5.2 Primitive values

Description

In this section, we further describe data-items that hold primitive values.

Floating point values

Data-items such as `balance` and `interestRate` are examples of data-items holding values of *float value types*, which represents a subset of the real numbers. These data-items are declared as follows:

```
balance: var float
interestRate: var float
```

The specification **`var float`** in the declarations of `balance` and `interestRate` specifies that these may hold floating point values, and that their type is `float`.

Examples of float values are: 100, 3.14, 0.56, -12.11, `13.5E7`, and `1.11E-7`. The `'E'` in `13.5E7` means that the value is 13.7 multiplied by 10^7.

The set of real numbers is infinite, and a computer can only represent a subset of the real values. This subset is called *floating point values*. Since only a subset can be represented, the evaluation of floating point expressions often imply that some values are rounded to a value that the computer can represent. For calculations of a large amount of numbers these roundings may lead to misleading and erroneous results. It is therefore recommended that people writing programs involving floating point calculations learn about how to avoid rounding errors.

The keyword **`var`** used in the declaration specifies that these data-items are variable, which means that they may be assigned different values during the execution of the program:

```
balance := 500
interestRate := 3.07
...
balance := balance + amount
interestRate := 2.9
...
```

It is possible to define a float constant using the keyword **`val`**:

```
pi: val 3.14
```

The default value for a `float` variable is zero (`0.0`).

Integer values

In section 3.4 `maxNoOfAccounts` and `noOfAccounts` of class `Customer` are examples of data-items holding integer values.

```
class Customer(name: var String):
   addr: var String
   email: var String
   maxNoOfAccounts: val 10
   noOfAccounts: var integer
   accounts: obj Array(maxNoOfAccounts, Account)
```

`Integer` is another example of a primitive value type. Examples of `Integer` values are 17, 111, -4, 0, and -1014.

As for `float`, an `Integer` may be declared as a constant or a variable. In the above example, `maxNoOfAccounts` is a constant and `noOfAccounts` is a variable.

The data-item `noOfAccounts` is an example of a variable that may hold different integers during the program execution,

while the data-item `maxNoOfAccounts` on the other hand is constant. i.e. holding the value `10` during the whole programm execution.

The default value for an `integer` variable is zero (`0`).

Boolean values

In section 3.4 the method `addAccount` of `Customer` has an if-then-else-statement:

```
if (noOfAccounts <= maxNoOfAccounts) :then
    accounts.put(acc):at[noOfAccounts]
:else
    console.print(“Cannot add Account”)
```

The expression `noOfAccounts <= maxNoOfAccounts` evaluates to one of the values `True` or `False`, which both are `Boolean` values. `Boolean` is another example of a primitive type and `True` and `False` are the only values of type `Boolean`. We may declare a data-time of type `Boolean` as follows:

```
b1: val True
b2: var Boolean
```

The data-item `b1` is a constant having the value `True` during the whole program execution. The data-item `b2` is a variable that may hold the values `True` or `False` at different points during the program execution. In the next example, `b2` is assigned the value `False`

```
b2 := False
```

We may add a boolean `isClosed` to class `Account` to represent whether or not the account is open or closed. An alternative may be to add a `closingDate` for the account and then make `isClosed` a method returning a boolean which is true if `closingDate <= thisDate` and false otherwise.

The default value for a `boolean` variable is the value `false`.

Character values

A `String` consists of characters from some character set like UniCode, Latin1, etc. The type `char` may be used to declare a data-item holding a value representing a character.

A character may be a letter, a digit, a special character, a blank/space or an end-of-line character. A character literal is typically written within single quotes like:

```
‘a’, ‘b’, ‘Q’, 'L’  – examples of letters
‘0’, 7?             – examples of digits
‘.’, ‘:’, ‘+’,      – examples of special characters
‘ ’                 – the blank character
```

An end-of-line character has no graphical form and is thus denoted as `‘\n’`. The character `‘\’` is a so-called escape character the implies that the next character defines a special character. The following are examples of characters defined using the escape character

```
‘\n’  – end-of-line
‘\t’  – tabulator
‘\”   – the quote character ‘
...
```

The following example shows the declaration of data-items of type `char` and assignment statements:

```
ch1: val ‘!’
ch2: var char
ch2 := ‘Q’
```

The data-item `ch1` is a a constant holding the exclamation mark character `'!'`. The data-item `ch2` is a variable of type `char`. It may hold different character values during the program execution. Execution of the assignment statement `ch2 := 'Q'` has the effect that `ch2` holds the character `'Q'`.

So far we have not seen examples using `char`, but such examples will be shown later in this book, see e.g. section .

The default value for a `char` variable is the zero char, which is a character that has the integer value zero.