# Singleton

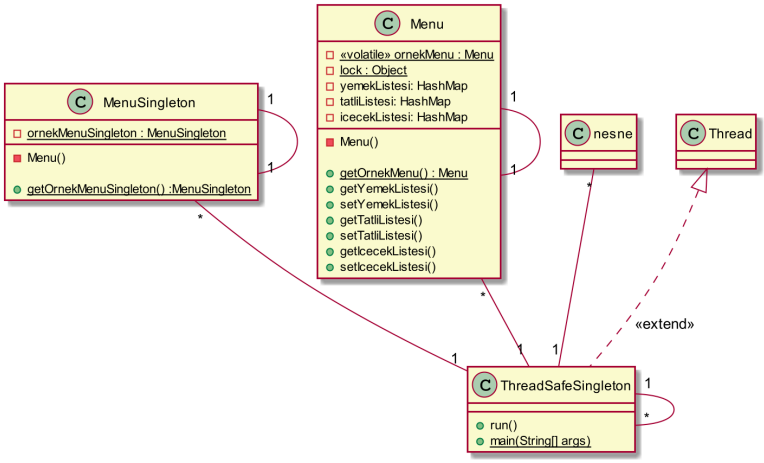


**Figure 1. Singleton and Thread Safe Singleton**

## 1. Singleton

```
İlk üç madde singleton yapısının taşıması gereken şartları içermektedir.
 private static nesne referansı. Private constructor ve dışardan erişimi
 sağlamak için public static getInstance methodu. Singletonda amaç ramde
 yer gereksiz yer kaplamayı engellemek adına tekrar tekrar üretilmesine
 gerek olmayan nesnelerden 1 tane üretilerek çağrıldığında aynı nesneyi
 döndürmektir.
```

```
public class MenuSingleton {

    private static MenuSingleton ornekMenuSingleton =null; ❶

    private MenuSingleton(){} ❷

    public static MenuSingleton getOrnekMenuSingleton() {  ❸

        if (ornekMenuSingleton == null) { ❹
                ornekMenuSingleton = new MenuSingleton();
        }
        return ornekMenuSingleton;
```

```
    }
```

❶ private static tipinde MenuSingleton Tipinde nesne refere edilir singletonun şartlarından biridir.

❷ private tipte constructor üretilir singleton yapısının şartlarından biridi yeni nesne oluşturulmasını engellemek için.

❸ public static tipinde dışardan erişilebilerek aynı nesneyi döndürecek fonksiyon oluşturulur singletonun şartlarındandır.

❹ daha önce bir nesne oluşturulmuşmu kontrol edilir oluşturulduysa o nesne döndürülür oluşturulmadıysa bir nesne oluşturulup o nesne döndürülür.

## 2. Thread Safe Singleton

```
Singleton threadlerle çalışıldığında nesnenin daha önce üretilip
 üretilmediğinin kontro edildiği kısımda threadlerle çalışıldığı için
 birden fazla nesne üretebilme sıkıntısı gösterebilmektedir. Bunun için
 singleton yapısının thread safe haline getirilmesi gerekir.
```

```
public class Menu {

    private static volatile Menu ornekMenu; ❶
    private static Object lock = new Object(); ❷
    private Menu(){} ❸

     public static Menu getOrnekMenu() { ❹
        Menu result = ornekMenu; ❺
        if (result == null) {
            synchronized (lock) { ❻
                result = ornekMenu;
                if (result == null) ❼
                    ornekMenu = result = new Menu();
            }
        }
        return result; ❽
    }
```

❶ private static volatile tipinde Menu Tipinde nesne refere edilir singletonun şartlarından biridir. Volatile program dışında etki altında bulunabilmesi için kullanılır.

❷ thread safe olayının sağlanması için synchronizeda parametre olarak verilir daha sonra 6. maddede anlatılacak.

❸ private tipte constructor üretilir singleton yapısının şartlarından biridi yeni nesne oluşturulmasını engellemek için.

❹ public static tipinde dışardan erişilebilerek aynı nesneyi döndürecek fonksiyon oluşturulur singletonun şartlarındandır.

❺ volatile etkisinden kurtarmak için local bir değişkene atama yapar

❻ thread safe özelliğinin sağlanması için javada synchronized özelliği kulanılır ve bu sayede belirlenen bir obje üzerinden threadlerin aynı anda çalışarak istenmeyen sonuçlar oluşturmasını engeller.

❼ daha önce nesne oluşturuldumu kontrol eder oluşturulmadıysa yeni nesne oluşturur.

❽ daha önce oluşturulan nesneyi yada daha önce nesne oluşturulmadıysa yeni oluşturulan nesneyi döndürür.