

KSI 2013/2014

Úloha 5-1: Pivnice

Jan Horáček

Gymnázium, Brno, Vídeňská 47; jan.horacek@seznam.cz

13. dubna 2014

1 Zobecnění zadaných úloh

Pokud zadané úlohy řešíme pouze z pohledu hierarchie objektů (resp. tříd) v pivnici, lze problém do značné míry zobecnit.

Uvažujme, že všechny objekty jsou trojrozměrné. Například pro řešení problému s myší nám stačí vytvořit si jakousi matici hranic všech objektů v pivnici ve výšce 0 (výška podlahy nad podlahou = 0). Zanedbáme výšku myši a provedeme prohledávání do šířky nad dvourozměrnou maticí (chcete-li grafem), které nám vrátí nejkratší cestu.

Druhá úloha mi přijde z hlediska trojrozměrnosti mírně nejednoznačně zadaná a tak jsem si stanovil, že alarm chrání pouze v ploše, nikoliv v prostoru. Pokud hledáme nejoptimálnější umístění alarmu, hledáme ho opět pro určitou výšku nad zemí (pro určitý řez místností rovnoběžný s podlahou). Opět nám tedy stačí 2D plocha v určité výšce.

Problém potopy je velmi podobný. Stačí nám vytvořit jakousi matici objektů v místnosti pro určitou výšku vody. Tedy řez pivnicí pro konkrétní výšku od podlahy. Z této matice se pak jednoduše pokusíme zachránit všechny objekty.

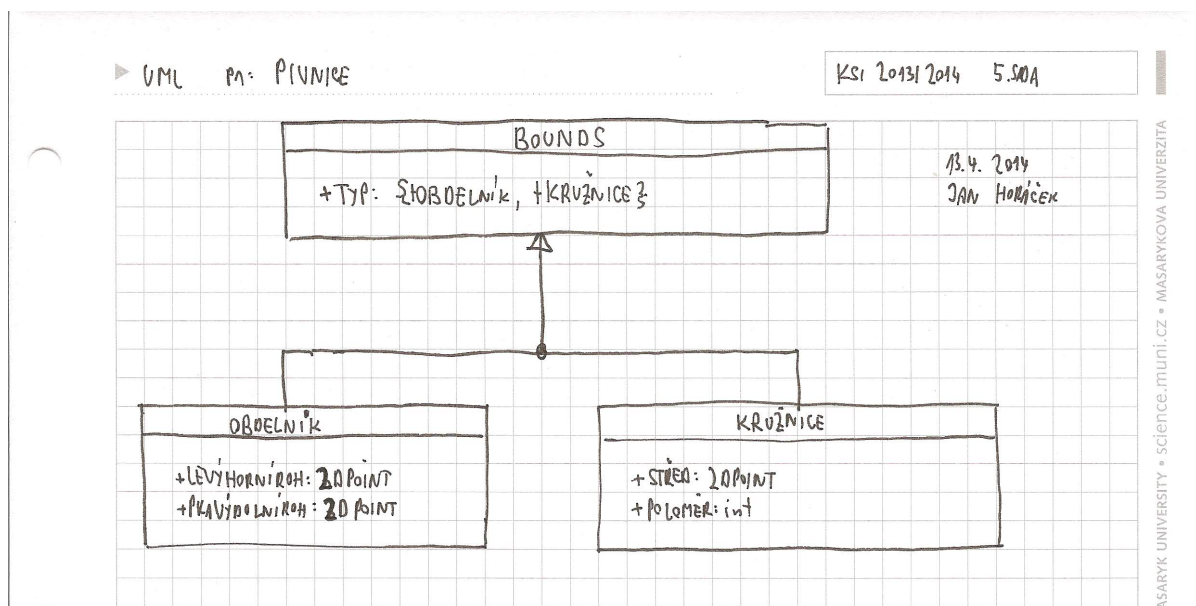
Požadavky na databázi předmětů jsou tedy jasné (vlastně je jen jeden): chceme vrátit všechny předměty v dané výšce.

2 Vrácení hranic

Řekněme, že se snažíme vyřešit především problém, jak od daného předmětu získat jeho hranice. Pokud jsme schopni získat hranice od každého předmětu, jsme schopni je algoritmem řešící konkrétní úlohy vrátit jako množinu hranic.

Povaha předmětů v pivnici je taková, že jejich dvourozměrný řez rovnoběžný s podlahou je buď obdélník (ve speciálním případě čtverec), nebo kružnice. Pokud byste měli pocit, že to tak není, vězte, že jsem si to sice trochu zjednodušil, ale pokud bychom chtěli být precizní, není problém další tvary implementovat tak, aby do celé hierarchie bez problémů zapadly.

Mějme hierarchii tříd zobrazenou v UML na obrázku 2. U každého objektu v pivnici pak stačí vrátit instanci třídy *Bounds* (resp. instanci některé nevirtuální třídy *Kružnice*, či *Obdelník*) obsahující konkrétní data o poloze tohoto objektu. Touto hierarchií jsme zobecnili různé tvary hranic do jedné virtuální třídy *Bounds*.



Obrázek 1: UML hraničních tříd

3 Hierarchie předmětů

Hierarchii tříd reprezentující pivnici přikládám v obrázku 3.

Zde bych rád upozornil na to, že téměř všechny vlastnosti (atributy) jsou privátní, protože pro potřeby daných úloh nejsou zapotřebí. Do UML je přikládám čistě z důvodu prezence atributů, které vyžaduje zadání. Ve skutečnosti jsou ale některé z nich naprosto nesmyslné, protože data sice obsahují, ale nikdo se k nim nedostane.

