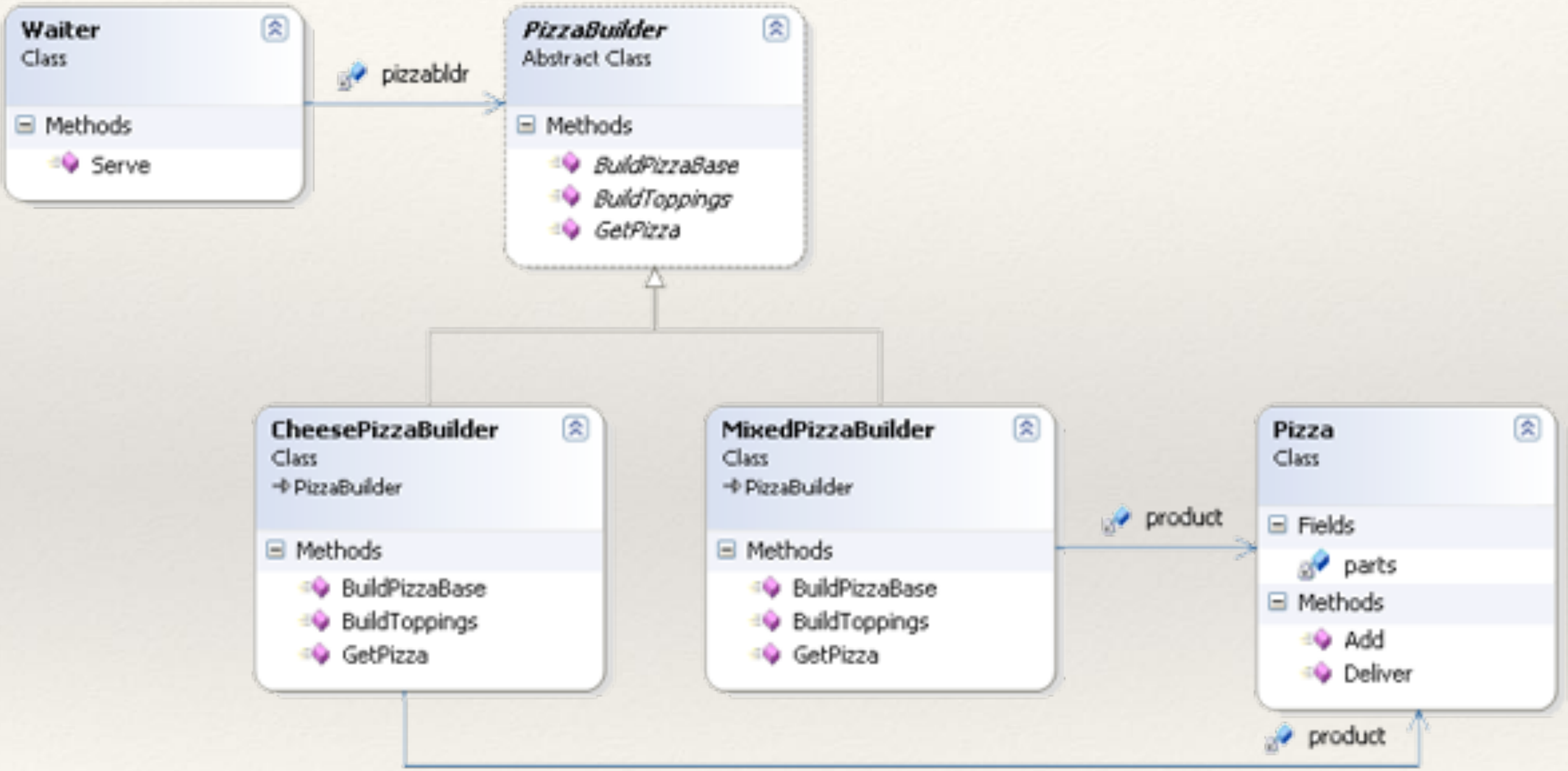
Creational Design Patterns : Builder - Örnekler

