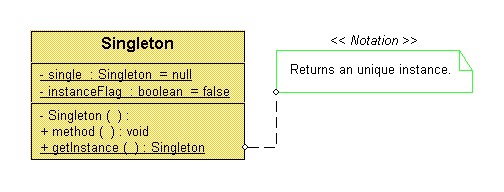
Singleton

* Дава възможност да се създаде само една единствена инстанция на класа.
* Класът дефинира метод, който да връща инстанцията (напр: getInstance()). Ако в момента на извикване все още нямаме създаде инстанция, тя се създава и се връща на потребителя. Ако е създадена, връща се вече създадената.

Има няколко особености:

* Контролира достъпа до единствената инстанция. Всеки път при вземане на инстанция се връща единствената.
* Подобряване на прозиводителността – отпада необходимостта от създаване и унищожаване на обекти всеки път при вземане на инстанция.

Singleton много прилича статичен клас и затова при решението си се ръководим от следните разсъждения: ако имаме набор от функции, които трябва да бъдат държани заедно, тогава избираме static клас. Всичко останало, което се нуждае от достъп до ресурси, е желателно да бъде имплементирано като сингълтон.



Код:

using System;

using System.Collections.Generic;

using System.Text;

using log4net;

using log4net.Config;

public class Singleton

{

// Reference to a sole instance.

private static Singleton instance = null;

// Атрибут на инстанцията.

private int data = 0;

/\*\*

\* The construvtor must be private, so that no client can

\* istantiate the Singleton object.

\*/

private Singleton()

{

}

/\*\*

\* Gives a reference to one sole instance.

\*/

public static Singleton getInstance()

{

if (instance == null)

instance = new Singleton();

return instance;

}

public int getData()

{

return data;

}

public void setData(int data) {

this.data = data;

}

public static void Main()

{

// Gives one sole instance.

Singleton s = Singleton.getInstance();

// Задаване на стойност.

s.setData(34);

Console.WriteLine("First reference: " + s);

Console.WriteLine("The value of the Singleton is: " + s.getData());

// Gives another instance of the Singleton.

s = null;

s = Singleton.getInstance();

Console.WriteLine("Second reference: " + s);

Console.WriteLine("The value of the Singleton is: " + s.getData());

}

}