**Singleton Design Pattern**

Намерение

Прави така че даден клас да има само едно копие, и предоставя една обща точка за достъп до него.

Проблем

Приложение се нуждае само и единствено от една инстанция на обект. Освен това, мързелива инициализация и глобален достъп са необходими.

Дискусия

Направа на класа съдържащ единствения инстанцииран обект отговорен за създаването, инициализацията, достъпа, и изпълнението. Деклариране на инстанцията като частен статичен обект. Осигуряване на публична статична функция на обекта която да капсулира целия инициализационен код, и осигури достъп до инстанцията.

Клиентът извиква допълнителна функция (използвайки името на класа и оператор) винаги когато е нужна една единствена инстанция.

Singleton трябва да изпълнява следните три критерии:

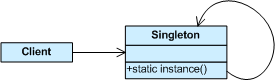
* Собствеността върху единствената инстанция не може да бъде логично определена.
* **„lazy initialization“** (инициализация непосредствено преди използването на даден клас) е желателна.
* Няма друг начин за глобален достъп.

Ако собствеността върху единствената инстанция, кога и как инициализация се случва, и глобалния достъп не са проблемни, „Singleton“ не е достатъчно подходящ.

Шаблонът Singleton може да бъде допълнен така че да подпомогне достъпа до конкретна бройка специфични за приложението копия.

Изтриване на Singleton класа / инстанцията не е тривиален проблем.

Структура



Други модели

Abstract Factory, Builder, и Prototype могат да използват Singleton в тяхната имплементация

Предимства

Предимството на Singleton пред глобалните променливи е, че сте абсолютно сигурни, за броят на инстанциите, когато използвате Singleton, и можете да промените мнението си и управлявате произволен брой инстанции.

using System;

**class** **MainApp**

{

**static** **void** Main()

{

// Constructor is protected -- cannot use new

Singleton s1 = Singleton.Instance();

Singleton s2 = Singleton.Instance();

**if** (s1 == s2)

{

Console.WriteLine("Objects are the same instance");

}

// Wait for user

Console.Read();

}

}

// "Singleton"

**class** **Singleton**

{

**private** **static** Singleton instance;

// Note: Constructor is 'protected'

**protected** Singleton()

{

}

**public** **static** Singleton Instance()

{

// Use 'Lazy initialization'

**if** (instance == **null**)

{

instance = **new** Singleton();

}

**return** instance;

}

}

}