Creational - Singleton Design Pattern

- Bu tasarım örüntüsündeki amaç, bir class'tan sadece bir instance yaratılmasını sağlar. Yani herhangi bir class'tan bir instance yaratılmak istendiğinde, eğer daha önce yaratılmış bir instance yoksa yeni yaratılır. Daha önce yaratılmış ise var olan instance kullanılır.
- **Singleton pattern logging, caching ve thread havuzları içinde kullanılırSingleton design pattern, Abstract Factory,Builder,Prototype ve Facade desenlerinin içerisinde de kullanılır**
- Diyelim ki bir üyelik sistemi projeniz var ve bu üyelik sistemi içerisinde kullanıcı bir hareket gerçekleştirdi. Bu gerçekleşen hareketin tarihini, saatini vs. kaydetmeniz gerekiyor. İşte bu durumda sistem kullanıcıya özel bir veri değil standart bir veri üretmesi gerekir. Bu durumda da Singleton Design Pattern uygulanarak bu ihtiyaç karşılanabilir. Singleton Design Pattern uygulanacağı zaman kullanıcıya ya da parametreye bağlı olarak çıktı üretilmeyeceği yani herkes için standart bir akış uygulanacak demektir.

Desing pattern neden kullanılmalıdır?

- Herkes bu nesneyi kullanıyormu?
- Yada aynı nesneyi herkes kullanacak mı?
- Çok nadir kullanılan bir nesne ise singleton olarak tasarlanmamalıdır

```
class Singleton{
   private volatile static Singleton singleton;
   private Singleton(){} //private constructor
   public static Singleton getInstance(){
      //synchronized her defasında çalışmasın diye objenin create edilip edilmediğini kontrol ediyoruz
      //çünkü synchronized pahalı bir işlem
      if (singleton == null){
         synchronized (Singleton.class){
            if (singleton==null){
               singleton = new Singleton();
            }
         }
      }
      return singleton;
   }
   void printTestMessage(){
      System.out.println("Singleton olarak çalıştı");
   }
}
```

Main

```
public static void main(String[] args) {
   Singleton s1 = Singleton.getInstance();
   s1.printTestMessage();
}
```