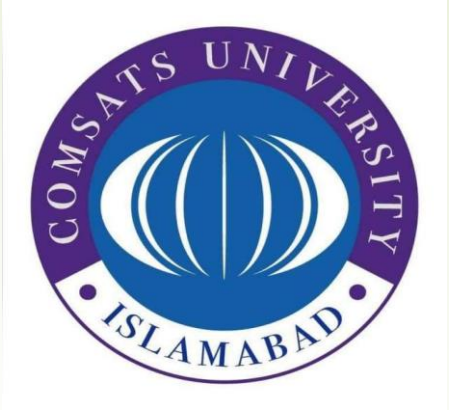

Department of Computer Science, CUI Lahore Campus

Formal Methods

By

Farooq Ahmad

Sequence type in VDM-SL

- To declare a variable to be of type *sequence* we place an asterisk after the name of the type contained within the sequence. For example, the statement *seq* : $\mathbb{Z}^*$ declares a variable *seq* to be a sequence of integers.
- If we had previously declared a type *SpaceCraft* then we could declare a variable *convoy*, for example, as follows:
- *convoy* : *SpaceCraft**

Specifying a Stack

- Stack is an ordered list that obeys a last-in-first-out (LIFO) protocol. Thus, items are added to the list, and when it comes to the time that an item is to be removed, then this item will be the last one that was added.

