Creational Design Patternler yeni bir nesne oluşturacağımız zaman kullandığımız design patternlerdir. Yeni nesne oluşturma işlemi örneği aşağıda verilmiştir.

Person person = new Person( );

Bazen new ile nesne oluşturmak istemeyebiliriz, uygulamada nesnelerin oluşturulmasını isteyebiliriz. Bu gibi durumlarda Types of Creational Design Patterns kullanılır. Kendi içerisinde altıya ayrılmıştır.

1. Factory Method Design Pattern

   Burada nesneler interface veya abstract class tarafından oluşturulur. Nesnenin türü subclasslardan biri olmalı ve iletilen girdiye bağlı olmalıdır. Factory Method Pattern'in en büyük avantajı girdide esneklik sağlamasıdır.

   Bir interface in veya abstract class programın sürekli değişmesi beklendiğinde, önceden yazılan program değişikliğe açık olmadığında Factory Method Pattern in kullanılması beklenir.

2. Abstract Factory Design Pattern

   Burada nesne oluşturmak için subclass belirtmeden sadece interface veya abstract class tanımlar. Avantajı, nesneler arası tutarlılık sağlamasıdır.

3. Singleton Design Pattern

   Tek nesnenin oluşturulması istendiğinde ve soyutlamanın istenmediği durumlarda tercih edilen bir patterndir. Singleton Patternde static kelimesi kullanılır.

private static Person person = new Person( );

```
public class Singleton  {
    private Singleton() {}

    private static class SingletonHolder {
        public static final Singleton instance = new Singleton();
    }

    public static Singleton getInstance() {
        return SingletonHolder.instance;
    }
}
```

Figure1 [2]

4. Prototype Design Pattern

Burada yeni nesne oluşturmak tercih edilmiyor, onun yerine önceden oluşturulan bir nesneyi klonlama ya da o nesneyi yeni gereksinimlere göre özelleştirme seçenekleri tercih ediliyor. Yeni nesne oluşturma işleminin maliyetli olduğu durumlarda Prototype tercih edilir. Avantajı, subclasslara duyulan ihtiyaç azalır ve nesne oluşturmanın karmaşıklığını arka plana gizler. Diğer sayfada alıntı bir UML örneği verilmiştir.

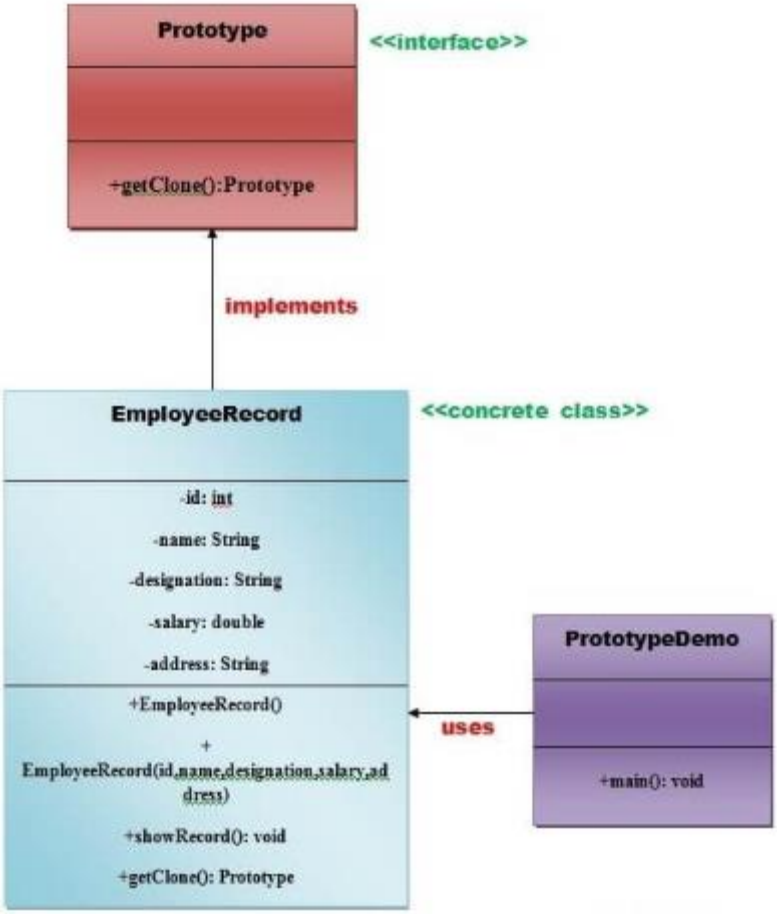

Figure 2 [1]

5. Builder Design Pattern

Çok karmaşık bir nesnenin sürekli kullanılabilir bir hale getirilmesinde ve nesne tek adımda oluşturulamadığında Builder Pattern tercih edilir. Avantajı; nesne oluşturma süresince daha iyi kontrol sağlamasıdır.

6. Object Pool Design Pattern

Bir sınıfı başlatma maliyetinin çok yüksek miktarda paralar gerektirdiği durumlar kullanılır.

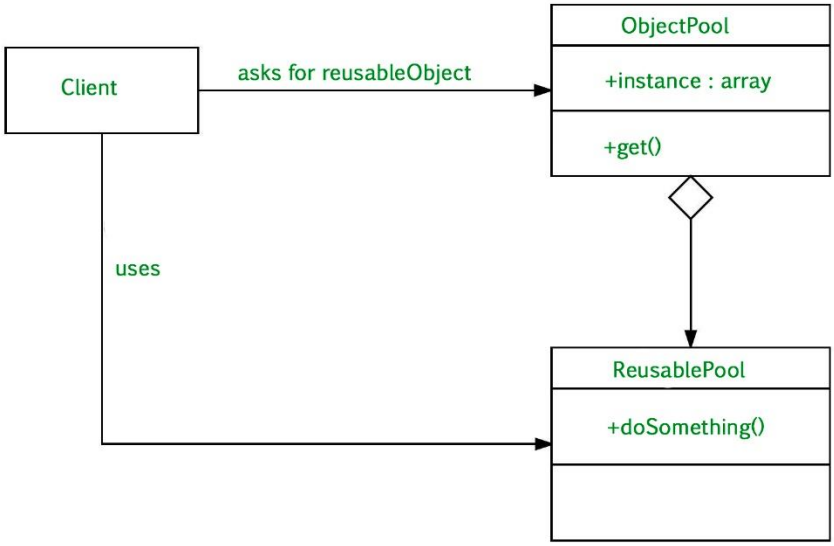

Figure 3 [3]

Kaynakça:

1- https://www.javatpoint.com/prototype-design-pattern
2- https://www.baeldung.com/creational-design-patterns
3- https://www.geeksforgeeks.org/object-pool-design-pattern/