3. Yemek firmaları istenilen **menüyü (ürün)** hazırlaması



https://sourcemaking.com/design_patterns/builder

Creational Design Patterns : Builder

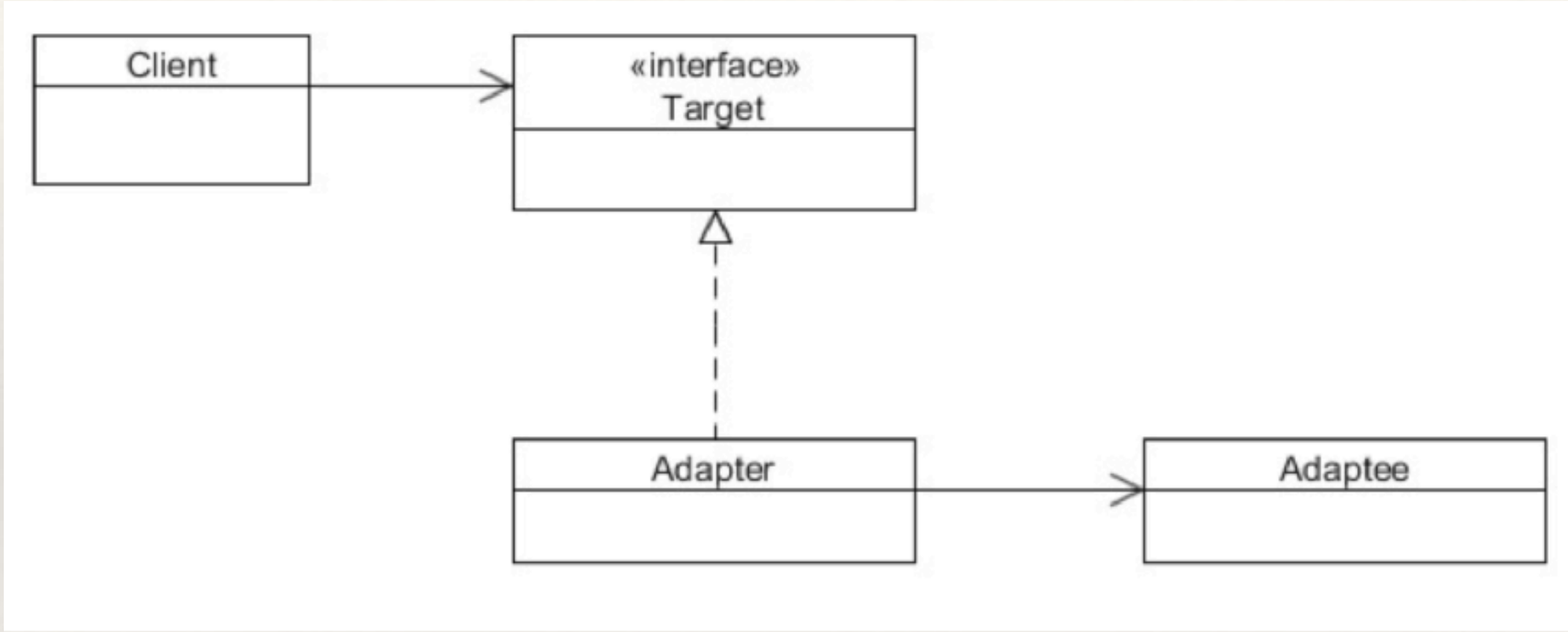


https://www.go4expert.com/articles/design-pattern-simple-examples-t5127/

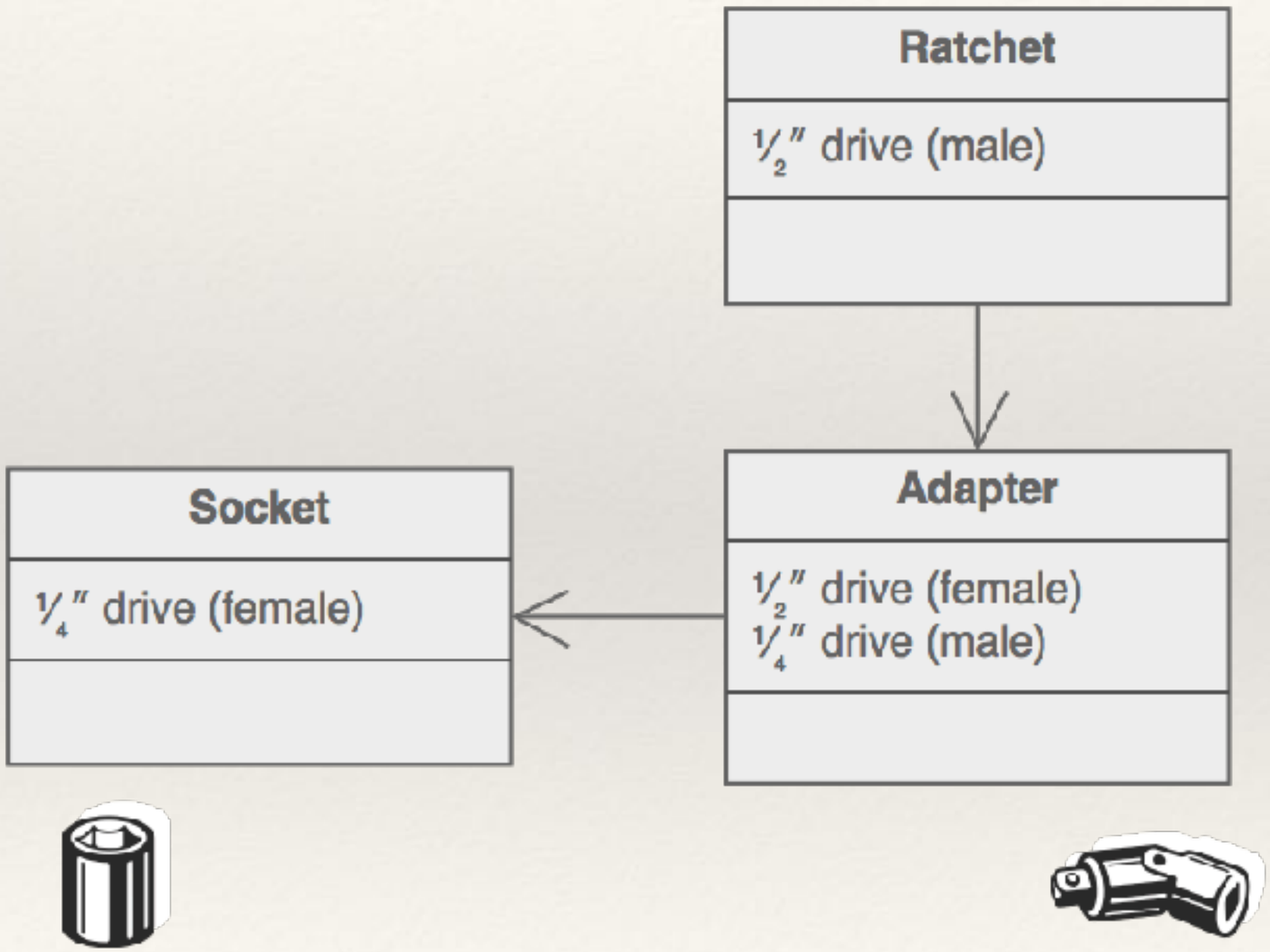
Structural Design Patterns: Adaptor

- Bir birine uyumlu olmayan iki sistemin entegre edilerek beraber çalışmasını sağlayan tasarım kalıbı Adaptor 'dür. (incompatible interfaces of the two objects which do not fit together can be used with an adaptor)
    - integrating a legacy code with a new code
    - changing a 3rd party API in the code
    - The Adapter pattern lets you to adapt what an object or a class exposes to what another object or class expects
    - **reusability**

