Strategy

Policy, Stratégia

Definuje rodinu algoritmov, každý zapúzdruje a umožňuje ich zamieňať. Stratégia umožňuje meniť algoritmy nezávisle na klientovi, ktorý ich používa.

Define a family of algorithms, encapsulates each one and make them interchangable. Strategy lets the algorithm vary independently from clients that use it.

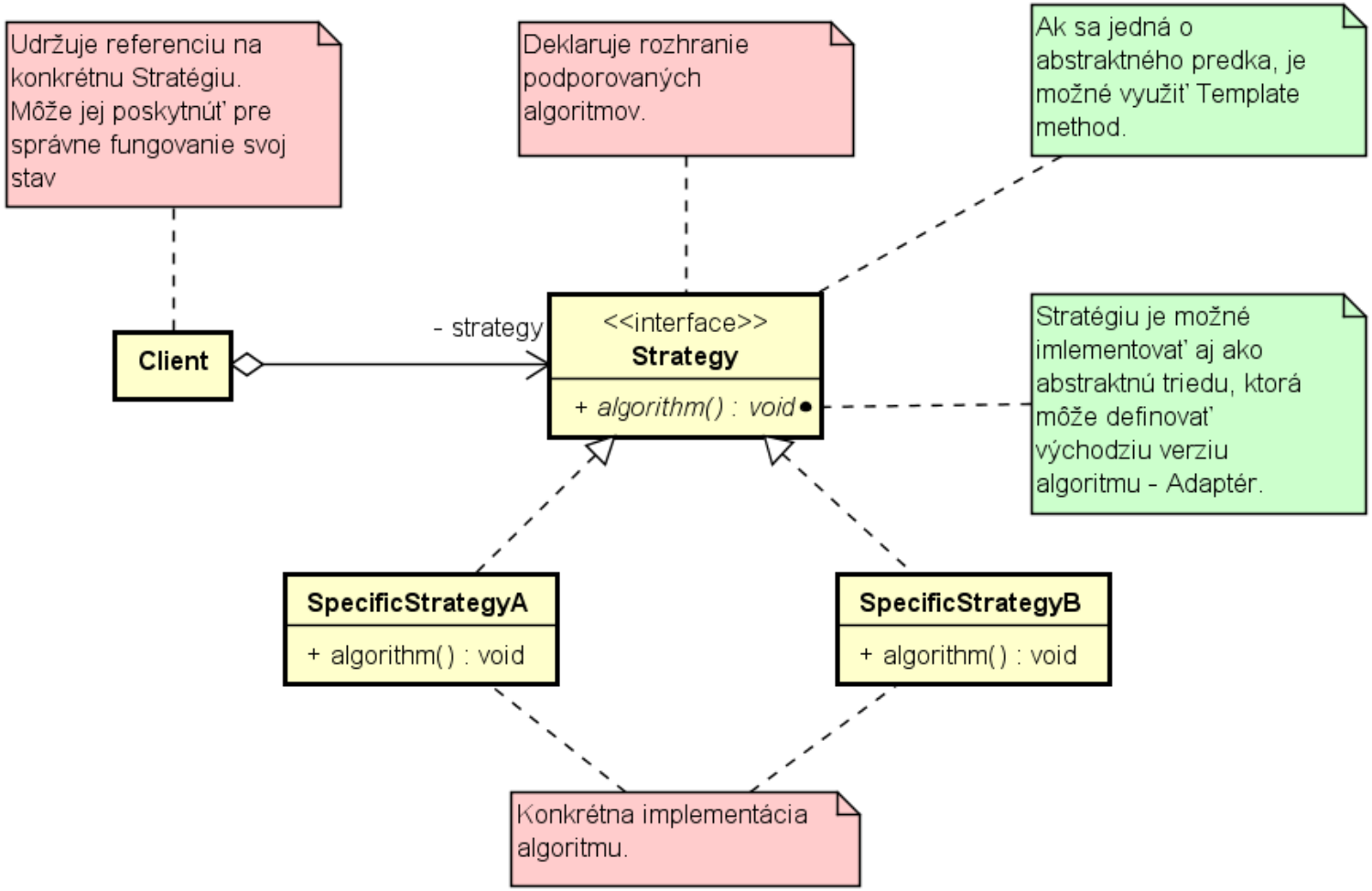
Motivácia

- Internetové obchody ponúkajú rôzne možnosti platenia za tovar:
  - karta,
  - Paypal,
  - dobierka..
- Všetky spôsoby realizujú platbu, ale každý to robí špecifickým spôsobom.
- Ako zabezpečiť, aby boli jednotlivé spôsoby platenia rovnocenné, klientská aplikácia ostala ne výbere konkrétnej platby nezávislá a aby bolo do budúcna možné jednoducho pridať ďalší spôsob platby?