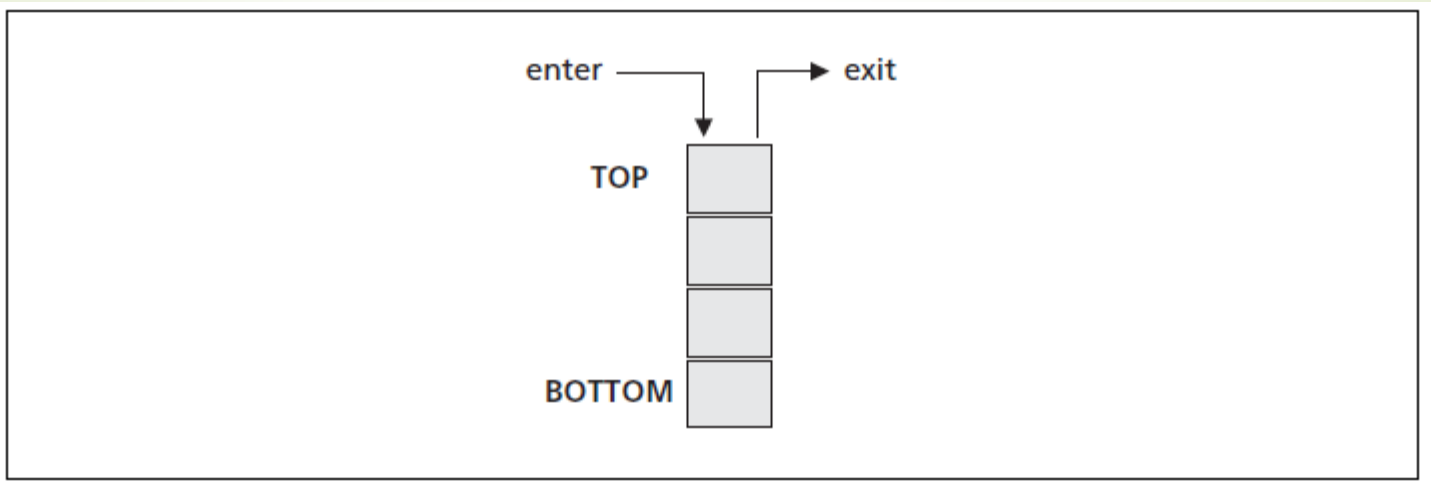

Figure 7.1 A stack

UML Model of Stack

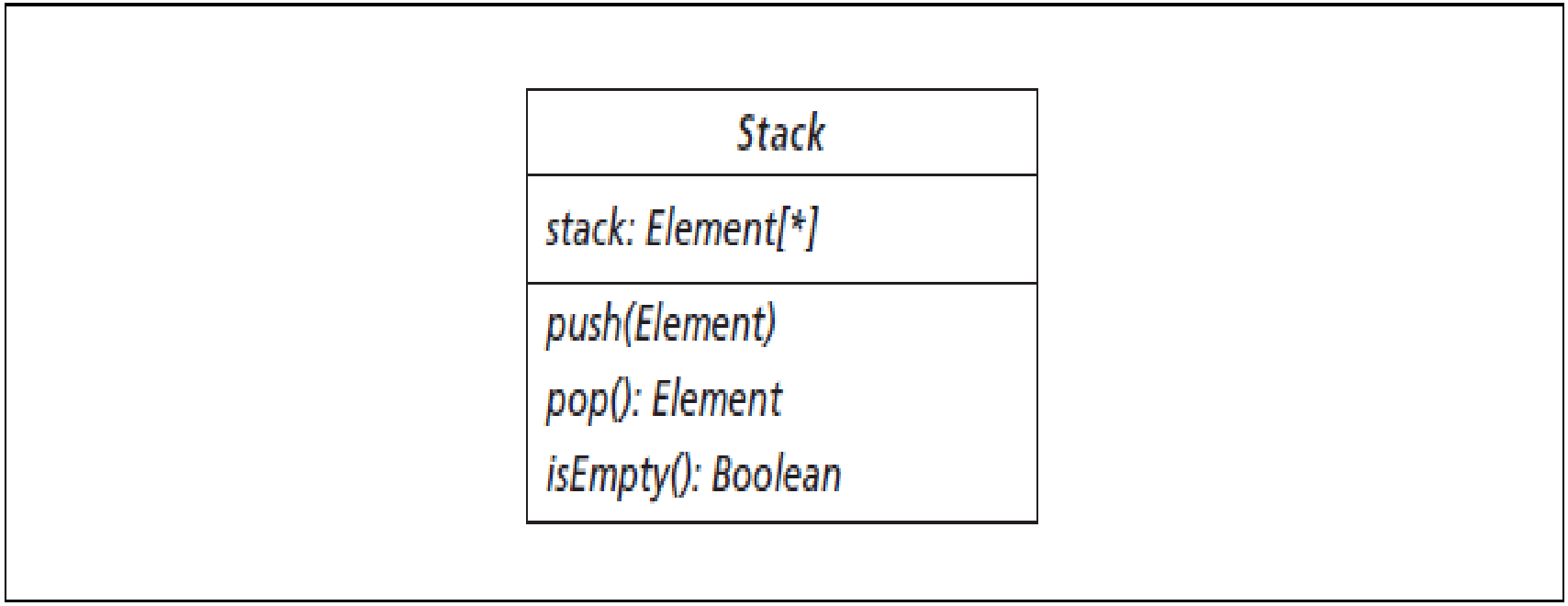

Figure 7.2 The UML specification for a *Stack* class

Types

- We are going to create a generic stack by specifying a token type called *Element* – this could be defined in detail at implementation time, but for the purposes of specification its internal details are not relevant. We can now proceed to the specification of the stack in VDM-SL.

```
types
Element = TOKEN
```

State of the System

- As you can see from the UML diagram, the stack must contain a collection of elements; as this must be an ordered collection we will choose the sequence type.

state *Stack* of

stack : *Element**

init mk-*Stack*(*s*) $\underline{\Delta}$ *s* = []

end

Operations

$push(itemIn : Element)$

ext wr $stack : Element^*$

pre TRUE

post $stack = [itemIn] \hat{} \overline{stack}$

Operations

$$
\begin{array}{ll}
pop() \; itemRemoved : Element & \\
\text{ext wr} & stack : Element^* \\
\text{pre} & stack \neq [] \\
\text{post} & stack = \text{tl } \overline{stack} \; \wedge \; itemRemoved = \text{hd } \overline{stack}
\end{array}
$$

Operations

$isEmpty()\ query : \mathbb{B}$

ext rd $stack : Element^*$

pre TRUE

post $query \Leftrightarrow stack = []$

The Airport System

- In our new system, when an aircraft is to take off from its original airport, then at that time it requests permission to land at airport. When it approaches the airport it is placed in a queue, and must circle the airport until a runway becomes available. Only aircraft that have permission to land are allowed to circle. The circling aircrafts are landed on a first-come-first served basis.

