Jazyk Smalltalk/V

Ľubomír ŠESTÁK

Zverejnením popisu systému *Smalltalk-80** (začiatkom 80-tych rokov), ktorý je dodnes považovaný za štandard pre všetky smalltalkové systémy, sa začína éra veľkého rozmachu objektovo orientovaného programovania. *Smalltalk-80* je rozsiahly systém, ktorý sa prevádzkuje na náročnom technickom vybavení (často špecializovanom) a u nás sa vyskytuje zatiaľ ojedinele. *Smalltalk/V*** je jeho zjednodušená verzia pre bežné typy osobných počítačov, verzia *Smalltalk/V 286*** pre silnejšie konfigurácie počítačov umožňuje už vytvárať náročné aplikácie.

Pri smalltalkových systémoch nie je možné oddeliť programovací jazyk od programového prostredia. V doteraz známych programovacích systémoch tvorí jazyk a jeho prekladač základ, okolo ktorého je komplex podporných programov (editor, spájací program, ladiaci program, . . .), ktoré tvoria súčasť štandardného vybavenia operačného systému alebo sú súčasťou integrovaného programového prostredia. Okrem toho je spravidla k dispozícii knižnica podprogramov, ktorá rozširuje schopnosti "čistého" programovacieho jazyka. Jazyk je tým bohatší, čím je dodaná knižnica rozsiahlejšia.

Na systém Smalltalk možno hľadieť ako na obrovskú knižnicu "podprogramov", z ktorých niektoré vykonávajú činnosti všeobecne chápané ako funkcie editora, iné predstavujú súčasti ladiaceho prostriedku či prekladača. Časť tejto knižnice umožňuje vytváranie okien a menu, základných prvkov jednotného grafického rozhrania človek-stroj, časť prácu so súbormi, atď. Celý systém je jednotne organizovaný s rovnakým prístupom ku každej jeho časti, s maximálnym pohodlím pre užívateľa.

Pokúsime sa teraz bližšie pozrieť na jazykovú stránku systému Smalltalk. Dôraz pritom budeme klásť ani nie tak na syntax jazyka, ako na zvláštnosti prístupu k systému z hľadiska metodológie tvorby programu.

* *Smalltalk-80* je chránená značka firmy ParcPlace Systems.
** *Smalltalk/V* a *V286* sú chránené značky firmy Digitalk, Inc.

Program v systéme Smalltalk

Základným stavebným kameňom systému Smalltalk je objekt. Objekt je aktívna entita, je schopný konať. Objekty medzi sebou komunikujú tým, že si posielajú správy. Poslanie správy je základnou jednotkou aktivity v systéme. Programom môžme chápať postupnosť posielaní správ medzi objektami.

Objekty sú delené do skupín (tried) podľa istých charakteristík. Objekty z jednej triedy majú podobné vlastnosti a podobne sa správajú. Každý objekt vie k akej triede patrí, na akú skupinu správ vie reagovať, akú činnosť pritom vyvíja a čo vráti ako výsledok reakcie na poslanú správu. Objekt je uzavretým modulom. Pretože objekt predstavuje typ údajov, z tohto pohľadu môžme výpočet v systéme Smalltalk chápať ako údajovo riadený.

Nie je jednoduché charakterizovať stručne, čo je v takomto systéme programom. V systéme sú neustále k dispozícii tisíce objektov z niečo vyše sto tried. Tieto objekty sú schopné vykonávať tie činnosti, na zabezpečenie ktorých boli do systému začlenené. Podľa toho, aký problém riešime, využívame príslušnú časť z tejto plejády objektov, pričom ostatné objekty (často prevažná väčšina) zostáva nevyužitých.

Pri riešení problému pomocou smalltalkového systému je najdôležitejšie si uvedomiť, aké typy údajov budeme pri tom potrebovať. Preto je dôležité poznať celú škálu tried systému a vedieť, čo ktorá trieda dokáže zabezpečiť, na čo je určená. Často sa totiž stáva (najmä pri jednoduchých problémoch), že úloha sa dá triviálne vyriešiť využitím už existujúceho kódu. Napríklad problém utriedenie hodnôt môžme jednoducho vyriešiť

```
#(22 12 234 54 6 55 67 89 1 23 31 4 50 ...
  ... 32 123 17 32 110 21) asSortedCollection.
```

Tak isto však môžme utriediť aj zoznam mien, zoznam dátumov alebo telefónnych čísel.

