ИТМО

Языки программирования. Семантика и система типов

Теоретическое задание. Тема 9

Бронников Егор

Следуя определениям императивных объектов с использованием открытой рекурсии, реализуйте подклассы `SetCounter` с возможностью сохранять текущее состояние и откатываться к (одному из) прежних состояний. В своей реализации вы можете использовать абстрации-заглушки или ссылки-заглушки.

Задание 1. Реализуйте класс `SingleBackupCounter` (функции `singleBackupCounterClass` и `newSingleBackupCounter`).

```
SingleBackupCounter = {
    get : Unit -> Nat,
    set : Nat -> Unit,
    inc : Unit -> Unit,
    backup : Unit -> Unit,
    restore : Unit -> Unit
}
```

Решение.

```
SingleBackupCounterRep = { x : Ref Nat, backup : Ref <nothing : Unit, just : T>}
```

```
SetCounter = {
    get : Unit -> Nat,
    set : Nat -> Unit,
    inc : Unit -> Unit
}
```

Смотреть продолжение на следующей странице.

$$
\begin{array}{l}
\mathit{let}\ \mathit{singleBackupCounter} = \\
\quad \lambda\, \mathit{rep} : \mathit{SingleBackupCounterRep}\,. \\
\quad\quad \lambda\, \mathit{self} : \mathit{SingleBackupCounter}. \\
\quad\quad\quad \mathit{let}\ \mathit{super} = \mathit{SetCounter}\ \mathit{rep}\ \mathit{self}\ \mathit{in} \\
\quad\quad\quad\quad \{\, \mathit{get} = \mathit{super.get}, \\
\quad\quad\quad\quad\ \ \mathit{set} = \mathit{super.set}, \\
\quad\quad\quad\quad\ \ \mathit{inc} = \mathit{super.inc}, \\
\quad\quad\quad\quad\ \ \mathit{backup} = \lambda\, \_ : \mathit{Unit}.\, \mathit{rep.backup} := < \mathit{just} = !\mathit{rep.x} >, \\
\quad\quad\quad\quad\ \ \mathit{restore} = \lambda\, \_ : \mathit{Unit}.\, \mathit{case}\ !\mathit{rep.backup}\ \mathit{of} \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad < \mathit{nothing} = \_ > \Rightarrow\ \mathit{unit}\ \mid \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad < \mathit{just} = x > \Rightarrow\ \mathit{rep.x} := x\ \} \\
\mathit{in} \\
\quad \mathit{let}\ \mathit{newSingleBackupCounter} = \\
\quad\quad \lambda\_ : \mathit{Unit}. \\
\quad\quad\quad \mathit{let}\ \mathit{rep} = \{x = \mathit{ref}\ 0,\ \mathit{backup} = \mathit{ref} < \mathit{nothing} = \mathit{unit} >\}\ \mathit{in} \\
\quad\quad\quad \mathit{fix}\ (\mathit{singleBackupCounter}\ \mathit{rep})
\end{array}
$$

Задание 2. Используйте списки, чтобы реализовать класс `BackupCounter`, который хранит стек сохранённых состояний (представленный списком). Повторный вызов `restore` должен быть поддержан (т.е. `restore` сбрасывает историю сохранённых состояний).

```
BackupCounter = {
    get : Unit -> Nat,
    set : Nat -> Unit,
    inc : Unit -> Unit,
    backup : Unit -> Unit,
    restore : Nat -> Unit // аргумент - индекс на стеке
}
```

Смотреть продолжение на следующей странице.

Решение.

```
BackupCounterRep = { x : Ref Nat, backup : Ref List Nat }
```

$$
\begin{array}{l}
letrec\ getByIndex = \\
\quad \lambda list : List\,Nat\,. \\
\quad \lambda index : Nat. \\
\qquad if\ \ iszero\ index\ \ then\ head\ list\ \ else\ \ getByIndex\,(tail\ list)\,(pred\ index) \\
\\
let\ backupCounter = \\
\quad \lambda rep : BackupCounterRep\,. \\
\qquad \lambda self : BackupCounter. \\
\qquad\quad let\ super = SetCounter\ rep\ self\ in \\
\qquad\qquad \{\, get = super.get, \\
\qquad\qquad\ \ set = super.set, \\
\qquad\qquad\ \ inc = super.inc, \\
\qquad\qquad\ \ backup = \lambda \_ : Unit.\, rep.backup\ :=\ cons\ !rep.x\ !rep.backup, \\
\qquad\qquad\ \ restore = \lambda i : Nat.\, super.set\ (getByIndex\ !rep.backup\ i)\} \\
in \\
\quad let\ newBackupCounter = \\
\qquad \lambda \_ : Unit. \\
\qquad\quad let\ rep = \{x = ref\ 0,\ backup = ref\ nil[Nat]\}\ in \\
\qquad\quad fix\ (backupCounter\ rep)
\end{array}
$$

Задание 3. Используйте списки, чтобы реализовать класс `AutoBackupCounter`, который автоматически сохраняет состояние при каждом `set`. Реализация `inc` не должна быть переопределена, но должна провоцировать сохранение (`backup`), поскольку метод `inc` в классе `SetCounter` реализован через вызов метода `set`.

```
AutoBackupCounter = {
    get : Unit -> Nat,
    set : Nat -> Unit,
    inc : Unit -> Unit,
    restore : Nat -> Unit // аргумент - индекс на стеке
}
```

Решение.

```
AutoBackupCounterRep = { x : Ref Nat, backup : Ref List Nat }
```

$$
\begin{array}{l}
\mathit{let}\ \mathit{autoBackupCounter} = \\
\quad \lambda\, \mathit{rep} : \mathit{AutoBackupCounterRep}\,. \\
\qquad \lambda\, \mathit{self} : \mathit{AutoBackupCounter}. \\
\qquad\quad \mathit{let}\ \mathit{super} = \mathit{SetCounter}\ \mathit{rep}\ \mathit{self}\ \mathit{in} \\
\qquad\qquad \{\, \mathit{get} = \mathit{super.get}, \\
\qquad\qquad\quad \mathit{set} = \lambda\, i : \mathit{Nat}.\, (\mathit{rep.backup} := \mathit{cons}\ !\mathit{rep.x}\ !\mathit{rep.backup};\ \mathit{super.set}\ i), \\
\qquad\qquad\quad \mathit{inc} = \mathit{super.inc}, \\
\qquad\qquad\quad \mathit{restore} = \lambda\, i : \mathit{Nat}.\, \mathit{super.set}\ (\mathit{getByIndex}\ !\mathit{rep.backup}\ i)\} \\
\mathit{in} \\
\quad \mathit{let}\ \mathit{newAutoBackupCounter} = \\
\qquad \lambda\_ : \mathit{Unit}. \\
\qquad\quad \mathit{let}\ \mathit{rep} = \{x = \mathit{ref}\ 0,\ \mathit{backup} = \mathit{ref}\ \mathit{nil}[\mathit{Nat}]\}\ \mathit{in} \\
\qquad\quad \mathit{fix}\ (\mathit{autoBackupCounter}\ \mathit{rep})
\end{array}
$$