2.4 The first program

A description written in a programming language is called a *computer program* or just a *program* but is also referred to as *source code* or simply *code*.

The description of `account_1010` in the previous section may be considered an example of a program, although it is not very useful since it is not used for any purpose.

Here we show an example of a program where money is deposited and withdrawn, represented as actions changing the balance of `account_1010`.

We may assign values to `balance` by an using assignment statements as shown here:

```
account_1010.balance := 350.56
```

The expression `account_1010.balance` access the attribute `balance` of the object `account_1010` and is an example of an *attribute accessor*.

Execution of this statement implies that `balance` in `account_1010` now holds the value 350.56.

The operator `:=` specifies that the variable `account_1010.balance` on the left-hand side gets a new value as specified by the *expression* 350.56 at the right-side of `:=`.

The ~~action/~~operation described by the assignment statement may represent a deposit of 350.56 on John Smiths account.

The assignment statement as shown above is just a program fragment and must be placed in an object in order to be executed. The next example shows such an object – our first example of a *program* carrying out operations:

```
myFirstProgram: obj
   account_1010: obj
      owner: val "John Smith"
      balance: var float
   account_1010.balance := 350.56
```

As said, a *program* is an object descriptor like the one for `myFirstProgram`. The opposite is not the case since an object descriptor is not necessarily a program. We return to this in chapter .

This program is also an object descriptor that consist of the following items:

- The declaration of the object `account_1010`.
- The assignment statement `account_1010.balance := 350.56`.

The object descriptor of `myFirstProgram` may be executed by a computer and this implies the following actions in that order:

- The object `myFirstProgram` is generated.
- The object `account_1010` is generated.
- The assignment `account_1010.balance := 350.56` is executed.

To illustrate what goes on during a program execution we use *snapshots* of (part of) the state of the program execution. The following figures illustrates what goes on at selected points of execution

Below the right columns contain *object diagrams* that shows the stage of the `account_1010` object at a given point of time during program execution. The left column shows the program text marked with a red arrow (►) that shows the next statement to be executed at the time of the snapshot. If the arrow is pointing beyond the last statement, the next action to be executed is termination of the object.

The first snapshot shows the situation when `account_1010` has been generated and before execution of the assignment:

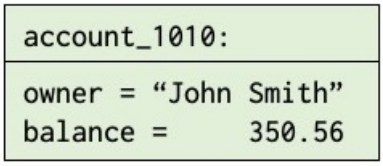
- The `owner` attribute of `account_1010` has the value `"John Smith"` since it is declared as constant using the keyword **`val`**.
- The `balance` attribute has the value 0 (zero), which is the default value for data-items of type `float`.

```
myFirstProgram: obj
   account_1010: obj
      owner: val "John Smith"
      balance: var float
►  account_1010.balance := 350.56
```

account_1010:
owner = “John Smith” balance = 0

The next snapshot shows the situation after execution of the assignment statement:

```
myFirstProgram: obj
   account_1010: obj
      owner: val "John Smith"
      balance: var float
►  account_1010.balance := 350.56
```



- The data-item `balance` now holds the value 350.56.
- The point of execution () is at the end of `myFirstProgram` pointing to an implicit action that when executed terminates execution of the object.