Ak v systéme nie je taká trieda, ale je podobná, vytvoríme jej podtriedu, ktorej pridáme potrebné metódy (prípadne aj premenné), čím upravíme jej kvalitu na požadovaný stav. Mechaniz-

mus dedenia vlastností (premenných) a správania sa (metód) z nadtried do podtried dovoľuje, aby sa v systéme informácia nemusela duplikovať, ale bola vždy len raz na tom mieste, kde má maximálnu platnosť. Smalltalk/V má len jednoduché dedenie.

Príklad. Trieda *OrderedCollection* predstavuje obecný typ usporiadanej postupnosti hodnôt. Hodnoty je možné pridávať a operať z oboch koncov kolekcie. Ak chceme do systému dodať údajový typ zásobník, s klasickými operáciami, dodefinujeme triede OrderedCollection podtriedu *Zásobník* s metódami

```
push: anObject                    pop
    self addLast: anObject            ^self removeLast
```

pričom metódu isEmpty automaticky zdedí od svojej nadtriedy. Kvôli čistote programovania predefinujeme metódy addFirst: a removeFirst tak, aby vyhlásili chybu. Tým dokáže zásobník vykonať len tie činnosti, ktoré sú preň potrebné, a žiadne iné.

Ak v systéme nie je žiadny podobný údajový typ, ktorý by nám poslúžil ako východisko, vytvoríme si vlastnú triedu.

Len málokedy je program tvorený výlučne novou triedou (prípadne niekoľkými) a jej (ich) metódami. Väčšinou je rozptýlený v systéme do rôznych tried, pričom využíva pokiaľ je to len možné už existujúce triedy a ich metódy. Včlení sa tak do rovnorodej masy objektov, tvorí jej neoddeliteľnú súčasť a nie je možné určiť jeho "hranice" v systéme. Spustenie takéhoto programu spočíva vo vyhodnotení výrazu (akejsi "hlavnej procedúry"), v ktorom sa posielaním aktivačnej správy významnému objektu uvedie do chodu celá sieť posielaní správ, ktoré vyústia do výpočtu celého algoritmu riešenia daného problému.

Ak budeme riešiť napríklad problém spracovania jednoduchej agendy pracovníkov podniku, vystačíme pravdepodobne s jednou triedou, ktorú nazveme povedzme *Kartotéka*, a s nejakou globálnou premennou (povedzme *ZoznamPracovníkov*), ktorá bude obsahovať pole takýchto kartoték. Napísaním metód triede Kartotéka, ktoré umožnia manipuláciu s položkami kartoték, máme hotovú časť jednoduchej aplikácie hotovú. Táto časť môže stáť v systéme relatívne izolovane. (Utriedenie agendy napríklad podľa dátumu

narodenia pracovníkov môžeme vykonať príkazom

ZoznamPracovníkov asSortedCollection: [:a :b| a dátumNarodenia <= b dátumNarodenia]

keď predpokladáme, že trieda Kartotéka obsahuje v nejakej forme takýto údaj a metóda dátumNarodenia ho sprístupňuje.)

Ak však chceme vybaviť aplikáciu rozhraním vhodným pre užívateľa, je potrebné vybudovať nejaký typ okna, ktorý bude mať tvar štandardnej kartotéky, a kde budú políčky pripravené na vypĺňanie. Takýto formulár je užívateľovi prívetivý a nevyžaduje od neho prispôsobovanie sa počítaču. Pri tvorbe takéhoto okna už využijeme existujúce triedy *Form*, *Menu* a *Dispatcher* a včleníme našu aplikáciu hlbšie do systému.

