Design Patterns(Tasarım Kalıpları)

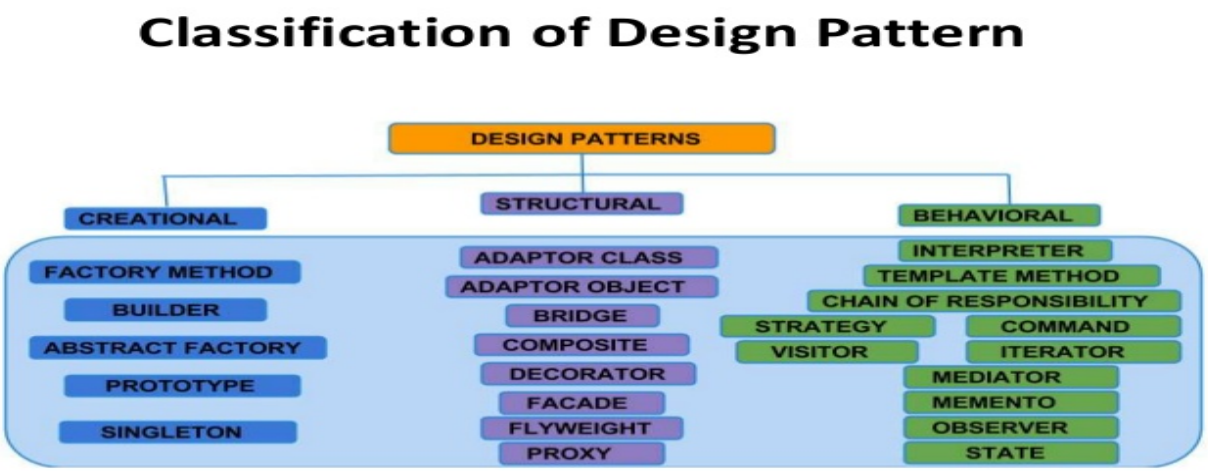
Yazılım geliştiricilerin yazılım geliştirme sırasında karşılaştıkları genel sorunların çözümüdür. GOF, kitaplarında da 23 adet Design Patterns'i konu almıştır. Ancak günümüzde bundan çok daha fazlası vardır. Yazılım tasarım kalıpları genel olarak 3 ana başlıkta incelenir. Bunlar şunlardır:

1- **Creational Patterns (Yaratımsal Kalıplar):** Bu tasarım deseni nesneleri doğrudan new operatörü kullanarak oluşturmak yerine nesne oluşturma mantığını gizleyerek sınıflardan nesne oluşturmaya alternatif çözümler sunar. Bu program akışında hangi nesneye ihtiyaç varsa onu oluşturmada esneklik ve kolaylık sağlar.

2- **Structural Patterns (Yapısal Kalıplar):** Bu tasarım deseni nesneler arasındaki ilişkinin yapısını düzenlemek için çözümler sunar.

3- **Behavioral Patterns (Davranışsal Kalıplar):** Bu tasarım deseni çalışma zamanında nesneler arasındaki davranışlar için çözümler sunar.

Bu 3 ana başlık altında yer alan ,yazılımcılar arasında bilinen popüler tasarım kalıpları vardır.



Şimdi Creational pattern lar içindeki Factory design patterni bir örnekle ile açıklayalım.

---Öncelikle bir Advert interface i oluşturalım.

```
public interface Advert {
    String getAdvertType();
       String getTitle();
        int getPrice();
        int getTime();
}
```

---Daha sonra bir Advert tipi olan CarAdvert ve HouseAdvert classlarını oluşturalım ve bunları Advert dan implemente edelim.Çünkü temelinde araba ilanları da ,ev ilanları da birer ilandır.Constructorlarını ve getter-setter metodlarını da oluşturalım.

```
public class CarAdvert  implements Advert{
       private String advertType;
       private String title;
       private int price;
       private int time;
       public CarAdvert() {

       }
       public CarAdvert(String advertType, String title, int price, int time) {
              super();
              this.advertType = advertType;
              this.title = title;
              this.price = price;
              this.time = time;
       }
       public String getAdvertType() {
              return advertType;
       }
       public void setAdvertType(String advertType) {
```

```
		this.advertType = advertType;
	}
	public String getTitle() {
		return title;
	}
	public void setTitle(String title) {
		this.title = title;
	}
	public int getPrice() {
		return price;
	}
	public void setPrice(int price) {
		this.price = price;
	}
	public int getTime() {
		return time;
	}
	public void setTime(int time) {
		this.time = time;
	}
}
public class HouseAdvert implements Advert{


	private String advertType;
	private String title;
	private int price;
	private int time;

	public HouseAdvert() {
```

```
}
public HouseAdvert(String advertType, String title, int price, int time) {
	super();
	this.advertType = advertType;
	this.title = title;
	this.price = price;
	this.time = time;
}
public String getAdvertType() {
	return advertType;
}
public void setAdvertType(String advertType) {
	this.advertType = advertType;
}
public String getTitle() {
	return title;
}
public void setTitle(String title) {
	this.title = title;
}
public int getPrice() {
	return price;
}
public void setPrice(int price) {
	this.price = price;
}
public int getTime() {
	return time;
}
public void setTime(int time) {
	this.time = time;
```

```
	}
}
```

--- Daha sonra new leme işlemini ,yeni oluşuracağımız AdvertFactory classı üstlensin diyoruz ve classı oluşturuyoruz.Gönderilen advertType ına göre nesne oluşturucak burada, eğer sistemde olmayan bir advertType gönderirse hata mesajı dönücek.

```
public class AdvertFactory {
public static Advert getAdvert(String advertType,String title,int price,int time) {
Advert advert;
if("CarAdvert".equalsIgnoreCase(advertType)) {
advert=new CarAdvert(advertType,title,price,time);
}else if("HouseAdvert".equalsIgnoreCase(advertType)) {
advert=new HouseAdvert(advertType,title,price,time);
}else {
throw new RuntimeException("Geçerli bir ilan değildir");
}
return advert;
}
}
```

---Son olarak veri girişi yapacağımız Main classımızı oluşturalım.Burada nesne oluşturmamıza gerek kalmadı çünkü AdvertFactory classımızdaki getAdvert metodumuzu kullanıyoruz ve nesne orada oluşturuluyor.

```
public class Main {
public static void main(String[] args) {
Advert CarAdvert=AdvertFactory.getAdvert( "CarAdvert", "Acill", 50000, 10);
Advert HouseAdvert=AdvertFactory.getAdvert( "HouseAdvert", "Acill satılık", 3000000,
20);
System.out.println(CarAdvert);
System.out.println(HouseAdvert);
	}
}
```