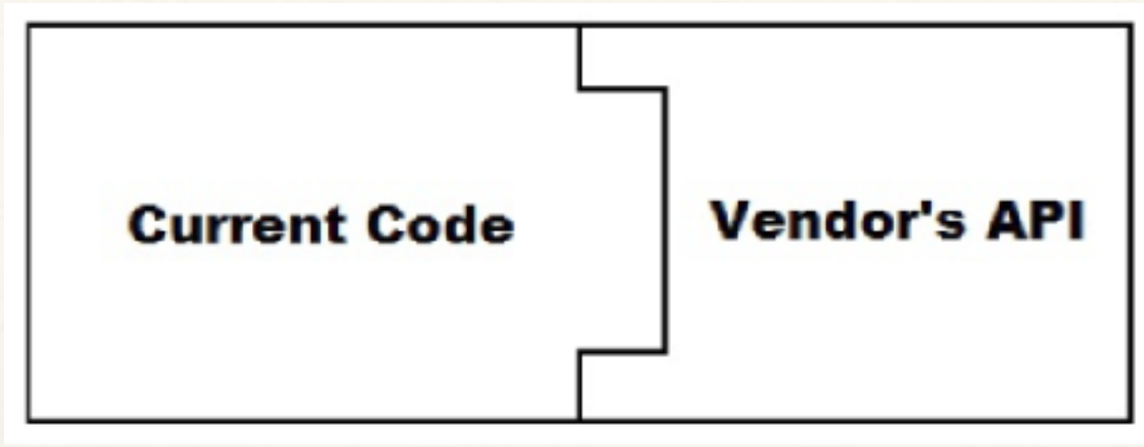
Structural Design Patterns: Adaptor

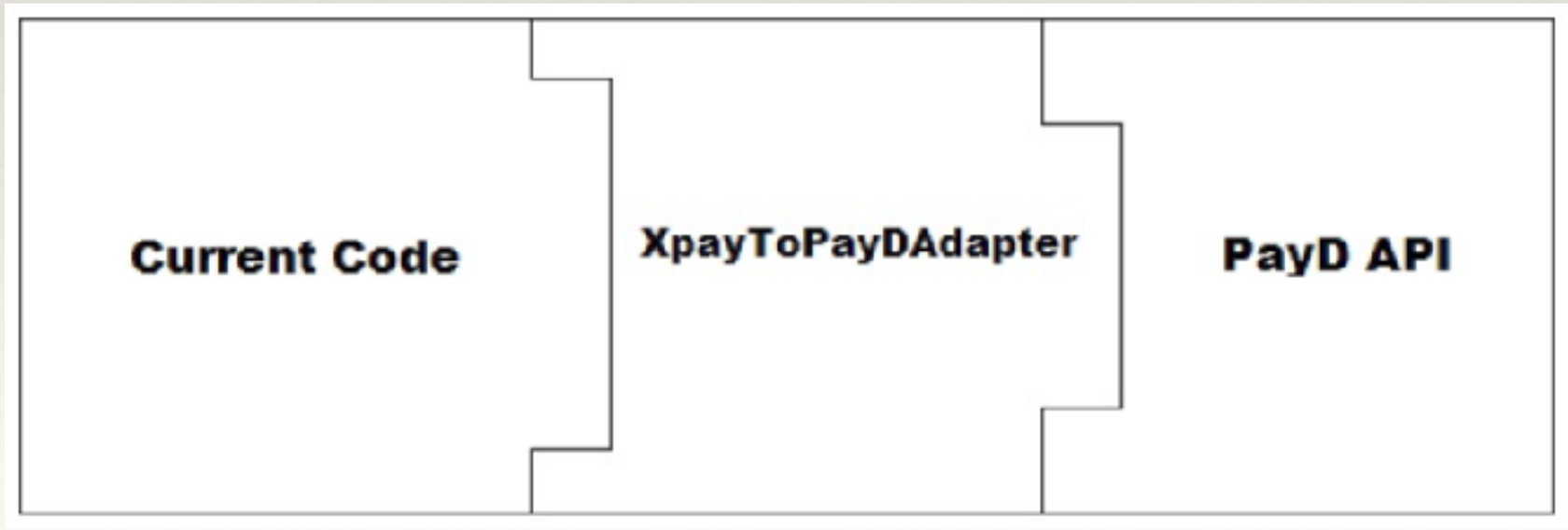


Structural Design Patterns: Adaptor



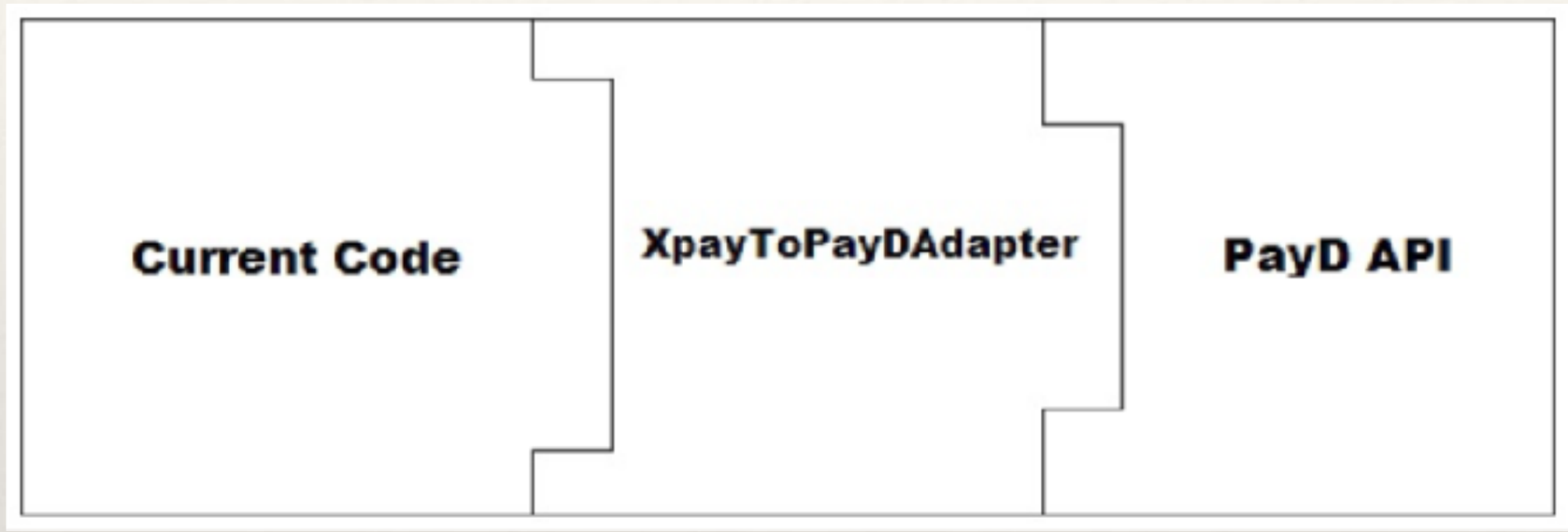
Structural Design Patterns: Adaptor





Structural Design Patterns: Adaptor

```
public interface PayD
```




```
public class XpayToPayDAdapter
implements PayD
```

