2.7 Method returning a value

For completeness, we add a method for withdrawing an amount from the account. The name of the method is `withdraw` and it has a parameter `amount` being the amount to be withdrawn from the account. In addition to withdrawing an amount, the method returns the updated balance to the invoker of the method.

```
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
   withdraw(amount: var float) -> newB: var float:
      balance := balance - amount
      newB := balance
```

The arrow (`->`) in `-> newB: var float` specifies that the value of the data-item `newB` is returned by `withdraw`.

As can be seen, `withdraw` contains two assignment statements. First `amount` is deducted from `balance` (`balance := balance - amount`) and next, the updated value of `balance` is assigned to `newB`, which is returned as the value of `withdraw`.

The method `withdraw` may be invoked as shown here:

```
newBalance: var float
newBalance := account_1010.withdraw(140)
```

1. We have declared a variable `newBalance` of type `float`.
2. The assignment statement `newBalance := account_1010.withdraw(140)` is executed.
3. The method `withdraw` of `account_1010` is executed with the parameter `amount` holding the value `140`.
4. Assuming that `balance` is `450.56` as in the example above, the new value of `balance` will then be `310.56`.
5. This new value of `balance` is returned by `withdraw`.
6. After the execution of `newBalance := account_1010.withdraw(140)`, the variable `newBalance` holds the value `310.56`.

Suppose we would like to prevent a withdraw if there is not enough money on the account. We can do this by using an `if:then` statement to test if `balance` is greater than the `amount` to be withdrawn:

```
   withdraw(amount: var float) -> newB: var float:
      if (balance >= amount) :then
```

```
      balance := balance - amount
   newB := balance
```

The `if:then` statement has a condition `(balance >= about)`, which is evaluated and results in a Boolean value that is either true or false. The operator `>=` means greater than or equal. If `balance` is greater than or equal `amount` then `(balance >= amount)` evaluates to true otherwise it evaluates to false.

If the condition evaluates to true, then and only then is the part after `:then` executed – in this case the statement `balance := balance - amount`.

The `if:then` statement thus ensures that a `withdraw` is only possible if there is sufficient money on the account. With the above solution, `withdraw` only does not update `balance` – in practice one has to take some action to inform about the `withdraw` not being possible. We return to that in section .