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 programovacich jazykov.
- Dinamické jazyky potrebujú oveľa menej elementov na dosiahnutie rovnakého cieľa, avšak niektoré elementy, aj keď nie sú v programe priamo definované (napr. interfacy) sú dôležité s konceptuálneho hľadiska.

Príklad - dekorátor

- 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.

Creational design patterns

- **Builder** - Objekt, ktorý postupne počas viacerých krokov vytvára nový objekt.
  - Ak je proves tvorby objektu komplexný existujú v tried de facto dve sady metód, na vytvorenie objijektu a na používanie objekto (porušuje SRP, musíme onjekt najprv vytvoriť až potom s ním pracovať)
  - Počas tvorby objektu môže byť na uchovanie objektu vhodná iná datová štruktúra
- **Object pool** - Namiesto vytvorenia triedy používame "vrátené inštancie tried".
  - Užitočné, ak je inštanciu triedy ťažké vyrobiť
- **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, ... (sourcemaking).
- **Flyweight** - Dozdelíme 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
- ...

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.

Code refactoring is the process of restructuring existing computer code—changing the factoring—without changing its external behavior.

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

Príklad: Test-driven development

1. Add a test
2. Run all tests and see if the new test fails
3. Write the code
4. Run tests
5. Refactor code
6. Run tests

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**
- ...

Zdroje

- Sourcemaking
- oodesign.com
- Wiki