

3.1 Using class

Instead of describing each account object separately, we may describe a *class* that defines a template/pattern for accounts in general as in the following example:

```
class Account:
  owner: val "???"
  balance: var float
  interestRate: var float
  addInterest:
    balance := balance + (balance * interestRate) / 100
  deposit(amount: var float):
    balance := balance + amount
  withdraw(amount: var float) -> newB: var float:
    balance := balance - amount
    newB := balance
```

The keyword **class** specifies that we describe a class as opposed to an object (using **obj**).

We may use a class as a template to generate an arbitrary number of objects having the *structure* as described by the class. All objects generated according to class Account thus have the attributes that are described in class Account. Each Account object will thus have data-items owner and balance and methods addInterest, deposit and withdraw. An Account object is also referred to as an *instance* of Account.

Different Account objects will typically have different values of the data-items owner, balance and interestRate. The methods in class Account may be invoked on different Account objects.

We say that these objects are *Account objects* or *instances of Account*.

Before we show how to generate Account object using class Account, we have to fix a problem with the above description of class Account. The problem is that the value of owner is specified to be the string "???". John Smith and Liza Jones and other accounts have different values of owner.

To handle this, we may make owner be a parameter of class Account. The actual value of owner must then be supplied when we create objects from class Account. This is similar to how parameters works for methods as described in section . Class Account with a parameter is shown below:

```
class Account(owner: var String):
  balance: var float
  interestRate: var float
  addInterest:
    balance := balance + (balance * interestRate) / 100
  deposit(amount: var float):
    balance := balance + amount
  withdraw(amount: var float) -> newB: var float:
    balance := balance - amount
    newB := balance
```

Data-items specified within the parentheses are the parameters of the class, in this case **owner**. A parameter like **owner** is also a local data-item similar to **balance**. As for methods, the actual value of **owner** may be supplied later as a string literal, see below.

We use a *class-diagram* as shown here to illustrate a class - see chapter for an explanation of the form of a class-diagram.

Account
owner balance interestRate addInterest deposit withdraw

Class Account may be used to create different Account objects as in the example below:

```
account_1010: obj Account("John Smith")
account_1022: obj Account("Liza Jones")
account_1025: obj Account("Mary Pole")
```

The following diagram illustrates the state of these objects after they have been generated:

account_1010: Account	account_1022: Account	account_1025: Account
owner = "John Smith"	owner = "Liza Jones"	owner = "Mary Pole"
balance = 0	balance = 0	balance = 0
interestRate = 0	interestRate = 0	interestRate = 0

The objects described in the previous chapter 2 are called singular objects since there is only one of its kind for each of them. Objects generated using a class are called *class-defined* objects.

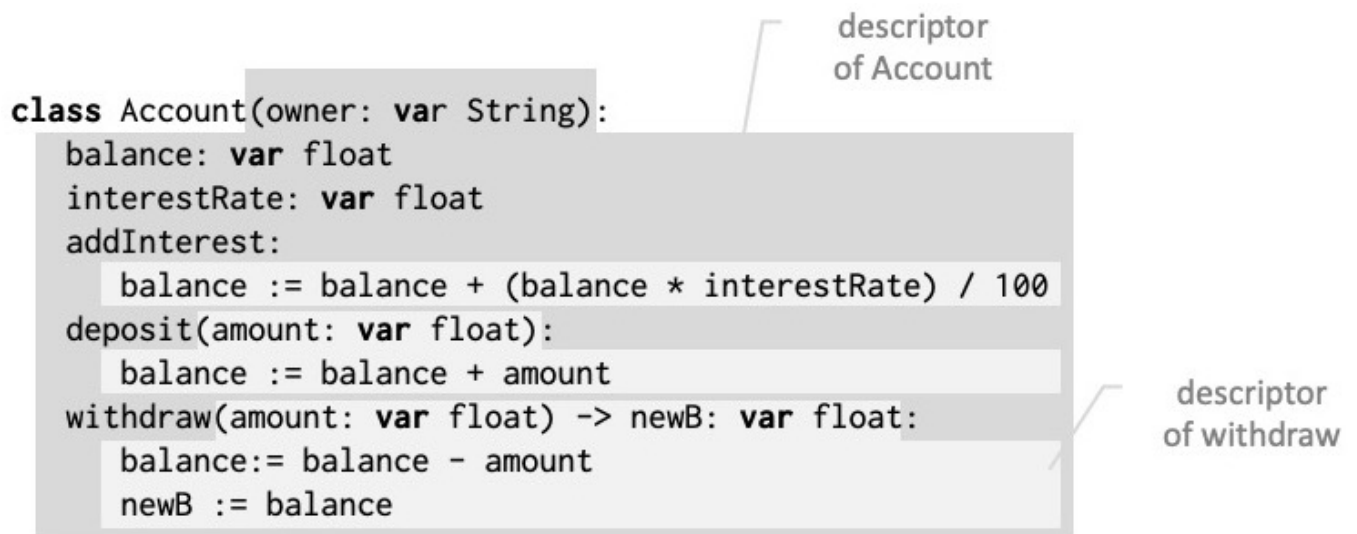
We of course need to be able to handle more than the three accounts shown above. We show how to do that in chapter .

The representation of an account by class Account as shown does not include all properties of an account needed for a real bank system. In the paper-based version as shown in section , we have an account number, which so far is encoded in the names of the Account-objects. In a proper model, it should be represented as an attribute of an Account-object. We leave this out to keep the examples small.

Terminology: object-descriptor again

A class is a name associated with an object-descriptor and in the figure below, the object-descriptor

for Account is enclosed in a box. An object-descriptor may thus be used to describe a singular object and/or a *class-defined object* as shown in this chapter.



As can be seen, the object-descriptor includes the parameter of class Account.