2.6 Method with parameter

Description

Next, we add a method for depositing an amount to the account, and since the amount to be deposited in most cases differ from time to time it is possible to define a method with a parameter defining the amount to be deposited.

In the next example, we add a method `deposit`, with a parameter `amount`:

```
account_1010: obj
   owner: val "John Smith"
   balance: var float
   interestRate: var float
   addInterest:
      balance := balance + (balance * interestRate) / 100
   deposit(amount: var float):
      balance := balance + amount
```

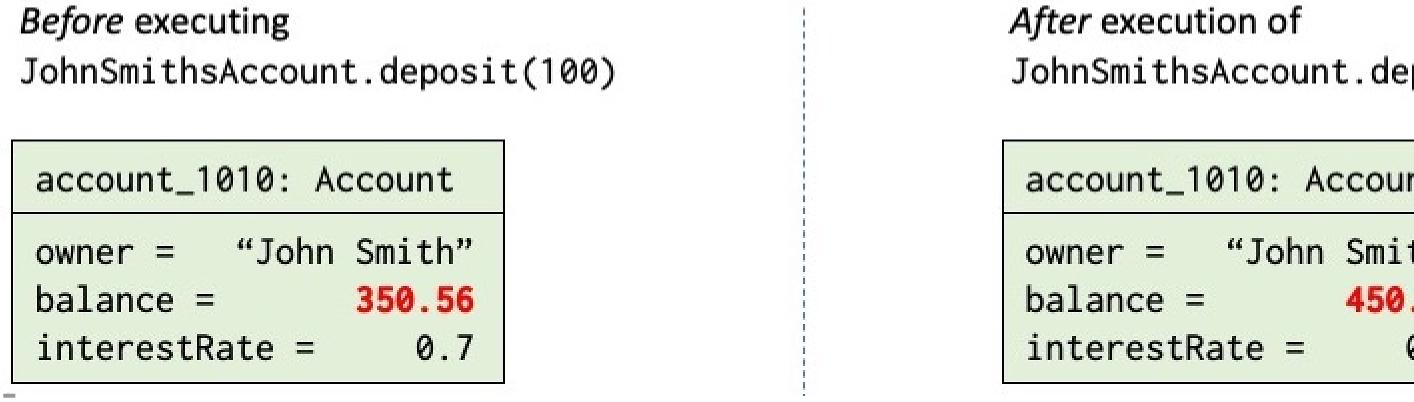
The variable `amount` is a *parameter* of `deposit`. A value for `amount` must be supplied when `deposit` is invoked as shown here:

```
account_1010.deposit(100)
```

The value supplied (here 100) is called the *actual paramete*r of `deposit` in contrast to `amount`, which is called the *formal parameter*. The term *parameter* is often used to mean the formal parameter whereas the term *argument* means the actual parameter.

The above statement is an example of a *method invocation*. It implies that an instance of `deposit` is generated and this implies that an activity as described by `deposit` is carried out by execution of the statement `balance := balance + amount`.

In this activity, the value 100 is supplied for `amount`. This means that `amount` holds the value 100 when the assignment statement of `deposit` is executed. The body of `deposit` adds `amount` to the current value of `balance`. If `balance` holds the value 350.56 before invocation of the method, it holds the value 450.56 after the execution of the invocation.



Method invocation

An instance of `deposit` is called a *method object* and is in many ways similar to an object like `account_1010` – we return to this later.

In general a method may have local data-items like `owner`, `balance` and `interestRate` of `account_1010`. The parameter `amount` is also an attribute (data-item) of the method object.