Analýza a návrh informačných systémov II 9

objektovo orientovaný návrh

Peter Bednár

Návrhové vzory - zhrnutie

Vzory pre vytváranie objektov (1)

- Singleton
  - Zabezpečuje iba existovala iba jedna inštancia triedy
  - Pre triedu T: vytvoriť privátnu statickú premennú typu T a verejnú statickú metódu ktorá inicializuje a vráti T
- Factory
  - Vytváranie objektu ak chceme dynamicky poskytnúť rôzne implementácie
  - Klient pracuje s rozhraním I pre ktoré môže existovať viacero implementácií T1, T2, ...
  - Vytvoriť verejnú statickú metódu, ktorá vracia objekt typu I a vnútorne vytvorí požadovanú implementáciu

Vzory pre vytváranie objektov (2)

- Lazy initialization
  - Vytvorenie objektu až keď sa prvý krát používa
  - Pre triedu T pridať verejnú statickú metódu, ktorá vráti T pre požadované parametre a uchová si odkaz na vytvorený objekt.
  - Ak je potrebné znovu vytvoriť objekt pre dané parametre, vráti už vytvorenú inštanciu

Vzory pre vytváranie objektov (3)

- Object pool
  - Na začiatku vytvorí množinu objektov, klient môže po použití objekt vrátiť aby ho mohli použiť iný klienti
  - Pre triedu T pridať triedu TPool, ktorá v konštruktore inicializuje množinu objektov T
  - Do TPool pridať verejné metódy "získaj" T a "vráť" T

Vzory pre vytváranie objektov (4)

- Builder
  - Vytváranie zložitých objektov typu T zložených z rôznych častí C1, C2, ...
  - Pridať TBuilder ktorý bude mať verejné metódy pre každú časť C1, C2, ... ktoré postupne pridávajú do nového objektu T jednotlivé časti
    - Metódy vracajú TBuilder aby sa dalo volanie zreťaziť
  - Do TBuilder pridať verejnú metódu ktorá na koniec vráti vytvorený T

Štrukturálne vzory (1)

- Marker
  - Pridávanie metadát do kódu
  - Definuje sa podobne ako rozhranie s `@interface`
- Adapter
  - Zmení rozhranie typu T1 na rozhranie T2
  - Pre požadované rozhranie T1 a triedu T2 vytvoríme triedu T2Adapter, ktorá bude implementovať T1
  - T2Adapter "obalí" T2 a bude volať jeho metódy

Štrukturálne vzory (2)

- Composite
  - Typ T zložený z typov C1, C2, ... (základný vzor)
  - V T vytvoríme členské premenné pre C1, C2, ...
    - Niektoré časti môžu byť aj rekurzívne T
- Facade
  - Chceme zjednotiť rozhrania I1, I2, ... do jednoduchšieho rozhrania I
  - Podobne ako pri Adapter, len je viacero rozhraní
  - Pre I vytvoríme triedu IFacade, ktorá "obalí" I1, I2, ... a bude volať ich metódy

Štrukturálne vzory (3)

- Decorator
  - Chceme zmeniť niektoré metódy objektu o typu T
  - Pre rozhranie T vytvoríme triedu TDecorator, ktorá implementuje rozhranie T a ktorá obaľuje menený objekt o
  - TDecorator zmení implementáciu niektorých metód, nezmenené metódy len deleguje na objekt o

Behaviorálne vzory (1)

- Command
  - Chceme zabaliť volanie metódy objektu T do objektu C
  - C bude mať odkaz na T a všetky parametre, ktoré sú potrebné pre volanie danej metódy
- Iterator
  - Chceme prechádzať prvky typu T objektu O
  - O bude implementovať `java.util.Iterable<T>`
    - metóda `Iterator<T> iterator()`
  - Implementujeme triedu TIterator ktorá bude implementovať rozhranie `java.util.Iterator<T>`
    - metódy `boolean hasNext()` a `T next()`

Behaviorálne vzory (2)

- Visitor
  - Pre objekt typu T zloženého z častí typu C1, C2, ... chceme prechádzať jeho časti
  - Vytvoríme triedu TVisitor, ktorá bude mať metódy "navštív" pre každý typ C1, C2, ...
  - V triede T vytvoríme metódu "príjmi"(TVisitor v) ktorá bude prechádzať časti a volať metódy "navštív" objektu v

Behaviorálne vzory (4)

- Publish/Subscribe alebo Observe
  - Chceme byť notifikovaný ak sa vykoná nejaká metóda nad objektom typu T
  - Vytvoríme triedu TObserver ktorá bude mať metódu "spracuj" pre notifikáciu o zmene
  - Do triedy T pridáme zoznam pozorovateľov typu TObserver
  - Do objektu T pridáme metódy "zaregistruj"(TObserver o) a "odregistruj"(TObserver o)
  - Ak sa vykoná pozorovaná metóda, zavoláme metódy "spracuj" všetkých pozorovateľov