Ak chceme urobiť agendu podrobnejšiu, už nevystačíme s jediným typom kartotéky. Začneme do systému vnášať "netradičné" typy údajov, napr. triedu *Pracovník* a jej podtriedy *VedeckýPracovník*, *TechnickoHospodárskyPracovník* a pod. Premenné v triede Pracovník budú charakterizovať údaje spoločné pre všetkých pracovníkov podniku (meno, vek, bydlisko, ...), kým napr. pre triedu VedeckýPracovník pribudne premenná *vedeckéPublikácie* alebo *navštívenéKonferencie*. A tak sa pomaly začne vytvárať zárodok toho, čo budeme môcť označiť ako smalltalkový program.

Pohľad na niekoľko tried systému Smalltalk/V

Základné vybavenie systému *Smalltalk/V* predstavuje asi 100 tried. V tomto príspevku nemáme dosť priestoru na prehľad celej hierarchie tried. Preto nebudeme spomínať skupiny tried, ktoré

- riadia správanie sa systému (triedy na vytváranie samotných tried a metód, kompilátor a pod.),
- vytvárajú interaktívne programovacie prostredie (rôzne typy okien a menu, browsery, celá bitová grafika, ...),
- manipulujú so súbormi a perifériami (triedy File, Directory, Stream a jej široká paleta podtried spracovania vstupu, ...).

Zmienime sa iba o troch skupinách tried, za ktorými sa skrývajú objekty, ktoré užívatelia prirodzene považujú za údajové typy.

Logicko-riadiace údaje

Trieda *Boolean* s podtriedami *True* a *False* a trieda *Context* predstavujú charakteristiku tých objektov, ktoré sú schopné zabezpečovať riadenie toku výpočtu programu. V syntaxi jazyka Smalltalk existuje konštrukcia, ktorá umožňuje uzavrieť časť kódu do bloku. Bloky môžu byť bez argumentov alebo môžu mať jeden či dva argumenty. Bloky sú inštanciami triedy Context.

Blok je možné priradiť do premennej alebo poslať ako argument v správe. Poslaním správy value (alebo value: či value: value:. pre argumenty) bloku je jeho kód aktivovaný. Tým je možné jednak pozdržať v prípade potreby výpočet a neskôr ho zase vyvolať alebo programovo zadefinovať činnosť, ktorá môže byť súčasťou správy poslanej inému objektu.

Inštancie tried True a False predstavujú elementárne hodnoty pravdy a nepravdy. Tieto objekty sú generované ako výsledky pravdivostných výrazov (napríklad x <= 5 alebo x between: 1 and: 10) a sú schopné reagovať nielen na správy typu and: alebo or:, ale aj na správy ako ifTrue:, ifTrue: ifFalse:, atď. Riadenie výpočtu sa tak dá efektívne zabezpečiť, keď napr. metóda ifTrue: ifFalse: je implementovaná rôzne v triedach

```
ifTrue: trueBlock ifFalse: falseBlock                    True
        ^ trueBlock value

ifTrue: trueBlock ifFalse: falseBlock                    False
        ^ falseBlock value
```

Podobne True takto implementuje známe metódy

```
or: aBlock              and: aBlock              not
    ^true                   ^aBlock value            ^false.
```

Blok vie reagovať na správy whileTrue: a whileFalse:, ktoré majú ako argument opäť blok. Inicializácia poľa môže vyzerať:

```
i := 1.
[i <= pole size]
    whileTrue: [pole at: i put: 0. i := i + 1]
```

Táto metóda je implementovaná ako

```
whileTrue: aBlock
    self value
        ifTrue: [aBlock value.
                 self whileTrue: aBlock].
```

Jednohodnotové údaje

Trieda *Magnitude* a jej podtriedy predstavujú základné jednohodnotové údajové typy. Trieda *Number* sa vetví na podtriedy *Float*, *Fraction* a *Integer*. V Smalltalku je možné pracovať so zlomkami, trieda Fraction zahŕňa objekty, ktoré majú čitateľa a menovateľa. Aritmetika zlomkov je prístupná v zdrojovom tvare. Celé čísla sa ešte delia na podtriedy *SmallInteger*, *LargePositiveInteger* a *LargeNegativeInteger*; v systéme nie je obmedzenie na veľkosť celého čísla, tá je limitovaná iba maximálnou veľkosťou akéhokoľvek objektu (64 Kbyte).

