Objektové modelování

Sekvence

Kromě standardních matematických pojmů, jako jsou množina, n-tice nebo funkce, budeme potřebovat sekvence. Neformálně řečeno, sekvence je posloupnost blíže neurčeného počtu prvků. Od množin se sekvence liší především tím, že jejich prvky jsou uspořádány v nějakém pořadí, od uspořádaných n-tic zase tím, že nemají pevně danou délku.

Sekvence budeme zapisovat v úhlových závorkách (např. $\langle a, b, c\rangle$), k-tý prvek sekvence s budeme označovat $s(k)$ a pro prázdnou sekvenci budeme používat symbol $\langle\rangle$. Ke spojení dvou sekvencí budeme používat operátor $\cdot$ (např. $\langle a, b, c\rangle \cdot \langle d, e, f\rangle = \langle a, b, c, d, e, f\rangle$), délku sekvence získáme pomocí operátoru $length$ (např. $length(\langle a, b, c\rangle) = 3$). Množinu všech sekvencí prvků z množiny A budeme značit A^* (např. $\{a, b, c\}^* = \{\langle\rangle, \langle a\rangle, \langle b\rangle, \langle c\rangle, \langle a, a\rangle, \langle a, b\rangle, \langle a, c\rangle, \langle b, a\rangle, \dots\}$).

Základní množiny

Základní množiny jsou množiny, jejichž existenci a vlastnosti považujeme za dané. Nejčastěji jsou tyto množiny definovány ve specifikaci programovacího jazyka.

V našem formalismu budeme používat následující množiny:

- množina všech logických hodnot $bool = \{true, false\}$,
- množina všech čísel $num = \{\dots, -2, \dots, -1.5, \dots, 0, \dots, 0.1, \dots, 1, 2, \dots\}$ (o této množině často přemýšlíme jako o množině reálných čísel, v praxi však jde spíš o množinu všech čísel s plovoucí desetinnou čárkou reprezentovatelných na daném procesoru),
- množina všech znaků $char = \{a, b, \dots, ?, \$, \dots\}$ (pro naše účely je celkem jedno, jestli jde o znaky ASCII, Unicode nebo nějaké jiné),
- množina všech adres $addr = \{\#1, \#2, \#3, \dots\}$ (náš virtuální stroj bude mít neomezeně mnoho paměti, proto je množina $addr$ nekonečná),
- množina všech jmen atributů $fieldName = \{a, b, \dots, count, \dots\}$,
- množina všech jmen lokálních proměnných $localName = \{this, a, b, \dots\}$,
- množina všech jmen metod $methodName = \{a, b, \dots, f, \dots, getSize, \dots\}$ a
- množina všech jmen tříd $className = \{a, b, \dots, java.util.List, \dots\}$ (tato a tři předchozí množiny mohou mít po dvou navzájem neprázdný průnik; to nám nevadí, protože z kontextu ve zdrojovém kódu jde vždy jednoznačně určit, jestli daným jménem myslíme např. jméno atributu nebo metody).

```
class Pair {
    int first;
    int second;

    /* metody */
}
```

Obrázek 1: Relevantní část kódu třídy $Pair$.

Hodnoty

Množina všech hodnot je definována jako

$$value = bool \cup num \cup char \cup addr \cup \{null\}.$$

Neformálně: jakákoliv hodnota je buď logická hodnota, číslo, adresa nebo *null*. Množina *value* je významná tím, že do proměnných (lokálních, statických nebo atributů) jdou přiřazovat pouze hodnoty; definice množiny *value* nám proto mimo jiné říká, že do proměnných nejdou přiřazovat objekty, jen adresy objektů.

Objekty (instance tříd)

Množina všech objektů má o něco komplikovanější definici:

$$object = className \times (fieldName \rightarrow value).$$

Objekt modelujeme jako dvojici složenou ze jména třídy a funkce, která mapuje atributy daného objektu na jejich hodnoty. Instanci třídy $Pair$ (na obrázku 1) s atributem $first$ nastaveným na 2 a atributem *second* nastaveným na 3 proto můžeme zapsat jako $(Pair, \{(first, 2), (second, 3)\})$. Tato klasická notace je ale příliš nepřehledná, proto budeme používat jinou, ve které se ten samý objekt napíše jako $Pair(first = 2, second = 3)$.

Pro práci s objekty si nadefinujeme dvě významné funkce. Funkce $class : object \rightarrow className$ vrací název třídy zadaného objektu[1]

$$class(o) = \underset{1}{\mathrm{Pr}}(o)$$

a funkce $field : (object \times fieldName) \rightarrow value$ vrací k zadanému objektu hodnotu zadaného atributu

$$field(o, f) = (\underset{2}{\mathrm{Pr}}(o))(f).$$

Halda

Halda je místo v paměti, ve kterém jsou uloženy všechny objekty (ve většině objektových jazyků objekty nikde jinde než na haldě nejsou). Formálně je halda definovaná jako prvek množiny *heap*:

[1] Operátor Pr_1 vrací první projekci, tzn. první prvek uspořádané n-tice.

$$heap = addr \rightarrow object$$

Tato definice zohledňuje hlavní funkci haldy: pokud máme adresu a a haldu h, dokážeme získat i objekt $h(a)$ ležící na této adrese. Zároveň je dobré si uvědomit, že v naší definici zanedbáváme fakt, že v reálném stroji se objekt do jedné paměťové buňky obvykle nevejde.