11.2 Class-constrained virtual class

Description

In chapter , we introduced class `Set`, which has a class `ElmType` as a parameter. Class `ElmType` is an example of a *class-constrained* virtual class.

We first show how to define class `ElmType` as a local attribute and not a parameter to illustrate the similarity with respect to virtual class `TravelInfo` defined in the previous section. To be able to distinguish from the `Set` defined in chapter , the class below is called `ObjectSet`:

```
class ObjectSet:
   class ElmType:< Object
   insert(E: ref ElmType): ...
   has(E: ref ElmType): ...
   remove(E: ref ElmType): ...
   ...
```

The declaration of class `ElmType` here specifies a local class attribute like class `TravelInfo` in the previous section. The difference is that class `ElmType` is constrained by a class, in this case the pre-defined class `Object`. The constraint of class `ElmType` specifies that it must be a subclass of `Object`.

We may extend the specification of `ElmType` in subclasses of `ObjectSet`. Here we define a class `AccountSet`:

```
class AccountSet: ObjectSet
   class ElmType ::< Account
```

With this binding/extension of `ElmType` to `Account`, instances of `AccountSet` may only contain objects of type `Account`.

```
as: obj AccountSet
r: obj Customer
a: obj Account
as.insert(r)  -- illegal
as.insert(a)  -- legal
```

We may further extend `ElmType` in subclasses of `AccountSet`:

```
class SavingsAccountSet: AccountSet
   class ElmType::< SavingsAccount
```

Instances of `SavingsAccountSet` may only contain objects of type `SavingsAccount`.

Virtual classes as parameters

As mentioned, class `Set` as introduced in chapter has class `ElmType` as a parameter.

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

A parameter is in general similar to a local attribute – the difference being that a value (an actual parameter) for the parameter must be supplied when the class (or method) is instantiated. You may also supply the actual parameter when a class or method is used as a superclass or supermethod respectively.

Declaring class `ElmType` as a parameter thus works in the same way as for the `ObjectSet` with a local virtual class `ElmType`.

In chapter , we showed how to create an instance of class `Set` and supplying an actual parameter for `element`:

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

I vores TODO står at vi mangler: Term for supplying argument to superclass.
Behøver vi det -jvnf teksten nedenfor

In this case, the specification of the virtual class is extended by supplying an actual parameter for `ElmType`. This is actually a binding of `ElmType` to `Account`. We may use class `ObjectSet` to make a similar object:

```
theAccountsFileX: obj ObjectSet
   class ElmType::< Account
   ...
```

Here the binding of `ElmType` to `Account` also extends the description of `ElmType` in a sub-descriptor of class `ObjectSet`.

We should have a section summarizing virtual classes.