Use of Generics

YORK UNIVERSITÉ UNIVERSITY

EECS3311 A & E: Software Design
Fall 2020

CHEN-WEI WANG

Learning Objectives

Upon completing this lecture, you are expected to understand:

1. How to ***write*** a generic class (as a ***supplier***)
2. How to ***use*** a generic class (as a ***client***)

Generic Collection Class: Motivation (1)

```
class STRING_STACK
feature {NONE} -- Implementation
  imp: ARRAY[STRING] ; i: INTEGER
feature -- Queries
  count: INTEGER do Result := i end
    -- Number of items on stack.
  top: STRING do Result := imp [i] end
    -- Return top of stack.
feature -- Commands
  push (v: STRING) do imp[i] := v; i := i + 1 end
    -- Add 'v' to top of stack.
  pop do i := i - 1 end
    -- Remove top of stack.
end
```

- Does how we implement string stack operations (e.g., `top`, `push`, `pop`) depends on features specific to element type `STRING` (e.g., `at`, `append`)? [***NO!***]
- How would you implement another class `ACCOUNT_STACK`?

Generic Collection Class: Motivation (2)

```
class ACCOUNT_STACK
feature {NONE} -- Implementation
  imp: ARRAY[ACCOUNT] ; i: INTEGER
feature -- Queries
  count: INTEGER do Result := i end
    -- Number of items on stack.
  top: ACCOUNT do Result := imp [i] end
    -- Return top of stack.
feature -- Commands
  push (v: ACCOUNT) do imp[i] := v; i := i + 1 end
    -- Add 'v' to top of stack.
  pop do i := i - 1 end
    -- Remove top of stack.
end
```

- Does how we implement account stack operations (e.g., `top`, `push`, `pop`) depends on features specific to element type `ACCOUNT` (e.g., `deposit`, `withdraw`)? [***NO!***]
- A ***collection*** (e.g., table, tree, graph) is meant for the ***storage*** and ***retrieval*** of elements, not how those elements are manipulated.

Generic Collection Class: Supplier

- Your design *"smells"* if you have to create an *almost identical* new class (hence *code duplicates*) for every stack element type you need (e.g., INTEGER, CHARACTER, PERSON, etc.).
- Instead, as **supplier**, use *G* to *parameterize* element type:

```
class STACK [G]
feature {NONE} -- Implementation
  imp: ARRAY[ G ] ; i: INTEGER
feature -- Queries
  count: INTEGER do Result := i end
     -- Number of items on stack.
  top: G do Result := imp [i] end
     -- Return top of stack.
feature -- Commands
  push (v: G ) do imp[i] := v; i := i + 1 end
     -- Add 'v' to top of stack.
  pop do i := i - 1 end
     -- Remove top of stack.
end
```


Generic Collection Class: Client (1.1)

As **client**, declaring ss: STACK[*STRING*] instantiates every occurrence of G as STRING.

```
class STACK [∉ STRING]
feature {NONE} -- Implementation
  imp: ARRAY[ ∉ STRING ] ; i: INTEGER
feature -- Queries
  count: INTEGER do Result := i end
     -- Number of items on stack.
  top: ∉ STRING do Result := imp [i] end
     -- Return top of stack.
feature -- Commands
  push (v: ∉ STRING ) do imp[i] := v; i := i + 1 end
     -- Add 'v' to top of stack.
  pop do i := i - 1 end
     -- Remove top of stack.
end
```


Generic Collection Class: Client (1.2)

As **client**, declaring ss: STACK[*ACCOUNT*] instantiates every occurrence of G as ACCOUNT.

```
class STACK [∉ ACCOUNT]
feature {NONE} -- Implementation
  imp: ARRAY[ ∉ ACCOUNT ] ; i: INTEGER
feature -- Queries
  count: INTEGER do Result := i end
     -- Number of items on stack.
  top: ∉ ACCOUNT do Result := imp [i] end
     -- Return top of stack.
feature -- Commands
  push (v: ∉ ACCOUNT ) do imp[i] := v; i := i + 1 end
     -- Add 'v' to top of stack.
  pop do i := i - 1 end
     -- Remove top of stack.
end
```


Generic Collection Class: Client (2)

As **client**, instantiate the type of G to be the one needed.

```
1  test_stacks: BOOLEAN
2   local
3     ss: STACK[STRING] ; sa: STACK[ACCOUNT]
4     s: STRING ; a: ACCOUNT
5   do
6     ss.push("A")
7     ss.push(create {ACCOUNT}.make ("Mark", 200))
8     s := ss.top
9     a := ss.top
10    sa.push(create {ACCOUNT}.make ("Alan", 100))
11    sa.push("B")
12    a := sa.top
13    s := sa.top
14  end
```

- **L3** commits that ss stores STRING objects only.
  - **L8** and **L10** ***valid***; **L9** and **L11** ***invalid***.
- **L4** commits that sa stores ACCOUNT objects only.
  - **L12** and **L14** ***valid***; **L13** and **L15** ***invalid***.

Index (1)