Princípy tvorby softvéru
Design patterny

Robert Lukoťka
lukotka@dcs.fmph.uniba.sk
www.dcs.fmph.uniba.sk/~lukotka

M-255

Čo to je?

Design pattern:

Repeatable solution to a commonly occurring problem in software design.

- Tak by sme mali znovupoužiť vhodný kus kódu....
- Design patterny sú skôr konceptuálne - ide o štruktúru tried.

Čo to je?

- Znovupoužiteľne riešenia pre design
- Vytvárajú terminológiu, ktorá uľahčuje komunikáciu o dizajne.
- Známe riešenie je ľahšie pochopiteľné.
- Inšpirácia pre situácie, ktoré nie sú pokryté design patternami.
- Hovoria o slabinách programovacích jazykov.
- Bežné design patterny prenikajú do programovacích jazykov.
- Duck-typing jazyky potrebujú oveľa menej elementov na dosiahnutie rovnakého cieľa, avšak niektoré elementy, aj keď nie sú v programe priamo definované (napr. interfacy) sú dôležité z konceptuálneho hľadiska.

Príklad - dekorátor

Decorator pattern

- Cieľ: Pridávať dynamicky zodpovednosti objektu
- Dá sa to vyriešiť mnohými inými spôsobmi, ale
  - Výsledný dizajn môže byť horší.
  - Výsledný dizajn bude ťažšie odkomunikovateľný a pochopiteľný.

Delenie

Delenie:

- Creational
- Structural
- Behavioral

Máme ale aj

- Concurrency patterns
- Domain-specific patterns
- ...

Creational design patterns

- **Factory method** - metóda (môže byť aj statická nie konštruktor) triedy, ktorá vracia nové inštancie nejakej triedy.
  - Volaná metóda môže za rôznych okolností vrátiť inštancie rôznych tried.
  - Volajúci nemusí vedieť presnú triedu vrátenej inštancie.
  - Factory method môže existovať vo viacerých verziách a môže byť "injectnutá" (je implementovaných viacero tried implementujúcich rovnakú factory metódu s rôznymi výsledkami).
- **Abstract factory** - abstraktná trieda obsahuje niekoľko súvisiacich factory metód.
  - Spravidla existuje viac implementácií tiedy.

Factory method - Príklad

- Graf môže mať sparse alebo dense reprezentáciu
- Ako napísať funkciu, ktorá vytvára grafy, ale nechce vedieť, či želaná reprezentácia je sparse alebo dense?
  - Graph - interface (obsahuje napriklad addVertex(...), addEdge(...))
  - SparseGraph - implementuje Graph, DenseGraph - implementuje Graph
  - GraphFactory - interface obsahyje metodu getEmptyGraph()
  - SparseGraphFactory a DenseGraphFactory implementujú GraphFactory
  - Parametrom funkcie bude objekt typu GraphFactory.
  - Funkcia si pomocou GraphFactory.getEmptyGraph() vytvori správny typ grafy a pomocou metód interfacu Graph vykoná potrebne zmeny.

Creational design patterns

- **Builder** - Objekt, ktorý postupne počas viacerých krokov vytvára nový objekt.
  - Ak je proces tvorby objektu komplexný existujú v tried de facto dve sady metód, na vytvorenie objektu a na používanie objektu (porušuje SRP, musíme objekt najprv vytvoriť až potom s ním pracovať)
  - Počas tvorby objektu môže byť na uchovanie objektu vhodná iná dátová štruktúra
- Príklad: StringBuilder

Builder príklad

- Graph má metódy addEdge() a removeEdge()
- Chceme vložiť do hrany vrchol.
- Môžeme odstrániť hranu a pridať novu to ale:
  - Zbytočne posúva zoznamy susedných hrán
  - Mení poradie hrán
- Riešením je GraphBuilder ktorý nemusí byť vždy konzistentný (hrana nemusí mať aj druhú stranu)

Creational design patterns

- **Object pool** - Namiesto vytvorenia triedy používame "vrátené inštancie tried".
  - Užitočné, ak je inštanciu triedy ťažké vyrobiť
  - ThreadPool, ConnectionPool
- **Prototyp** - Nový objekt vytvárame kopírovaním fixného objektu - prototypu.
- **Singleton** - Trieda, ktorá môže mať iba jednu inštanciu.
- ...

Structural patterns

- **Decorator**
- **Composite** - Stromová štruktúra zložených objektov.
- **Fasáda** - Trieda, ktorá reprezentuje celý podsystém.
- **Adaptér** - Trieda na upravenie interfacu triedy.
- **Proxy** - Objekt reprezentujúci iný objekt.
  - Access proxy, remote proxy, virtual proxy, ...
- **Flyweight** - Rozdelíme triedu na časť spoločnú pre viacero inštancií a na časť špecifickú pre inštanciu.

Behavioural patterns

- **Iterator**
- **Observer** - Notifikovať objekty o zmenách.
- **Strategy** - Enkapsulácia algoritmu v triede.
- **Template method** - metóda v abstraktnej triede používajúca abstraktné metódy.
- **Null object**
- **Memento** - Uchová a obnoví stav objektu
- **Visitor** - Reprezentuje operáciu, ktorá sa má vykonať na prvkoch objektovej štruktúry. Vizitor umožňuje definovať novú operáciu bez zmeny tried nad ktorými pracuje.
- ...

Code smells

Code smell is

"a code smell is a surface indication that usually corresponds to a deeper problem in the system" - - M.Fowler - -

- Code smell nie je bug.
- Môže indikovať technical debt.

Refaktorizácia zdrojového kódu je proces reštrukturalizácie existujúceho kódu bez zmeny externého správania.

Software entrophy

Ivar Jacobson et al.:

The second law of thermodynamics, in principle, states that a closed system's disorder cannot be reduced, it can only remain unchanged or increase. A measure of this disorder is entropy. This law also seems plausible for software systems; as a system is modified, its disorder, or entropy, tends to increase. This is known as software entropy.

M.M.Lehman, L.A.Belady:

1. *A computer program that is used will be modified*
2. *When a program is modified, its complexity will increase, provided that one does not actively work against this.*

Incorporating refactoring into software development process

- Refaktorizácia by mala byť pravidelnou súčasťou prác na programe

Príklad: Test-driven development

1. Pridaj test
2. Spusti testy aby si sa presvedčil, či zlyhali
3. Napíš kód
4. Spusti testy aby si sa presvedčil, či prešli
5. Refaktoruj kód
6. Spusti testy aby si sa presvedčil, či prešli

Všimnite si striktné oddelenie pridávania funkcionality a refaktorovania.

Code smells - príklady

Code smells - sourcemakong

- **Dlhá trieda**
- **Veľa argumentov metódy**
- **Switch statement**
- **Parallel inheritance hierarchies**
- **Opakovaný kód**
- **Veľa komentárov** (ktoré vyzerajú užitočne a potrebne)
- ...

Refactorizácie - príklady

Code smells - sourcemakong

- **Decompose complex conditional**
- **Extract Method**
- **Extract Variable**
- **Replace Nested Conditional with Guard Clauses**
- **Introduce Parameter Object**
- **Form Template Method**
- ...

UML

- Class diagrams - abstract, signatúry atribútov/metód, static
- Sequence diagrams main elements

Zdroje

- Sourcemaking
- oodesign.com
- Wiki