Yazılım Tasarımı ve Mimarisi Dersi

Singleton (Tek şey) Deseni Deney Föyü

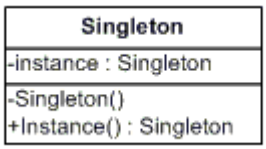
Hazırlık Soruları:

1. Statik sınıf/metod nedir, neden kullanılır?
2. Bir nesnenin birden fazla üretilmemesi neden istenir, ne gibi bir faydası olur?
3. "Early Instantiation" ve "Lazy Initialization" kavramları nedir, örnek vererek açıklayınız?

Giriş

Amacı bir nesnenin program içerisinde birden fazla üretilmemesini sağlamaktır. Dolayısıyla bir nesnenin yalnızca bir defa üretilip, bu üretilen nesnenin diğer nesnelerden ulaşılabilir olması gerektiği durumlarda kullanılır. Bu sayede gereksiz yere sürekli nesne üretilip kaynak tüketimi yapılmaz. Aynı kaynağa erişimin sıralı olması sağlanır. (Printer havuzu gibi) Daha çok loglama, thread havuzu ve konfigürasyon ayarlama gibi yerlerde kullanılır.

UML



Singleton

Sınıfın tek nesnesi üzerinden erişilecek işlemi tanımlar. Yalnızca tek bir nesnesin üretilmesini sağlar ve bu durumu korur.

Yapısal Kod:

```
using System;
using System.Threading.Tasks;

namespace ConsoleApplication10
{
    class Program
    {
        static void Main(string[] args)
        {
            #region Singleton 1
            //SINGLETON: 1. Tip
```

```
            Singleton1 sO11 = Singleton1.Nesne1();
            Singleton1 sO12 = Singleton1.Nesne1();

            if (sO11 == sO12)
            {
                Console.WriteLine("Durum 1: Nesneler aynidir. (If yapisi ve metod
kullanildi. Single Thread.)");
            }

            Console.ReadKey();
            #endregion

            #region Singleton 2
            //SINGLETON: 2. Tip
            Singleton2 sO21 = Singleton2.Nesne2;
            Singleton2 sO22 = Singleton2.Nesne2;
            if (sO21 == sO22)
            {
                Console.WriteLine("Durum 2: Nesneler aynidir.(Get kullanildi, If
yapisi yok. Single Thread)");
            }

            Console.ReadKey();
            #endregion

            #region Singleton 3
            //SINGLETON: 3. Tip
            Singleton3 sO31 = Singleton3.Nesne3();
            Singleton3 sO32 = Singleton3.Nesne3();
            if (sO31 == sO32)
            {
                Console.WriteLine("Durum 3: Nesneler aynidir.(Metod kullanildi, If
yapisi var. Multi Thread)");
            }

            Console.ReadKey();
            #endregion

        }
    }
    class Singleton1
    {
        private static Singleton1 nesne1;

        private Singleton1()
        {
        }

        public static Singleton1 Nesne1()
        {

            if (nesne1 == null)
            {
                nesne1 = new Singleton1();
            }

            return nesne1;
        }
    }

    class Singleton2
    {
```

```
        private static Singleton2 nesne2 = new Singleton2();

        private Singleton2()
        {
        }

        public static Singleton2 Nesne2
        {

            get
            {
                return nesne2;
            }


        }
    }

    public class Singleton3
    {
        private static Singleton3 nesne3;

        private static Object kilit = new Object();

        private Singleton3()
        {
        }

        public static Singleton3 Nesne3()
        {
            if (nesne3 == null)
            {
                lock (kilit)
                {
                    if (nesne3 == null)
                    {
                        nesne3 = new Singleton3();
                    }
                }
            }

            return nesne3;
        }
    }
}
```

Soru 1: Yukarıdaki yapısal kod örneğinde 2. durum örneğinde nesneye “get” kullanılarak erişilmişti. Bu örneği “get” ile erişmek yerine, metod kullanarak erişecek şekilde yeniden düzenleyiniz.

Örnek Uygulama:

```
using System;
using System.Collections.Generic;

namespace ConsoleApplication11
{
```

```
class Program
{
    static void Main(string[] args)
    {
        YukDengeleyici yd1 = YukDengeleyici.YukDengeleyiciyiGetir();
        YukDengeleyici yd2 = YukDengeleyici.YukDengeleyiciyiGetir();
        YukDengeleyici yd3 = YukDengeleyici.YukDengeleyiciyiGetir();
        YukDengeleyici yd4 = YukDengeleyici.YukDengeleyiciyiGetir();

        // Nesne kontrol
        if (yd1 == yd2 && yd2 == yd3 && yd3 == yd4)
        {
            Console.WriteLine("Nesnelerin ayni oldugu goruluyor.\n");
        }

        // Yuk dengelemek icin 15 sunucu istegi
        YukDengeleyici dengeleyici = YukDengeleyici.YukDengeleyiciyiGetir();
        for (int i = 0; i < 15; i++)
        {
            string server = dengeleyici.Sunucu;
            Console.WriteLine("Su sunucuya istek gonderildi: " + server);
        }

        Console.ReadKey();
    }
}

class YukDengeleyici
{
    private static YukDengeleyici nesne;
    private List<string> sunucular = new List<string>();
    private Random rnd = new Random();

    // Lock islemi icin gerekli
    private static object kilit = new object();


    protected YukDengeleyici()
    {
        sunucular.Add("Sunucu_I");
        sunucular.Add("Sunucu_II");
        sunucular.Add("Sunucu_III");
        sunucular.Add("Sunucu_IV");
        sunucular.Add("Sunucu_V");
    }

    public static YukDengeleyici YukDengeleyiciyiGetir()
    {
        // Multi thread
        if (nesne == null)
        {
            lock (kilit)
            {
                if (nesne == null)
                {
                    nesne = new YukDengeleyici();
                }
            }
        }

        return nesne;
    }
```

```
        // Yuk dengelemek icin rastgele sunucu acar.
        public string Sunucu
        {
            get
            {
                int r = rnd.Next(sunucular.Count);
                return sunucular[r].ToString();
            }
        }
    }
}
```