Návrhové vzory

Návrhové vzory jsou obecná řešení často vyskytujících se problémů v architektuře softwaru. Nejedná se o knihovnu nebo přesný kus kódu, ale popsání postupu / šablona, jak problém řešit.

Řeší různé problémy, jako je například omezení výkonu počítače.

Jsou dohromady čtyři typy.

Creational

Jedná se o vzory, které pomáhají ke tvorbě objektů. Často nahrazují nebo se obalují okolo keywordu „new“, Mezi nejznámější patří například:

- Singleton - Zamezuje vytvoření druhé instance třídy.
- Lazy Initialization – Odkládá vytvoření objektu do té doby, dokud není potřeba

Singleton

Singleton slouží k tomu, aby se nevytvořilo více instancí - vytvoři pouze jednu a v případě vytvoření druhé se jedná o zcela tu samou se stejným hash codem

Lazy Initalization

Lazy initalization, jak už název napovída, odkládá vytvoření objektu do té doby, dokud není potřeba. V Javě se tato implementace řeší pomocí keywordu synchronized a řeší se přes vlákna. Výhody jsou zrychlení počáteční inicializace + inicializace začíná až v tu chvíli, kdy je opravdu potřebná.

Structural

Jedná se o vzory, které se týkají celkové struktury programu. Mezi nejznámější patří například

- Flyweight – Umožňuje řízení počtu instancí.
- Composite – Řešení uspořádání malých objektů do velkých.
- Fasáda – Ulehčuje komunikaci s uživatelem.

Flyweight

Řídí počet instancí. Je dobrý v případě, že vzniká mnoho drobných objektů. Sdílí jednu instanci pro všechny výskyty, klasický příklad: místo 10x instance písmena „a“ máme jednu sdílenou instanci písmene „a“.

Composite

Umožňuje řešení, jak uspořádat malé objekty a velké objekty, kterými jsou ony malé součástí.
Příklad : Stádo, malých stád, ve kterém jsou zvířata. Po velkém stádu chceme, aby všechna zvířata udělala zvuk.

Facade

Facade (Fasáda) je návrhový vzor, který je vhodný v případě, že máme mnoho tříd a systém je komplexní pro splnění jednoduchého úkolu. Účelem je zamezit velkému počtu tříd, které komunikují s uživatelem a místo toho vytváří jednotnou třídu / jednotnou skupinu tříd.

V Javě se využívá zejména s JOptionPane, který nahrazuje Jframe a poskytuje jednoduché prostředí.

Behavioar

Jedná se o vzory, které pomáhají řešit celý běh programu, respektive jeho chování. Mezi nejznámější patří například

- Iterátor – Umožňuje nám procházet prvky v různých strukturách.
- Null Object – Nahrazuje null hodnotu.

Iterátor

Iterátor nám umožňuje procházet prvky v různých strukturách bez toho, abychom znali jejich implementaci. Lze vytvořit pomocí pole, ale i jiných datových struktur. Pomocí ArrayListu je implementace nejjednoduší.

Java jako taková má **vlastní** iterátor, který se dá získat pomocí **importu**. (import java.util.Iterator)

```
Iterator<String> it = struktura.iterator();

it.next();
```

Null objekt

Místo toho, aby určitá hodnota byla null, můžeme ji nahradit Null Objektem, který se snaží eliminovat většinu problémů s ním spojeným, jako NullPointerException a podobné. Tím zamezíme i duplicitními kódu kontrolující null v podmínkách například.

Concurential

Jedná se o vzory, které řeší problémy, při kterých program běží na několika vláknech najednou a řeší souběžně určité operace. Mezi nejznámější patří například

- Thread Pool – Bazének několika vláken, které řeší určité problémy seřazené ve frontě.
- Read Write Lock – Tzv. RWL slouží k umožnění vláknům k přístupu pro čtení v jeden a ten samý čas.

Thread Pool

Thread pool je bazének vláken, které řeší různé problémy seřazené ve frontě a hlavní výhodou tohoto návrhového vzoru je dosažení konkurence – Stavu, ve kterém se výpočty a části programů mohou řešit mimo normální stanovený průběh.

Read Write Lock

Read Write Lock, také zvaný RWL umožňuje přístup k jednotlivým proměnným nebo polím v jeden moment. Write je exkluzivní, musí se použít lock. Jedná se také o synchronizační primitivum.

Zámky fungují na tom principu, že jedno vlákno přistoupí k metodě. Ta na prvním řádku spustí lock -> V tu chvíli do ní nemůže jiné vláknou vstoupit a musí počkat, až první vlákno dojede. Teprve poté se zámek odemkne, druhé vlákno může vstoupit do metody.

Locky zabraňují možnostem v jednom momentu zapsat na stejnou pozici item, přistoupit k něčemu co by už nemuselo existovat a podobné.