

KSI 2013/2014

# Úloha 5-4: Elektronické bastlení

Jan Horáček

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

13. dubna 2014

## 1 UML

Výsledné UML přikládám v obrázku 1.

## 2 Úpravy oproti zadání

Nyní bych rád popsal změny, které jsem ve schématu provedl.

- **Cena : int**

Nechápu, proč je v zadání cena datového typu *string*. Toto je naprosto nevhodný datový typ pro operace, které po nás zadání požaduje (jako např. sčítání). Proto jsem všechny ceny změnil na datový typ *integer*.

- **Abstraktní třída *Souc***

*Souc* je abstraktní třída "součástka", která je předkem všech součástek. Tj. všechny součástky, jako *Rezistor*, či *Potenciometr* z ní dědí a zároveň nelze vytvořit instanci třídy *Souc* jako takové. Cena součástky je nově implementována ve třídě *Souc*, abychom si ulehčili práci například při výpočtu celkové ceny obvodu, nebo abychom nemuseli mít desítky funkcí *spoj\_vodic*.

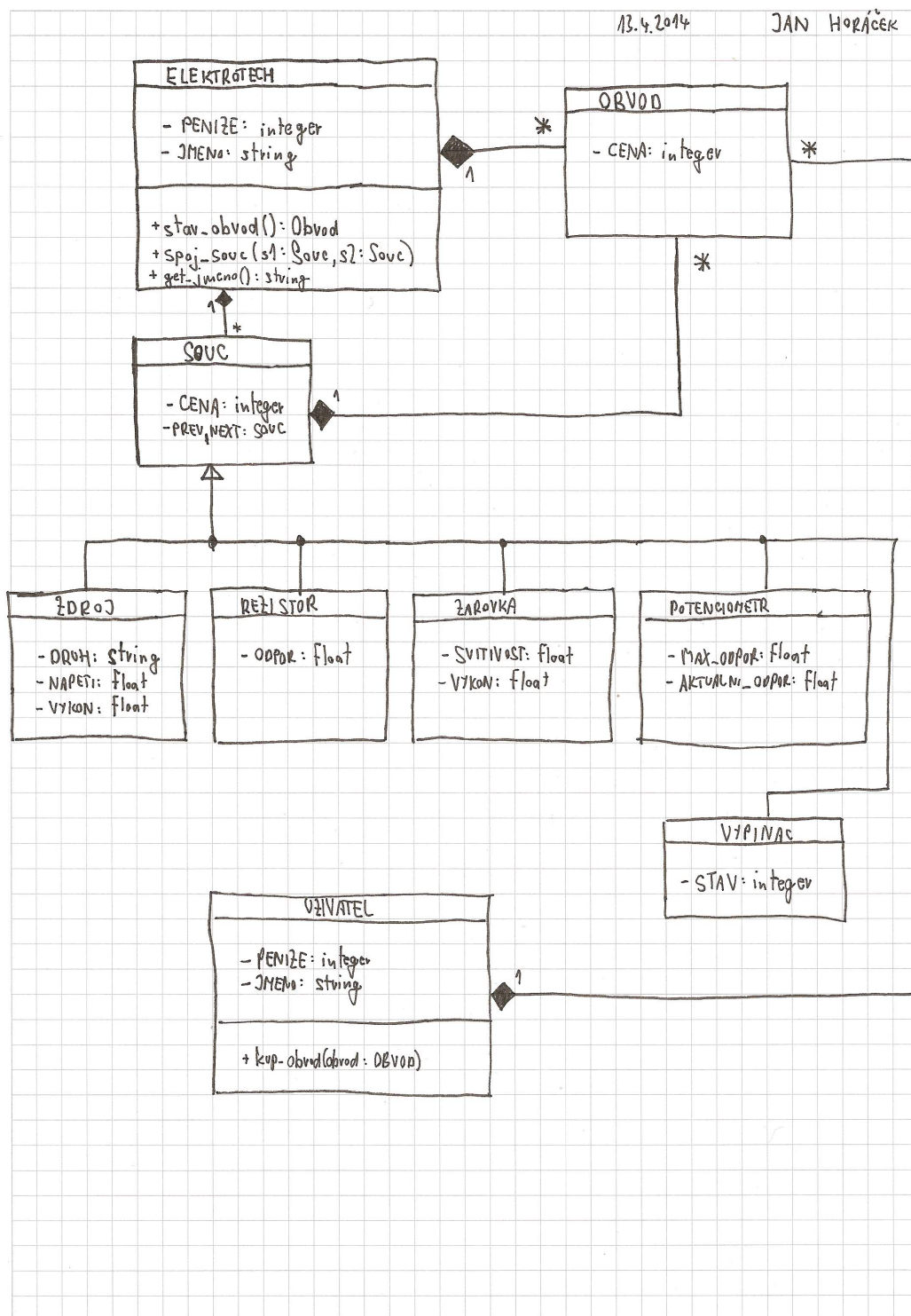
- **Lokální atributy *PREV* a *NEXT* ve třídě *Souc***

Tyto atributy fakticky nahrazují třídu *Vodic*. Instance této třídy plnily jednu důležitou funkci, na kterou nemůžeme jen tak zapomenout: totiž byla v ní uložena struktura obvodu. Bez těchto objektů by byl obvod jen zmetí nepropojených součástek.

