8.1.1 Common superclass

Description

As mentioned, both types of accounts have a `withdraw` method, but the statement part of of these two methods are different.

We may define the common attributes in a general class `Account` like the one we have defined in the previous chapters:

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

We have included a partial description of a `withdraw` method to indicate that accounts in general have a `withdraw` method that returns the value of `balance`. In section below, we introduce a language mechanism called *virtual method* that makes it possible to describe such partial methods.

We may the use class `Account` in the description of the more specific accounts. For class `SavingsAccount` this looks as follows:

```
class SavingsAccount: Account
   releaseDate: var Date
   withdraw(amount: var float) -> newB: var
 float:
      if (today > releaseDate) :then
         balance := balance - amount
      :else
         console.print("It is not possible to withdraw")
      newB := balance
    newReleaseDate(newDate: var Date, newInterest: var float):
      releaseDate := newDate
      interest := newInterest
```

Class `SavingsAccount` is described as a *subclass* of class `Account` — specified by `Account` following the `':'` (colon) in class `SavingsAccount`, like: **`class`** `SavingsAccount: Account -"-`.

Making `SavingsAccount` a subclass of `Account` implies that all the attributes defined in class `Account` are also attributes of `SavingsAccount`. A `SavingsAccount` objects thus have attributes `owner`, `balance`, `interest` and `deposit`. Only the attributes special for a savings account need to be described in class `SavingsAccount`.

We note again, that a `withdraw` method is defined in both class `Account` and class `SavingsAccount` and we will show how to specify this using a virtual method in the next section.

We may make a similar description of class `CreditAccount`:

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

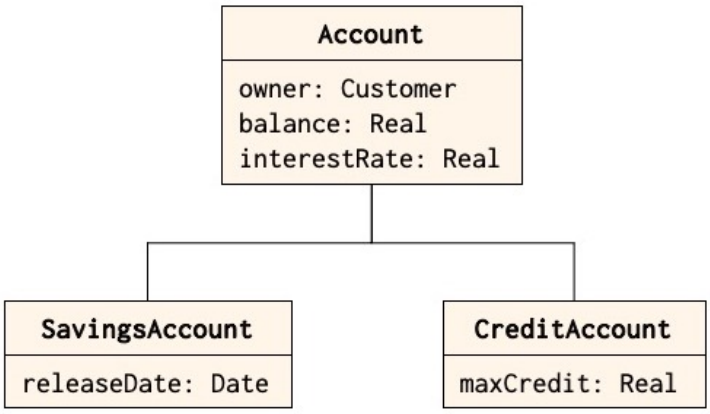
```
changeCredit(newMax: var float, newInterest: var float):
    maxCredit := newCredit
    interest := newInterest
```

Class `CreditAccount` is also described as a subclass of `Account` and only the attributes that are special for a credit account are specified.

As mentioned, `SavingsAccount` and `CreditAccount` are subclasses of `Account`. `Account` is a *superclass* of `SavingsAccount` and `CreditAccount`.

The figure below illustrates the sub/super-class relationships between the classes `Account`, `SavingsAccount` and `CreditAccount`. As used before, a rectangle with a yellow background represents a class and an arrow is used to point from a subclass to its superclass. We do not list the attributes from a superclass in a subclass – only attributes added in the subclass are shown.

We use a diagram of this form to illustrate the realtionships between a class, its subclasses and possible superclass. See chapter .



- In general the term *superclass* refers to a class from which other classes are derived.
- The direct superclass is the class from which the class/singular object is explicitly derived as specified in the object descriptor for the class/singular object – see the extended definition of object descriptor in section .
- The term *direct subclass* is an immediate subclass of a given class.

`Account` is a direct superclass of `SavingsAccount` and `CreditAccount`, and these are both direct subclasses of `Account`.

It is in general possible to create as many subclasses as needed to represent a given concept classification hierarchy. In the above examples we may have several subclass of class `Account` and we may also have subclasses of `SavingsAccount` and `CreditAccount`. We give several examples of this in the following.