8.3 Virtual methods

In this section, we will introduce virtual methods. Virtual methods may be used to specify partial methods that may be further extended in subclasses.

In the next example, we define `withdraw` of `Account` as a virtual method:

```
class Account(owner: ref Customer):
   ...
   witdraw(amount: var float) -> newB: var float:<
      inner(witdraw)
      newB := balance
```

The symbol `':<'` specifies that `withdraw` is a virtual method.

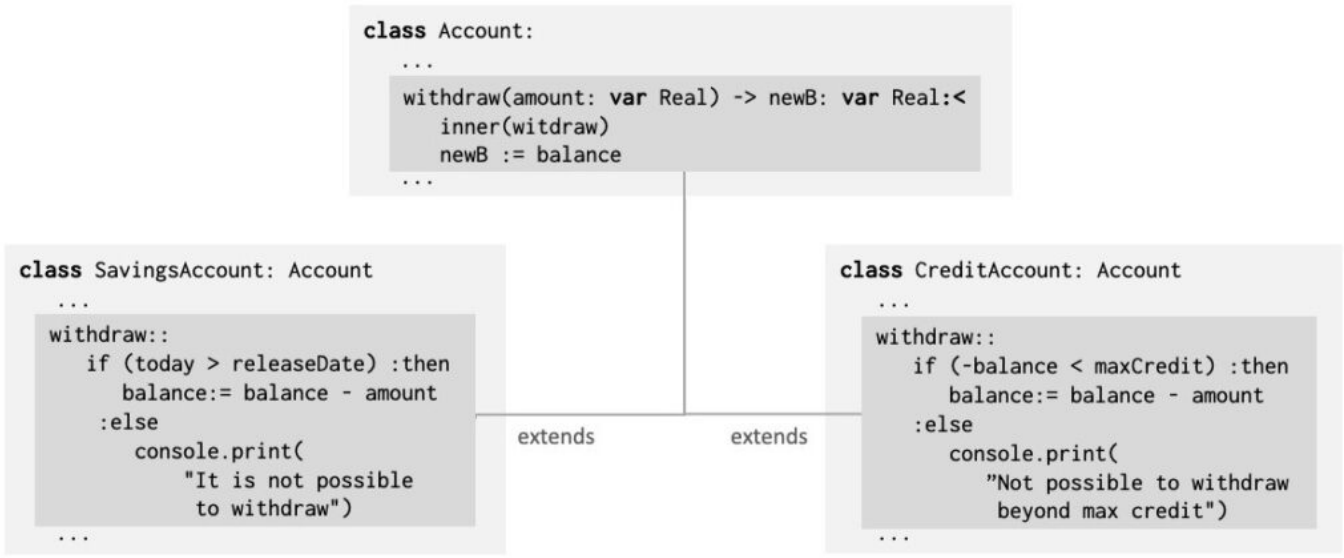
The statement part of `withdraw` consists of `inner` followed by `newB := balance`.

The statement `inner` specifies that when executed, possible statements in a further extension of withdraw in a subclass of `Account` will be executed.

In the next example, we show how to extend `withdraw` in a class `SavingsAccount`:

```
class SavingsAccount: Account
   ...
   withdraw ::
      if (today > releaseDate) :then
         balance:= balance - amount
      :else
         console.print("It is not possible to withdraw")
```

The symbol `'::'` specifies that `withdraw` is an *extension* of `withdraw` in the superclass `Account` of `SavingsAccount`.