Celé čísla reagujú okrem štandardných numerických správ aj na niektoré správy riadenia toku výpočtu, napr.

```
4 timesRepeat: [Pero go: 100; turn: 90]
1 to: pole size do: [:index| pole at: index put: 0]
1 to: 100 by: 2 do: [:i | ...]
```

ktoré zabezpečujú jednoduché opakovanie výpočtu (argument je blok), parametrizované na intervale, prípadne s krokom.

Trieda *Association* predstavuje dvojicu (kľúč, hodnota), reprezentácia tried *Character*, *Date* a *Time* je zrejmá z názvu.

Kolekcie údajov

Trieda *Collection* a jej podtriedy združujú dátové typy s viacerými hodnotami. Triedy *Bag* a *Set* sú množinového typu. Každý prvok sa v *Set* môže nachádzať len raz, kým *Bag* povoľuje viacero výskytov toho istého prvku. Prístup k jednotlivým prvkom je možný iba ich hodnotou, zatiaľ čo v *IndexedCollection* aj indexom prvku. Dôležitou podtriedou triedy *Set* je trieda *Dictionary*, poskytujúca slovníkové operácie.

Trieda *IndexedCollection* sa člení na podtriedy *FixedSizeCollection* a *OrderedCollection*. Prvá má pevný počet prvkov a podtriedy *Array*, *Interval*, *String*, *Symbol* atď. Druhá obsahuje kolekcie s vopred neurčeným počtom prvkov (špeciálne jej podtrieda *SortedCollection* má tieto prvky utriedené).

Pri kolekcii sa nevyžaduje, aby obsahovala prvky toho istého "typu", (tej istej triedy), môžu to byť ľubovoľné objekty.

Na príklade kolekcií možno dokumentovať zdieľanie kódu dedením metód. Hlavná trieda *Collection* obsahuje metódy, na ktoré reagujú všetky druhy kolekcií. Patria sem najmä špeciálne generátory, ktoré veľmi uľahčujú prácu s kolekciami. Výrazy

```
#(1 2 3 4 5 6) select: [:each| each even]      vráti #(2 4 6)
#(1 2 3 4 5 6) reject: [:each| each even]      vráti #(1 3 5)
#(1 2 3 4 5 6) detect: [:each| each even]      vráti 2
#(1 2 3 4 5) collect: [:i | factorial]         vráti (1 2 6 24 120)
#(1 2 3 4 5) do: [:i | ...]
```

predstavujú mechanizmy automatického postupného preberania jednotlivých prvkov kolekcie a vyhodnotenia kódu (argument blok) nad nimi. Select vyberie z kolekcie tie prvky, pre ktoré kód v bloku dáva *true* a vráti ich ako novú kolekciu prvkov; reject funguje presne opačne; detect vráti prvý prvok, ktorý vyhovuje bloku; collect vráti kolekciu výsledkov vypočítaných pre jednotlivé prvky kolekcie; do: jednoducho zabezpečí výpočet bloku pre všetky prvky kolekcie.

Pretože jednotlivé podtriedy *Collection* sa odlišujú najmä v spôsobe pridávania a odstránenia prvku, každá z nich definuje svoje vlastné metódy **add:** a **remove:**.

Záverom

Objektovo-orientované programovanie je svojou filozofiou veľmi blízke človeku, lebo najviac odráža ľudské uvažovanie. Preto si získava takú popularitu. Systémy Smalltalk v tomto trende hrajú poprednú úlohu a zrejme nadalej aj budú.

Literatúra:

[1] BYTE – Smalltalk Issue, August 1991; May 1985, August 1986

[2] Goldberg, A. – Robson, D.: *Smalltalk-80: The Language and Its Implementation*, Reading, MA: Addison-Wesley, 1983

[3] Goldberg, A.: *Smalltalk-80: The Interactive Programming Environment*, Reading, MA: Addison-Wesley, 1984

[4] Krasner, G.: *Smalltalk-80: Bits of History, Words of Advice*, Reading, MA: Addison-Wesley, 1983

[5] *Smalltalk/V: Tutorial and Programming Handbook*, Digitalk Inc., 1986