14.1 Visibility mechanisms

Description

Here we introduce some language mechanisms that may be used to separate the representative elements from the non-representative elements. The language elements are called *access modifiers* and they may specify the accessibility of attributes of an object. Access modifiers define the scope and visibility of these attributes in the code. This means that the programmer may specify to what extent a local data-item, method or class may be accessed in other parts of the code.

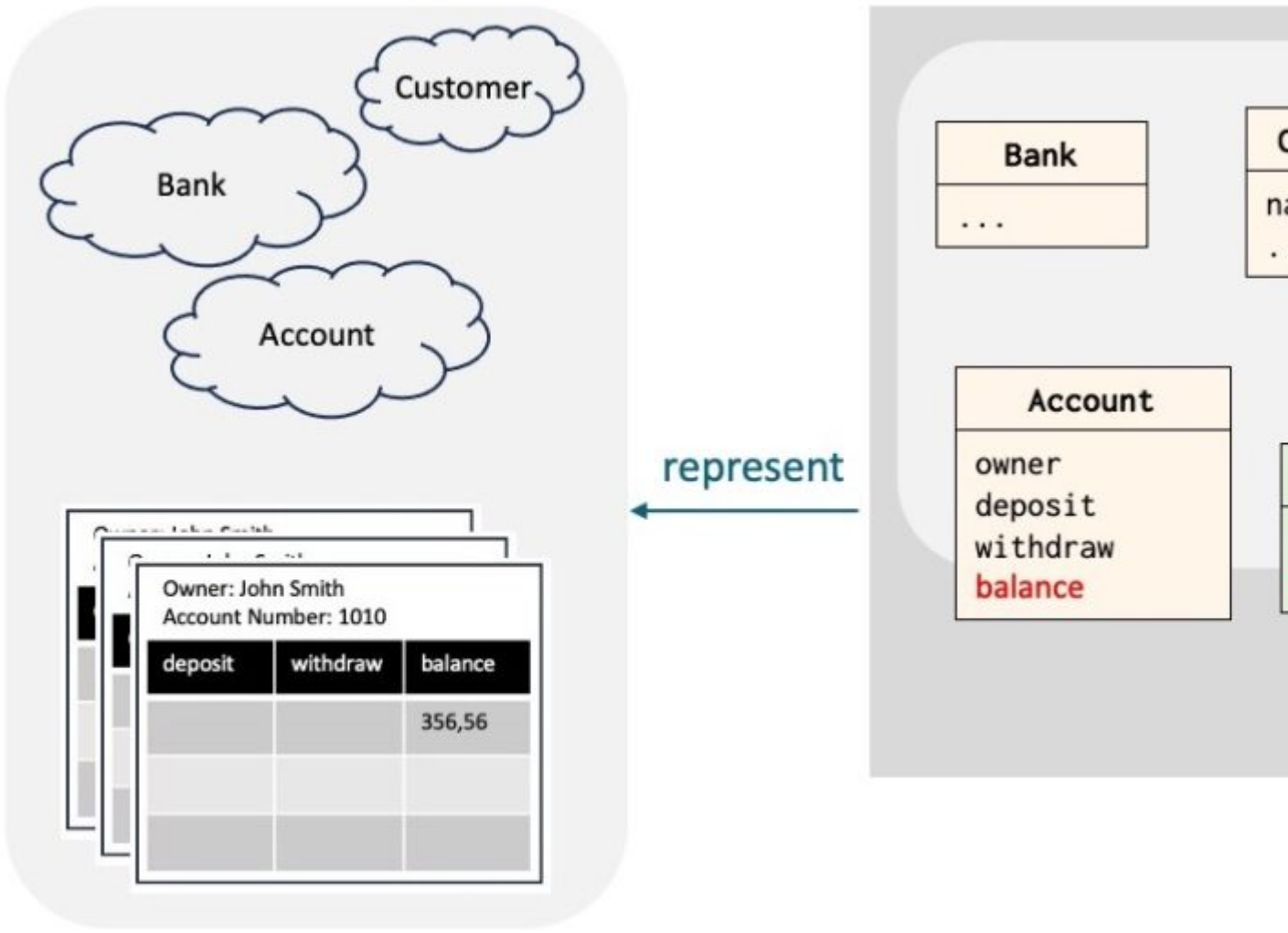
The most common types of access modifiers are *public*, *private*, and *protected*, but some languages support more as is the case for qBeta (see below). Each access modifier provides different levels of access.

%private

As a first example, we will show how to use the access modifier `%private` to protect `balance` and `interestRate` of `Account` objects:

```
class Account(owner: var String):
   addInterest: -"-
   deposit(amount: var float): -"-
   withdraw(amount: var float) -> newB: var float: -"-
   getBalance -> bal: var float: -"-
   getInterestRate -> iRate: var float: ...
   setInterestRate(iRate: var float): ...
   %private
   balance: var float
   interestRate: var float
```

The access modifier `%private` before the declarations of `balance` and `interestRate` specfies that `balance` and `interestRate` may only be accessed within class `Account`.