Nově je informace o propojení přímo atributem součástky. A protože ze zadání plyne, že každá součástka má 2 vývody (včetně např. potenciometru ☺), je struktura obvodu uložena tak, že každá součástka si pamatuje, co je připojeno na její vývody (resp. do jaké další součástky příslušný vývod vede). V terminologii *C++* lze říci, že *PREV* a *NEXT* je ukazatelem na *Souc*. Celý obvod je tak definován pouze pomocí příslušných součástek (resp. součástek do konkrétního obvodu zapojených). Je důležité zdůraznit, že před sestavením obvodu mají všechny součástky u Karlíka na skladě *PREV* a *NEXT* nastavený na *NULL*. O navázání těchto ukazatelů se stará až metoda *spoj\_souc*.

13.4.2014

JAN HORÁČEK



Obrázek 1: UML

- **Metoda *spoj\_souc***

Metody *spoj\_vodic* jsem nahradil jednou metodou *spoj\_souc*, která spojí součástky, které dostane jako argumenty. Abstraktní třída *Souc* umožňuje, že stačí jediná metoda pro všechny typy součástek.

- **Zrušení třídy *Vodic***

Po té, co jsme efektivněji vyřešili vše, co řeší jednotlivé vodiče, si můžeme klidně dovolit tuto třídu zrušit. Dalším argumentem pro zrušení vodičů je například to, že jich máme nekonečno a tudíž si nemusíme pamatovat dokonce ani jejich počet.

- **Zrušení autora obvodu**

Třída *Obvod* si nově nepamatuje autora jako svůj atribut, ale prostřednictvím vazby na objekt *Elektrotech*, který zpřístupňuje své jméno pomocí getteru *get\_jmeno()* : *string*. Tím jsme odstranili duplicitu.

- **Obvod si udržuje databázi součástek**

Nově je do návrhu přidána vazba mezi třídou *Obvod* a třídami *Souc*. Obvod si totiž udržuje databázi součástek (které zároveň nesou informaci o tom, jak jsou navzájem propojené). Zde bych rád zdůraznil, že při tvorbě obvodu (v metodě *stav\_obvod*) přechází součástky od Karlíka (resp. z jeho skladu) do obvodu. Součástka tedy může být navázána buď na obvod, nebo na elektrotecha, nikoliv na obojí zároveň. To zvyšuje přehlednost celého systému (Karlík si například může jednoduše spočítat, kolik nepoužitých součástek má aktuálně na skladě).

- ***Vypinac* má *stav***

2 nesmyslné stringy *Vypnuto* a *Zapnuto* jsem nahradil jednou stavovou proměnnou *Stav*. Tím jsem zvýšil přehlednost a snížil paměťovou náročnost systému.

### 3 Předpoklady pro funkčnost systému

- **Obvod má jen jednoho autora**

Plyne z typu vazby mezi třídami *Elektrotech* a *Obvod* (a ze zadání).

- **V jeden okamžik lze stavit maximálně jeden obvod**

Metoda *stav\_obvod* vrací obvod, resp. součástky podle toho pravidla, že projde všechny součástky a ty, které mají nastavené *PREV* a *NEXT* zařadí do produkováného obvodu. To znemožňuje stavbu více obvodů zároveň, což nás ale příliš netrápí, pokud není Karlík nepořádný a nedělá tisíc věcí zároveň.

### 4 Poznámky

- **Cena obvodu**

Přesto, že by šla cena obvodu realizovat pomocí getteru, který by prošel všechny součástky, sečetl jejich cenu a toto vynásobil číslem 1.2, obvod si svou cenu pamatuje "natvrdo". To je motivováno snížením časové náročnosti při požadavku na cenu. Sice se do jisté míry jedná o duplicitu, ale dle mého názoru nemá cenu složitě počítat cenu při každém požadavku. Je mnohem jednodušší (a rychlejší!) vrátit cenu uloženou v lokálním atributu. Dalším argumentem pro toto řešení je, že cena obvodu se od okamžiku jeho tvorby nemění.

## 5 Závěr

Pracovní postup, který má celá hierarchie tříd umožňovat, zde nebudu detailněji rozepisovat. Pro jeho pochopení si stačí přečíst postup v zadání a aplikovat na něj změny v kapitole 2.