Aplikovateľnosť

- Stratégiu je vhodné použiť, keď:
  - sa mnoho príbuzných tried líši iba v ich správaní. Stratégia umožňuje vybrať si konkrétne správanie.
  - požadujete rôzne verzie algoritmu.
  - jednotlivé algoritmy požadujú pre svoj beh vlastné dáta, o ktorých klient nepotrebuje vedieť. Bez použitia stratégie by došlo k odhaleniu implementačných detailov.
  - sa trieda môže správať rôzne a o jej správaní rozhodujú relatívne komplikované podmienky. Namiesto zložitého vetvenia je možné každé správanie zapúzdriť do Stratégie a komplikované podmienky presunúť tam.

Implementácia



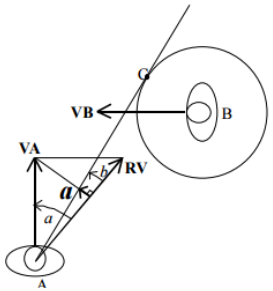
Dôsledky

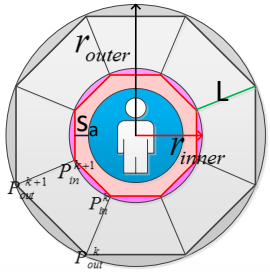
- Hierarchia Stratégií definuje rodiny príbuzných algoritmov, pričom predok vie definovať spoločné správanie.
- Je to alternatíva k dedičnosti, čo sa týka zmeny funkčnosti objektu. Stratégie je možné meniť dynamicky za behu programu.
- Redukuje vetvenie v kódu.
- Umožňuje využiť rôzne algoritmy pre riešenie toho istého problému (triedenia). Klient sa pre konkrétny algoritmus môže rozhodnúť po zvážení všetkých plusov a mínusov (napr. koľko prostriedkov má k dispozícii).
- Pre efektívny výber konkrétnej stratégie musí klient vedieť, ako daný algoritmus stratégia implementuje.
- Nakoľko všetky stratégie využívajú rovnaké rozhranie na komunikáciu s klientom, môže sa stať, že klient dodá konkrétnej Stratégii viac informácií, ako potrebuje (získanie týchto informácií pritom môže byť časovo náročné).
- Zavádza do aplikácie ďalšie objekty. Ak sa to stane problémom, je možné Stratégie implementovať s využitím Mušej váhy ako bezstavové objekty.

