4.1 Class Set

Description

A collection is a set, which is an object that may hold a set of objects. The `Set` collection is defined as follows:

```
class Set(class ElmType:< Object):
   insert(e: ref ElmType): :::
   has(e: ref ElmType): :::
   remove(e: ref ElmType): :::
   :::
```

Class `Set` has a parameter, `ElmtType`, which is a class name. The `ElmType` parameter specifies the type of the elements that may be inserted into the `Set`.

The description of class Set as shown here is a simplified description of the class. In a later chapter, we show a more complete description of the Set class.

Class `Set` has three methods, `insert`, `has` and `remove`, each having a parameter e being a reference to an object of type `ElmType` – the class parameter of class `Set`.

Note! We have not specified the details of the three methods — the colons (:::) represents the details not shown. The three methods actually define what is called the *interface* of class `Set` — the interface of a class (or object) are the attributes (here methods) that may be used to access the attributes of a given instance of the class. The details are often referred to as the *implementation* of the class.

It is a fundamental principle of programming to define classes and objects and only expose their interfaces. The advantage of this is that as long as the interface of a given class is not changed, it is possible to change the implementation of the class without affecting the use of the class, which may appear in many places in the code.

We may use class `Set` to represent our `Account`-file in the following way:

```
theAccountsFile: obj Set(Account)
```

We may insert a new account into `theAccountsFile` as follows:

```
aCustomer: ref Customer
anAccount: ref Account
aCustomer:= Customer("Linda Berry","England","linda@google.com")
anAccount := Account(aCustomer)
theAccountsFile.insert(anAccount)
```

We do not need the two variables `aCustomer`, and `anAccount`, since we may inline the expressions for `Customer` and `Account` in the `insert` expression as in:

```
theAccountsFile.insert(
   Account(
      Customer("Linda Berry","England","linda@google.com")))
```