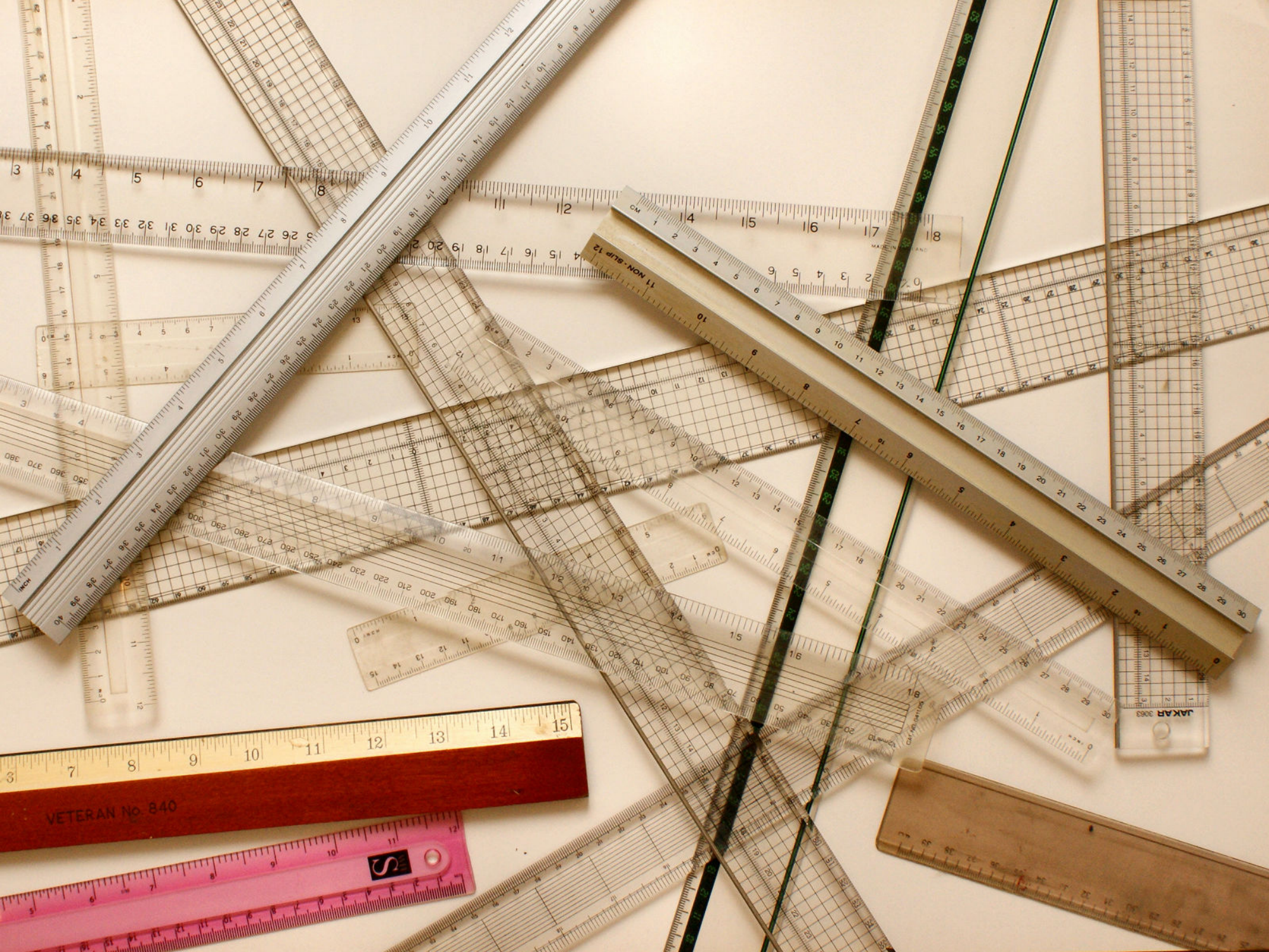
VETERAN No. 840

Minule

Objekty

- Dvojice složené ze jména třídy a mapování (funkce) jmen atributů na hodnoty
- Množina všech objektů $object = className \times (fieldName \to value)$
- Kvůli větší čitelnosti máme pro objekty upravenou notaci, např. $(Pair, \{(first, 2), (second, 3)\})$ zapisujeme jako $Pair(first = 2, second = 3)$
- Můžeme si nadefinovat přístupové metody $class : object \to className$ a $field : (object \times fieldName) \to value$
  - $class(o) = \mathrm{Pr}_1(o)$
  - $field(o, f) = (\mathrm{Pr}_2(o))(f)$

Haldy

- Halda je mapování z adres na objekty (zanedbáváme, že v reálném stroji se objekt do jedné paměťové buňky obvykle nevejde)
  $heap = addr \rightarrow object$
- Pokud máme haldu h a adresu a, pak objekt na této adrese získáme voláním $h(a)$

Zásobníky

- Náš zásobník je složitější než ten hardwarový; chceme zohlednit vnořená volání metod a vnořování bloků
- $stack = ((localName \rightarrow value)^*)^*$
- Kvůli čitelnosti opět používáme upravenou notaci
- Hodnotu (lokální) proměnné x na vrcholu zásobníku s získáme jako $s(1, i)(x)$, kde i je největší přirozené číslo takové, že $s(1, i)(x)$ je definováno

Zásobníky kódu

- Předpokládáme, že máme definovanou množinu příkazů *statement* (za chvíli uvidíme jak)
- Množina zásobníků kódu je $codeStack = (statement^*)^*$
- V zásobníku kódu *cs* se jako první vykoná příkaz $cs(1, 1)$

Oblasti statických proměnných

- Množina všech oblastí statických proměnných
  $global = (className \times fieldName) \rightarrow value$
- V oblasti statických proměnných g je hodnota proměnné $System.out$ rovna $g(System, out)$

Konfigurace virtuálního stroje

- Množina všech konfigurací virtuálního stroje
  $conf = heap \times stack \times codeStack \times global$

Uvažování o kódu

Dva typy tvrzení

$$(h, s, g) \vdash e_1 \mapsto^* e_2$$

$$(h, s, g) \xrightarrow{stmt}{}^* (h', s', g')$$

Příklad

Kód

```
class Node {
    int val;
    Node next;
    boolean contains(int v) {
        if (val == v) return true;
        else return next == null ? false : next.contains(v);
    }
}

class LinkedList {
    Node first;
    boolean isEmpty() { return first == null; }
    boolean contains(int v) {
        return first == null ? false : first.contains(v);
    }
}
```

Metoda `contains` pracuje správně

$$elems : addr \times heap \to num^*$$

$$elems(n, h) = \begin{cases} \langle\rangle \text{ pokud } n = null \\ \langle v \rangle \cdot elems(a, h) \text{ pokud } h(n) = Node(val = v, next = a) \end{cases}$$

$$(h, s, g) \vdash a.contains(v) \mapsto^* true$$

$$\Updownarrow$$

$$elems(field(h(a), first), h) = \langle z_1, \ldots, z_k \rangle \wedge v \in \{z_1, \ldots, z_k\}$$

Metoda `contains` pracuje správně (II)

$$(h, s, g) \vdash a.contains(v) \mapsto^* true$$

$$\Updownarrow$$

$$elems(field(h(a), first), h) = \langle z_1, \ldots, z_k \rangle \wedge v \in \{z_1, \ldots, z_k\}$$

$a.contains(v)$ volané nad haldou h vrátí *true*

tehdy a právě tehdy, pokud

sekvence hodnot v seznamu odkazovaném proměnnou a obsahuje hodnotu v