Softvérové inžinierstvo

8

KST Katedra softvérových technológií

Fakulta riadenia a informatiky, Žilinská univerzita v Žiline

Návrhové vzory

Úvod

Obsah

- Čo to je?
- Ukážka
- Teoretické východiská
- Prehľad základných vzorov

Čo je to návrhový vzor?

Každý vzor popisuje problém, ktorý sa objavuje znova a znova, a potom popisuje jadro riešenia takéhoto problému a to takým spôsobom, že ho môžete použiť milión krát dokola, bez toho, aby ste ho čo len raz robili ešte raz rovnakým spôsobom

Problém

- Navrhujem rozsiahly systém
- V systéme potrebujem množstvo objektov:
  - Stačí mi len jediná inštancia
  - Musí byť ľahko dostupná – z ľubovoľného miesta systému

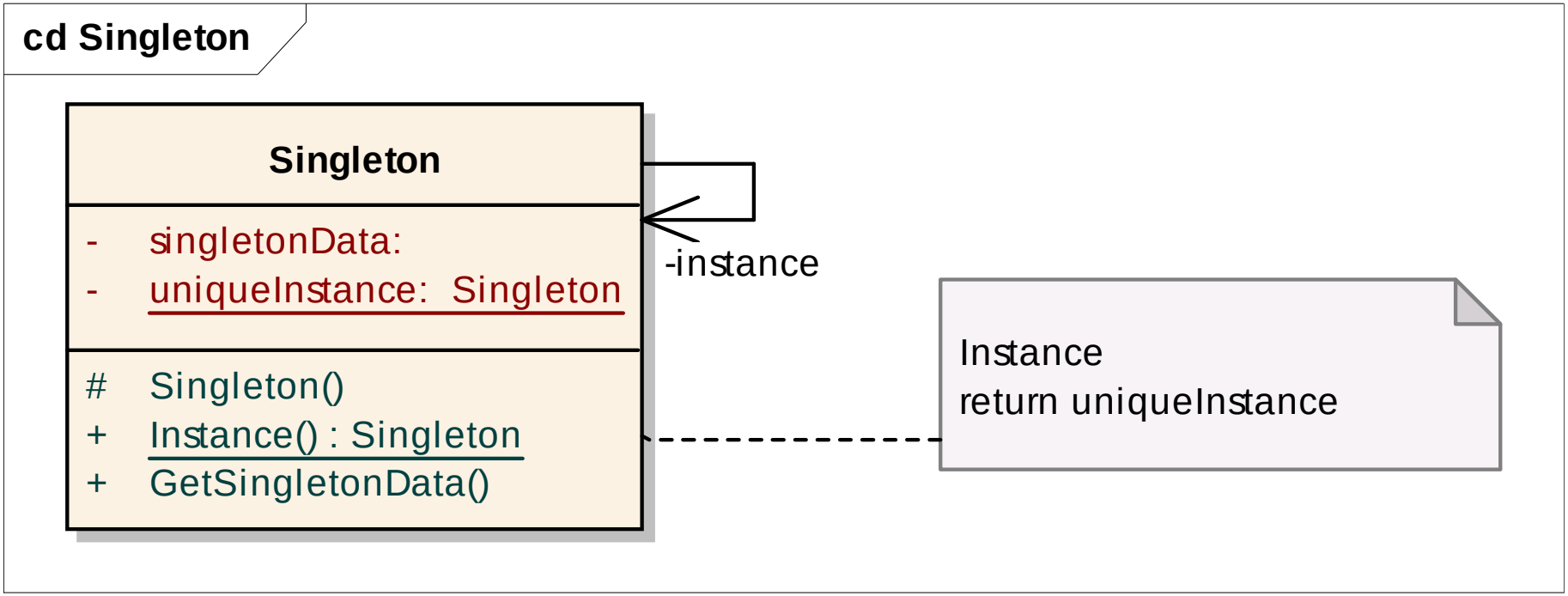
Vaše návrhy?

- Globálny objekt
- Statická trieda a statické metódy

Ale čo ak?

- Keď základná inštancia by mala byť rozširovaná dedením a klienti by mali byť schopný používať rozšírenú inštanciu bez modifikácie vlastného kódu
- Statické metódy nemôžu byť polymorfné

Štruktúra


cd Singleton
Singleton
- singletonData:
- uniqueInstance: Singleton
Singleton()
+ Instance() : Singleton
+ GetSingletonData()
-instance
Instance
return uniqueInstance

Zabezpečenie jedinej inštancie?

```
class Singleton {
private:
        static Singleton* uniqueInstance;
protected:
        Singleton(){};
public:
        static Singleton* Instance() {
                if (uniqueInstance == null) {
                        uniqueInstance = new Singleton;
                }
                return uniqueInstance;
        };
}

Singleton* Singleton::uniqueInstance = 0;
```

Vytváranie potomkov Singletonu

- Inicializácia „uniqueInstance“ inštanciou potomka
  - V potomkoch
  - V Singleton::Instance
    - Podmienené príkazy
    - Register singletonov

Vytváranie potomkov Singletonu?

```
class Singleton {
private:
        static Singleton* uniqueInstance;
        static List<NameSingletonPair>* registry;
protected:
        static Singleton* Lookup(char* name);
public:
        static Singleton* Instance();
        static void Register(char *name, Singleton*);
}
```

Vytváranie potomkov Singletonu

```
Singleton* Singleton::Instance() {
        if (uniqueInstance == 0) {
                char* name = GetNastavenySingletonName();
                uniqueInstance = Lookup(name);
        }
        return uniqueInstance;
}

MySingleton::MySingleton()
{
        Singleton::Register("MySingleton", this);
}

static MySingleton theSingleton;
```