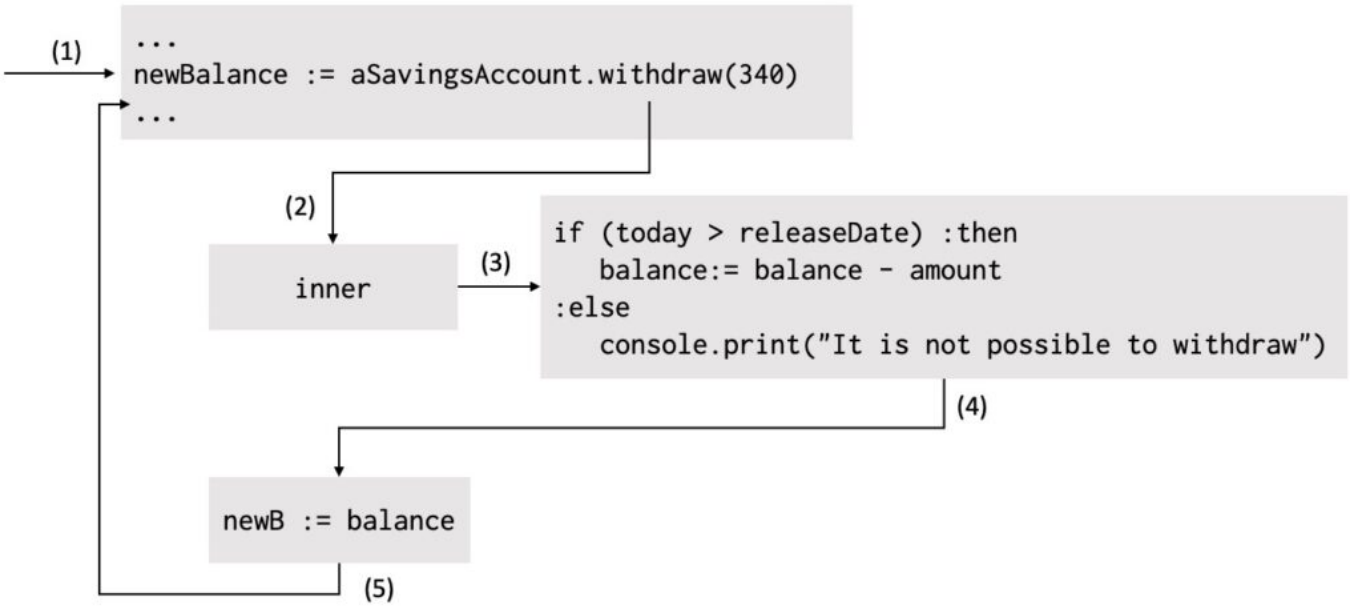
In the following example, we invoke `withdraw` on a `SavingsAccount`:

```
aSavingsAccount: obj SavingsAccount(JonhSmith)
newBalance: var float
newBalance := aSavingsAccount.withdraw(340)
```

Execution of `aSavingsAccount.withdraw(340)` takes place as follows:

1. Execution starts by execution of the statements in `withdraw` as defined in `Account`.
2. The first statement to execute is `inner`.
3. Execution of `inner` implies that the statements of withdraw in `SavingsAccount` are executed.
4. When the statements in `SavingsAccount` have been executed, control returns to the statements after `inner` in `Account`.
5. This implies that `newB := balance` is executed
6. Finally execution of `withdraw` is completed.

The figure below illustrates the execution of `aSavingsAccount.withdraw(340)`. The numbers show the order of execution of the involved statements:



Sequence of actions for the call SavingsAccount.withdraw(340)

We may extend `withdraw` in a similar way in `CreditAccount`:

```
class CreditAccount: Account
   ...
   withdraw ::
      if (-balance < maxCredit) :then
         balance := balance - amount
      :else
         console.print("Not possible to withdraw beyond max credit")
```

Invocation of `withdraw` on a `CreditAccount` object implies that execution of `inner` will execute the statements in `withdraw` in `SavingsAccount`.

A virtual print method

We now add one more virtual method to `Account` and its subclasses in the form of a print method that prints the status of a given account on the console.

Part of the print of status is the same for all accounts whereas other parts are special for the subclasses.

```
class Account(owner: ref Customer):
   ...
   createStatement:<
      stmt: var String
      stmt := "Account statement for " + owner.name + '\n'
              + "The account is a "
      inner(Account)
      stmt := stmt + "The balance is: " + balance + '\n'
   print:
      console.print(createStatement)
   sendEmail:
      owner.email(createStatement)

class SavingsAccount: Account
   ...
   createStatement::
       stmt := stmt + "savings account." +
               "\nRelease date is " + releaseDate.asText
```