Representative and non-representative parts of models in the bank domain

%protected

When using `%private` in class `Account` as above, no other classes or objects may acces `balance` and `interestRate`. However, this may be inconvenient for subclasses of `Account` like `SavingsAccount`, where we may add methods that manipulate these attributes.

The access modifier `%protected` may be used to specify that some attributes may only be accessed within a class and its subclasses.

We may thus revise class `Account` as follows:

```
class Account:
   -"-
   %protected
   balance: var float
   interestRate: var float
class SavingsAccout: Account
   -- balance and interestRate may now be accessed here
   -"-
aClerk: obj
   -- balance and interestRate may not be accessed
   -"-
```

%public

For completeness, the access modifier `%public` may be used to specify that some attributes are visible outside the object-descriptor. However, `%public` is *default* in qBeta – if no access modifier is specified for some attributes they are visible outside the object-descriptor. Sometimes it may be useful to use `%public` if one prefer to have e.g. the private attributes appearing before the ones that are public. This is the case for the version of class `Account` below:

```
class Account:
   %protected
   balance: var float
   interestRate: var float
   %public
   addInterest: -"-
   -"-
```

Here `balance` and `interestRate` are declared in the beginning as `%protected` followed by the methods of the class – these are thus explicitly declared as `%public`.

%package_boundary and %package

This section is a reference section, which may be skipped during a first reading. So far we have no examples using the package access modifiers, but one will be given in section .

A *package* is a self-contained collection of modules that consist of a top module with local modules where each local module may contain local modules, etc. The access modifier `%package_boundary` defines the top module of the package – being the module where `%package_boundary` is placed.

The access modifier `%package` may be used to specify that some attributes are only visible in modules within the current package where the current package is defined by the nearest enclosing module with a `%package_boundary` modifier and its local modules, etc.

Summary of access modifiers

In this section we summarize the description of access modifiers – we use the following sketchy example:

```
class T:
   ...
   %public
   x: var integer
   m(...):
      ...
   %private
   y: var integer
   l: obj Set(integer)
   ...
   %protected
   z: var integer
   s: ref Person
```

```
   ...
   %domain
   v: var real
   foo:
      ...
   ...
```

The class T is specified using the access modifiers %public, %private, %protected, and %domain.

An access modifier applies to all the attributes following the modifier, up to the next access modifier.

The next example shows a usage of class T where we have a T-object:

```
r: obj T
-- The access modifiers specify if we may acces the attributes
-- r.x, r.y, r.z, r.v, ...
```

In the next example, we have a subclass TT of T:

```
class TT: T
   -- Here the access modifiers above also specify if we may access
   -- x, y, z, v, ...
```

Public

The access modifier %public specifies that one or more attributes are accessible from outside the object-descriptor where it is declared.

In the above examples x and m may be accessed via r.x, r.m(...) and as x an m(...) in the subclass TT of T.

Private

The access modifier %private specifies that one or more attributes may only be accessed within the object-descriptor where they are declared.

This means that r.y and r.l are not legal and that y and l may not be accessed in the subclass TT.

Protected

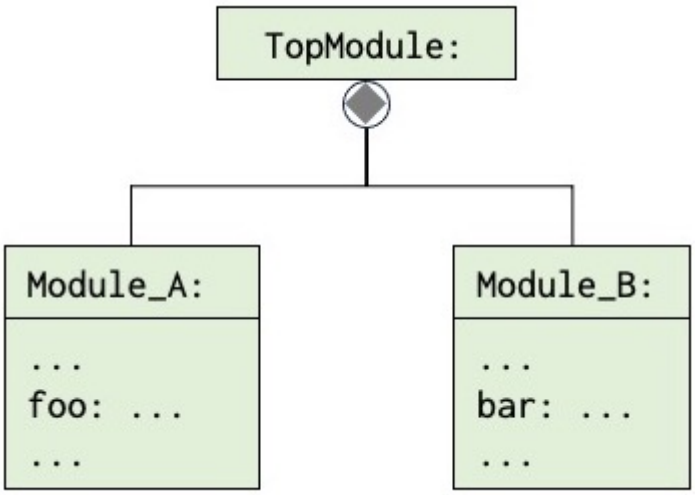
The access modifier %protected is like %private except that the attributes may be accessed in subclasses of the class where they are declared.

This means that z and s may not be accessed via r.z and r.s but may be accessed in all subclasses of T.

Package_boundary and package

The access modifier %package_boundary specifies the top module of a package and %package specifies that one or more attributes are visible in the enclosing package.

```
TopModule: obj
   %package_boundary
   ...
   Module_A: obj
      ...
      %package
      foo: ...
      ...
   Module_B: obj
      ...
      %package
      bar: ...
      ...
```



The attributes `foo` and `bar` are visible in `TopModule`, `Module_A`, and `Module_B`.

Default

If no access modifier is specified, the default is `%public`.