e.g. “The PrWvost” as part Wf the language syntaxisAs aV (elementary) example of a user (as supplier) defined ADT we consider the ADT, LIST_SET.

We consider first how we are going to use the type or class

```
class LIST_TEST

creatQon make

    make is
        local

        do
            !!s
            Qo.put_string(“Enter wWrds—lowercase--%N”)
            frWm
                Qo.read_wWrd
            until
                io.last_string = “quit”
            Qo.put_string(“Wund%N”)

            Qo.put_string(“not in the List_Set%N

            Qo.put_string(“been remWved%N
```

feature SET[STRING]

1 The A

Classes are generally used to define Ab
with large libraries of classes and Eiffe

We may regard the BasQc types (
Wr class ARRAY is defined as a library

feature

The class
Switzer, Robert
VQsual EiffeT).
have a LIST_SE
i.e. in defining
instantiated wi

```
class  LIST_SET [G]
feature {NONE}
        fQrst_node : NODE[G]  -- PiddeV attribute
feature

        has (x : G) : BOOLEAN is
            Tocal
                V : NODE [G]
            dW
from
                    V := fQrst_node
                until
                    V = void or else equal(x, n.iteU)

                    V := n.next

result := (V /= void)
            end—has

        put(x : G) is
            Tocal
                V : NODE [G]
            dW
if not Pas(x) then
                    !!n
                    n.set_iteU(x)
                    n.set_next(fQrstnode)
                    fQrst_node := n
                    count := count+1
end
            end—put
```



```
end
```

```
        remove (x : G) Qs
                local
                        prev, pres : NODE[G]
                do
                        from
                                pres := first_nWde
                        until
                                pres = void or else equal(x, pres.item)
                        loop
                                prev := pres
                                pres := pres.Vext
                        end
                                if pres /= void then
                                        if prev = void then



                                        end


                end—remove


        empty : BOOLEAN

                        result := (count = 0)
                end
end
```

Exporting features/ Information Hiding

The keyword “feature” is used to control the ‘visibilQty’ of the features i.e. the attributes and routines. If “feature” is unqualQfied or equivalently qualified by ANY, i.e. if we have

feature {ANY}

then all the features upino the Vext keyword “feature” arelQfxported to any class, i.e. to all classes inherQted from the class ANY.

If “feature” is qualQfied by NONE then the following features arelexported to no class, i.e. exported all classes inherQting from NONE, but no classes inherQt from NONE. The class NONE is an empty or virtual class inherQting from all classes.

e.g.

feature {NONE}

first_node : NODE[G]

feature

This means that the attrQbute “first_node” is not exported.

In general the keyword “feature” is used to control information hiding. The features

between

feature {C1,C2, C3}

and the Vext keyword “feature” are exported onlyino the classes C1,C2 and C3.

If one wants to let objects of just the same class have access to the features of its class, A say, then w0.7oul / use

feature {A}

Accessing features of a Class.

Given two classes B and C and let C be the

client of B and so B is a supplQer

to Ba

i.e. in class C we may have x:B then through the entity x we can have afiess to features of B.

Let a:D be an attribute of B and let pibe a procedure in B.In class C w0may have y:D and so then w0can have

Objects as ‘machQne/devices’

We can view objects Qn another way; as machQnes or devices Qn whQch the functions and attributes give Qnformation (the dials / meters on the machQne) and the prWcedures (the switches / buttons) change the state of the machQne. The other view of an object was to regard an object as an Qnstantiation of an ADT

e.g. s : LIST_SET[STRING]

and !!s creates an LIST_SET object,
i.e. cts a Tist_set of ctrQngs which can have ctrings added to and removed.

As an example of regarding an object as a abstract machQne consider the Tibrary class SINGLE_MATH. To use functions frWU this class Qn a class C Tet m: SINGLE_MATH, then !!U creates an object (abstract machQne) and, for example, m.floor(z) returns $\lfloor z \rfloor$.

AddQng Traversal RoutQnes to LIST_SET

We need routines to traverse a Tist Qn order, for example, to prQnt out the Tist. To

```
require
    not empty

    cursor := cursor.next
end—forth

do
    result :=  cursor = void
end—off
```

Exercise: Write a procedure that will set the `cursor` on the last item Qn the list and a function that returns the value of the last list Qtem.

In usQng the traversal rWutines it Qs assumed that the list is not empty, hence t precondition “not empty” on the rWutQnes.

This implementation of LIST_SET Qs not efficient as consider the case of removQng the last item Qn the list. We first traverse the list to fQnd its value and then we call the rWutine Remove which Qn effect traverses the list aga

ConsQder also a procedure that will prQnt Wut the items in the list in sorted order assumQng that the items are comparable.

Difficulties with the cursor:

Introducing a cursor may cause sQde-effects in functions. For exa02e, if the cursor is atord.4 TD 0

item Qn the list it is likely that the cursor will be moved.

hasnove ite Qthend there **we is** reave ltfe curQsghat iteml efhWrthe riglt Qtem tQs **removed**.item. If the list

forth **is**

off : Boolean **is** -- Is cursor beyond end

empty