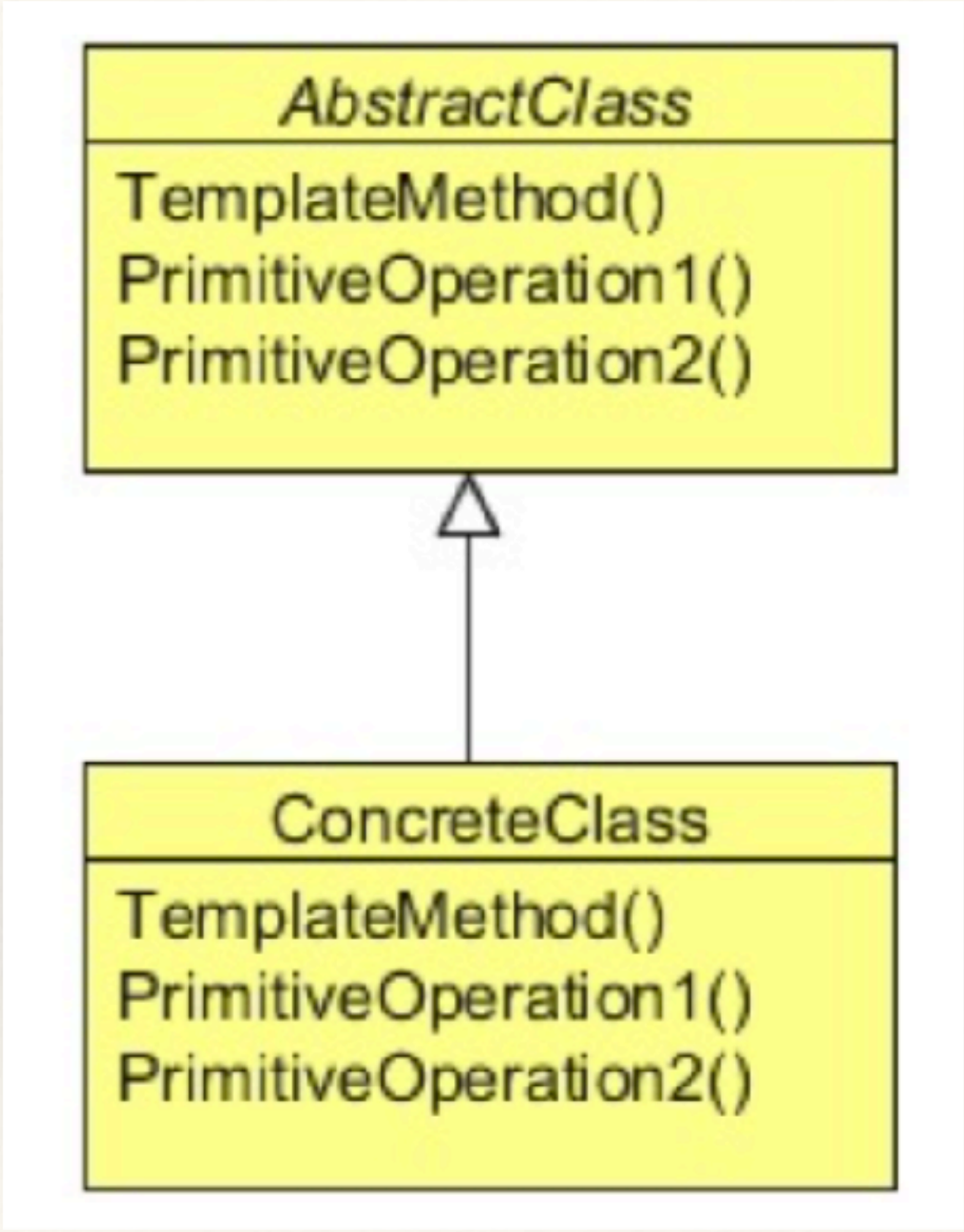
Structural Design Patterns: Adaptor

Behavioral Design Patterns: Template Method

- Aşamaları belirli olan bir algoritmanın tanımlanıp kullanıcı isteklerine ihtiyaçlarına göre farklı kodlanılması imkanın sağlayan tasarım kalıbı Template'dır.

  - provides a template or a structure of an algorithm which is used by users.
  - A user provides its own implementation without changing the algorithm's structure

Behavioral Design Patterns: Template Method



Behavioral Design Patterns: Template Method

```java
public abstract class ConnectionTemplate {

    private boolean isLoggingEnable = true;

    public ConnectionTemplate(){
        isLoggingEnable = disableLogging();
    }

    public final void run(){

        setDBDriver();
        logging("Drivers set ["+new Date()+"]");
        setCredentials();
        logging("Credentails set ["+new Date()+"]");
        connect();
        logging("Conencted");
        prepareStatement();
        logging("Statement prepared ["+new Date()+"]");
        setData();
        logging("Data set ["+new Date()+"]");
        insert();
        logging("Inserted ["+new Date()+"]");
        close();
        logging("Conenctions closed ["+new Date()+"]");
        destroy();
        logging("Object destoryed ["+new Date()+"]");

    }

    public abstract void setDBDriver();

    public abstract void setCredentials();

    public void connect(){
        System.out.println("Setting connection...");
    }

    public void prepareStatement(){
        System.out.println("Preparing insert statement...");
    }

    public abstract void setData();

    public void insert(){
        System.out.println("Inserting data...");
    }

    public void close(){
        System.out.println("Closing connections...");
    }

    public void destroy(){
        System.out.println("Destroying connection objects...");
    }

    public boolean disableLogging(){
        return true;
    }

    private void logging(String msg){
        if(isLoggingEnable){
            System.out.println("Logging....: "+msg);
        }
    }
}
```

Behavioral Design Patterns: Template Method

```
public class MySqLCSVCon extends ConnectionTemplate{

	@Override
	public void setDBDriver() {
		System.out.println("Setting MySQL DB drivers...");
	}

	@Override
	public void setCredentials() {
		System.out.println("Setting credentials for MySQL DB...");
	}

	@Override
	public void setData() {
		System.out.println("Setting up data from csv file....");
	}

	@Override
	public boolean disableLogging() {
		return false;
	}

}
```

Behavioral Design Patterns: Template Method

```
public class OracleTxtCon extends ConnectionTemplate{

        @Override
        public void setDBDriver() {
                System.out.println("Setting Oracle DB drivers...");
        }

        @Override
        public void setCredentials() {
                System.out.println("Setting credentials for Oracle DB...");
        }

        @Override
        public void setData() {
                System.out.println("Setting up data from txt file....");
        }


}
```

Behavioral Design Patterns: Template Method