7.3 Parameter transfer and return value

For a class instantiation, an object is created and for a method invocation, a method object is created.

Then the actual parameters of the invocation/instantiation are evaluated and transferred to the corresponding data-items of the method object/object. When the method has been executed, a possible value may be returned.

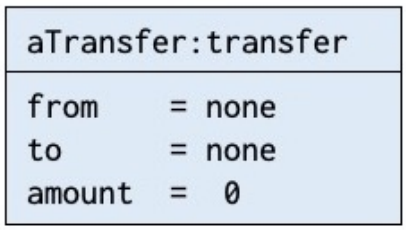
Parameter transfer

The transfer of parameters corresponds to assignment of the actual parameters to the formal parameters. Consider the following method invocation:

```
transfer(account_1010,account_1022, 218)
```

The invocation consists of the following steps:

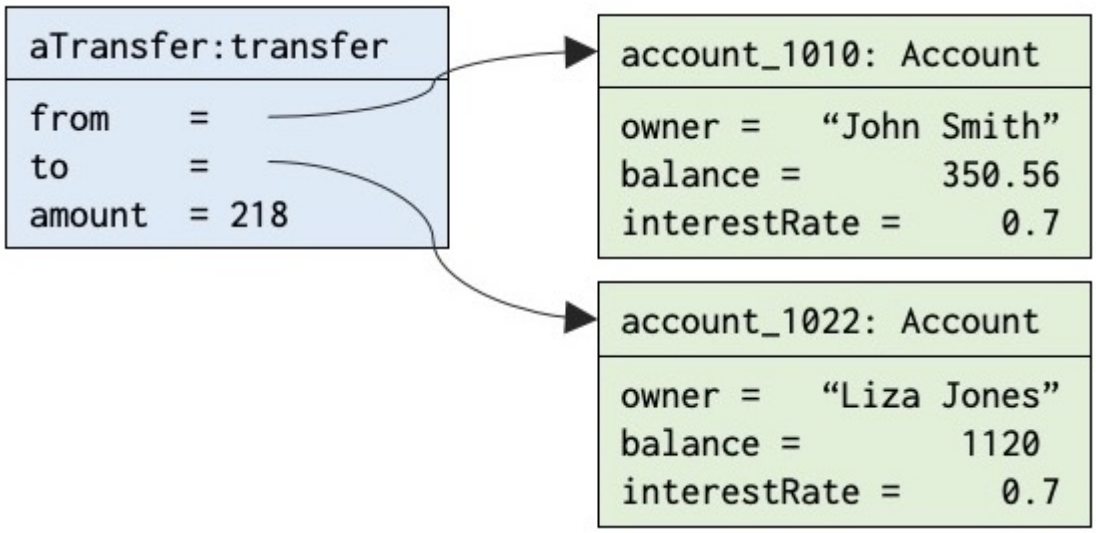
1. A `transfer`-object is generated – assume that `aTransfer` refers to this object. The situation is then as shown in the snapshot where the reference variables `from` and to have the initial value `none` and `amount` has the initial value 0 (zero).



2. The actual parameters are then assigned to the corresponding data-items in `aTransfer`:

```
aTransfer.from := account_1010aTransfer.to := account_1022aTransfer.amount := 218
```

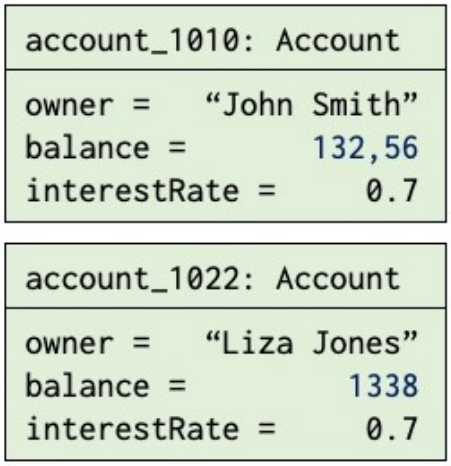
The situation is then as shown in the snapshot where the `from` refers to `account_1010`, `to` refers to `account_1022`, and `amount` has the value 218.



3. The statements of `aTransfer` are executed:

```
from.withdraw(amount)
to.deposit(amount)
```

The situation is then as shown in the snapshot.



In the example, we have two types of parameters:

`Amount` is a a *value parameter* of type integer. A value type is copied to the data-item of the method object.

`From` and `to` are `reference parameters`. For a reference parameter the reference is assigned to the data-item in the method object.

For an elaboration of the difference between value and object and thus value assign and reference

assign, see section .

There is a third kind of parameter being virtual methods and/or virtual classes. In section . we saw an example of a virtual class as a parameter:class `ElmType` as a virtual class parameter of class `Set`. In section , we show examples of virtual method parameters.

Parameter transfer for a class instantiation, takes place in a similar way.

Return value

As mentioned, a method invocation may return a value being computed by the method. This is e.g. the case for the `withdraw` method of `Account`:

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

The value to be returned is defined by the clause `-> newB: var float`. It defines a data-item `newB`, which holds the value to be returned. In `withdraw`, the statement `newB := balance` assigns a value to `newB`.

The return value of `withdraw` may e.g. be used in as shown here:

```
anAmount := anAccount.withdraw(300)
```