Príklad – pohyb chodca

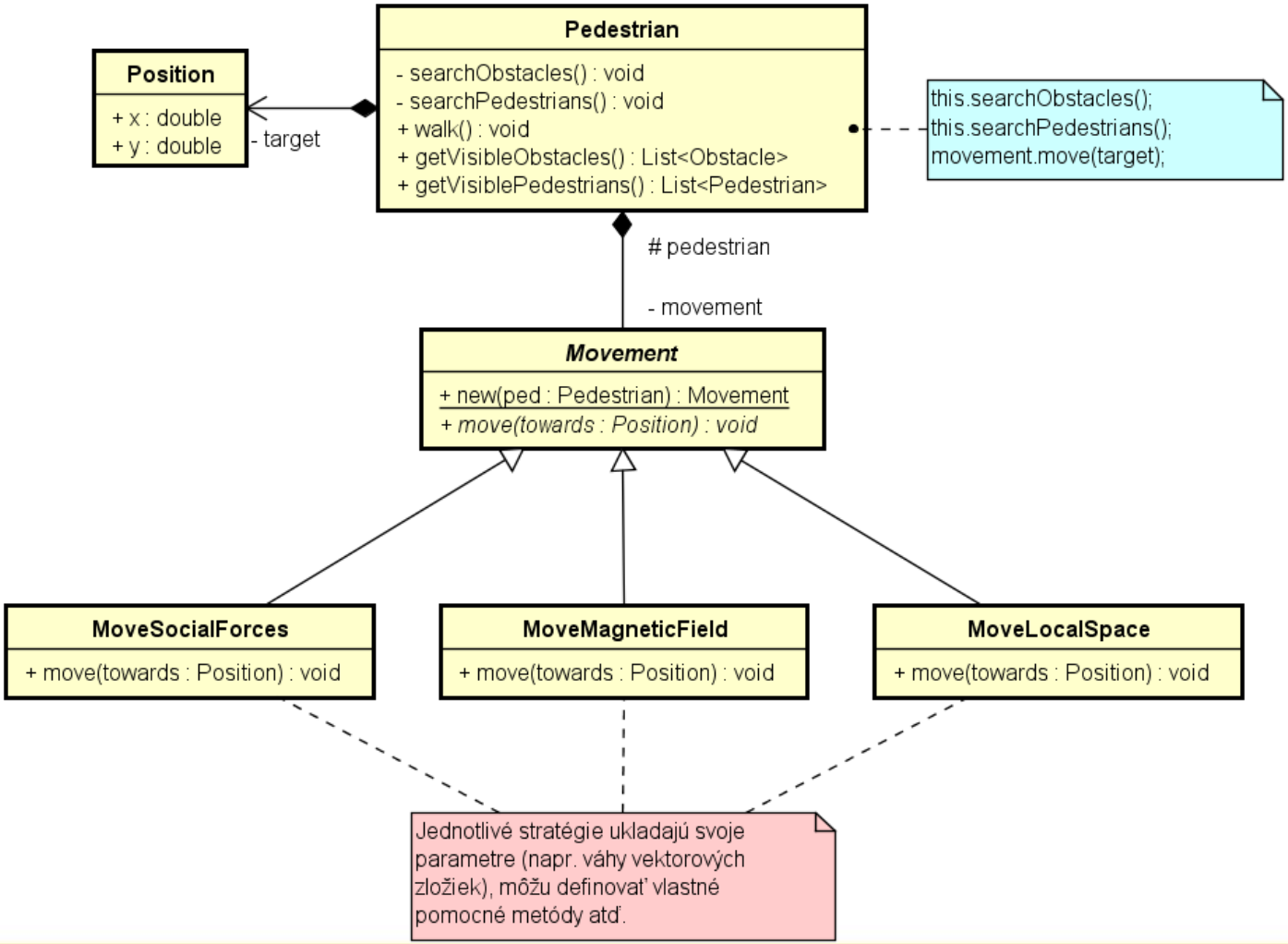
- Chodci môžu meniť model svojho pohybu počas behu programu (jednotlivé pohybové modely sú lepšie/horšie v rôznych prostrediach).
  - Sociálne sily
  - Magnetické pole
  - Ohodnotenie lokálneho okolia.
- Zmena modelu pohybu je určená riadiacim mechanizmom, pre správanie chodca sa však nič nemení – mení sa len spôsob výpočtu ďalšej polohy.

$$\overrightarrow{f_i}(t) = \overrightarrow{f_i^{av}}(t) + \sum_{j(\neq i)} \left(\overrightarrow{f_{ij}^{soc}}(t) + \overrightarrow{f_{ij}^{att}}(t)\right) + \sum_b \overrightarrow{f_{ib}}(t) + \sum_k \overrightarrow{f_{ik}^{att}}(t)$$





Príklad – pohyb chodca



Príbuzné vzory

- *Flyweight* – Stratégie sa často implementujú ako mušie váhy.
- *Bridge* – Na rozdiel od Mostu sa Stratégia zameriava na chovanie systému (výmena jeho súčastí za behu), nie na jeho architektúru. Most sa zaoberá prepínaním (premostením) medzi trvalými súčasťami aplikácie.

Upozornenie

- Tieto študijné materiály sú určené výhradne pre študentov predmetu 5II132 Návrhové vzory (Design Patterns) na Fakulte riadenia a informatiky Žilinskej univerzity v Žiline.
- Reprodukovanie, šírenie (i častí) materiálov bez písomného súhlasu autora nie je dovolené.

Ing. Michal Varga, PhD.
Katedra informatiky
Fakulta riadenia a informatiky
Žilinská univerzita v Žiline
Michal.Varga@fri.uniza.sk