

## 9.3 Loop statements

A loop statement specifies that one or more statements are executed a number of times and as mentioned, cycle and for:to:repeat are examples of loop statements.

### Defining for:to:repeat

The for:to:repeat is defined as follows:

```
for(first: var integer):to(last: var integer):repeat{repeat:< object}:
  inx: var integer
  inx := first
  doIt: do
    if (inx <= last) :then
      repeat
        inx := inx + 1
        restart(doIt)
```

A for:to:repeat has the following structure:

- The name of the control method is for:to:repeat.
- It has three parameters: first, last and repeat.
- It has a local integer variable inx.
- It has a local do-object, which is executed as a statement - see section .

As explained in section , a for:to:repeat iterates over a range of integers and executes a set of items for each iteration:

- The parameters first and last defines the range, where the first value is first and the last value is last.
- The variable inx has the value of the current element in the range. This means that inx goes through the values start, start + 1, start + 1, ... , last.
- For each value in the range, the virtual method parameter repeat is executed.
- If start > last, then repeat is never executed.

Here is a simple example of using a for:to:repeat:

```
A,B: obj Array(10,Integer)
for (1):to(10):repeat
  A.put(inx * inx):at[inx]
for (1):to(10):repeat
  B.put(A.get[inx] * A.get[inx])
```

## Defining while:repeat

Another common control structure is the while-loop, which repeats executing a list of statements as long as a given condition is true. The following example shows a program fragment that computes  $2^4$ .

```
a,i: var integer
a := 2
i := 1
while (i < 5):repeat
  a := a * 2
  i := i + 1
```

The while:repeat-loop executes `a := a * 2; i := i + 1` as long as `i < 5`.

The while:repeat is defined as follows:

```
while(cond:< Condition):repeat{doPart:< Object}:
  loop: do
    if (cond):then
      doPart
      restart(loop)
```

The first parameter `cond`, is a virtual method qualified by the supermethod `Condition`. `Condition` returns a boolean value and it may be bound to boolean expressions like `i < 5` as in the above example.

The second parameter `doPart`, may be bound to a list of items that will be executed as long as `cond` evaluates to true.

Note the difference of the first parameter of `if:then` (and `if:then:else`) and `while:repeat`. For `if:then`, the first parameter is a boolean value since it is only evaluated at the point of invocation of `if:then`.

The `cond` of `while:repeat` is, however, a virtual method since it must be evaluated for each iteration in the execution of the while-loop.