Types

types

Aircraft = TOKEN

State

state *Airport2* of

permission : *Aircraft*-set

landed : *Aircraft*-set

circling : *Aircraft**

Initialization

$$\textbf{init mk-}\mathit{Airport2}(p, l, c) \triangleq p = \{\} \wedge l = \{\} \wedge c = [\,]$$

Invariant

$$\text{inv mk-}Airport\ 2(p,l,c) \triangleq l \subseteq p \wedge \text{elems } c \subseteq p \wedge \text{elems } c \cap l = \{\} \wedge isUnique(c)$$

System Description

types

$Aircraft = \text{TOKEN}$

state *Airport2* **of**

$permission : Aircraft\text{-}\mathbf{set}$

$landed : Aircraft\text{-}\mathbf{set}$

$circling : Aircraft^*$

inv mk-*Airport2*$(p,l,c) \triangleq$ $l \subseteq p$
$\wedge$ **elems** $c \subseteq p$
$\wedge$ **elems** $c \cap l = \{\}$
$\wedge$ $isUnique(c)$

init mk-*Airport2*$(p, l, c) \triangleq p = \{\} \wedge l = \{\} \wedge c = [\,]$

end

Functions

functions

$$isUnique(seqIn : Aircraft^*)\ query : \mathbb{B}$$

pre $seqIn \neq []$

post $query \Leftrightarrow \forall\, i_1, i_2, \in \text{inds}\ seqIn \bullet i_1 \neq i_2 \Rightarrow seqIn(i_1) \neq seqIn(i_2)$

Operations

operations

getPermission() *permissionOut* : *Aircraft*-set

ext rd *permission*: *Aircraft*-set

pre TRUE

post *permissionOut* = *permission*

Operations

getLanded() landedOut : Aircraft-set

ext rd *landed : Aircraft-set*

pre TRUE

post *landedOut = landed*

Operations

*getCircling() circlingOut : Aircraft**

ext rd *circling : Aircraft**

pre TRUE

post *circlingOut = circling*

Operations

givePermission (*craftIn* : *Aircraft*)

ext wr *permission*: *Aircraft*-set

pre $craftIn \notin permission$

post $permission = \overline{permisssion} \cup \{craftIn\}$

Operations

allowToCircle (*craftIn* : *Aircraft*)

ext wr *circling* : *Aircraft**

rd *permission* : *Aircraft*-set

rd *landed* : *Aircraft*-set

pre $craftIn \in permission \wedge craftIn \notin \textbf{elems}\ circling \wedge craftIn \notin landed$

post $circling = \overline{circling}\ \frown\ [craftIn]$

Operations

recordLanding( )

ext wr *circling* : *Aircraft**

wr *landed* : *Aircraft*-set

pre $circling \neq []$

post $landed = \overline{landed} \cup \{ \text{hd } \overline{circling} \} \wedge circling = \text{tl } \overline{circling}$

Operations

recordTakeOff (*craftIn* : *Aircraft*)

ext wr *landed*: *Aircraft*-set

pre *craftIn* ∈ *landed*

Operations

numberWaiting( ) *total* : ℕ

ext rd *landed* : *Aircraft*-set

rd *permission* : *Aircraft*-set

pre TRUE

post *total* = card(*permission**landed*)

Reference and reading material

- Formal Software Development From VDM to Java, Chapter # 7: **Sequences**