Nyní bych se rád zabýval principem vrácení hranic. Abstraktní třída *Objekt* implementuje virtuální metodu *get_bounds*, která je v každé konkrétní nevirtuální třídě (rozuměj konkrétním předmětu) přetížena. Ve třídách předmětů je dokonce upřesněna návratová hodnota této funkce, protože např. míč nemůže mít řez typu *Obdelnik*. Zde bych rád zdůraznil, že třída *Pneumatika* tyto návratové hodnoty detailněji nespecifikuje, protože tvar řezu je závislý na orientaci pneumatiky.

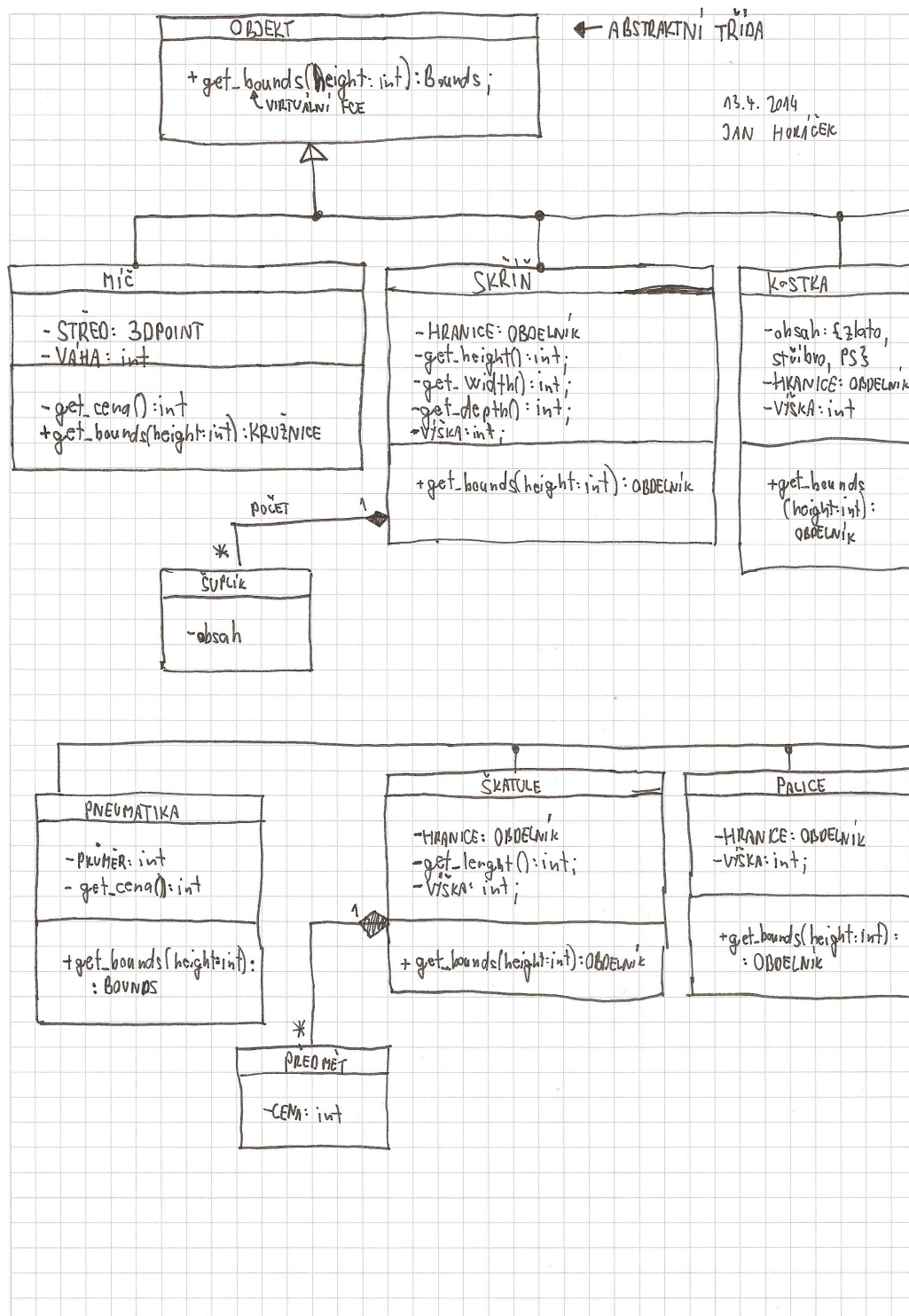
4 Praktická realizace

Ukažme si řešení na praktickém příkladu.

Uvažujme například problém myši - algoritmus prohledávání do šířky nad maticí hranic předmětů. Před prováděním prohledávání vzneseme požadavek na databázi předmětů, např.

```
pivnice.get_all_bounds(0);
```

kde 0 značí výšku, ve které chceme udělat řez (v našem případě 0 = podlaha). Databázový systém pak zavolá funkci *get_bounds* pro každý předmět v pivnici. Každý předmět vrátí na základě polohy, kterou má uloženou v lokálních atributech, své hranice. Pokud se daná rovina nachází mimo tento objekt, nevrátí tento předmět hranice (resp. vrátí prázdné hranice). Databázový systém spojí jednotlivé hranice do množiny hranic a tuto množinu vrátí. Pak může dojít například k převedení množiny hranic do matice (dvojměrného pole) a prohledávání do šířky může začít.



Obrázek 2: UML předmětů v pivnici