Ada Fundamentals with GNAT: Workshop 4

1. Stack package

Objective: This problem illustrates information hiding through private types, and also provides practice with discriminated records.

One of the standard “container” data structures in computer science is a pushdown stack, which has the following properties (we are assuming that the elements in the stack are integers):

* The maximum size of a stack object is specified when the object is created
* A stack object has a current size, an integer between 0 and its maximum size, inclusive. When a stack object is created, its current size is 0
* There are four client-visible subprograms defined for stacks
* **procedure** Push(S : **in out** Stack, E : **in** Integer);  
  Add element E to stack S. Precondition: S is not full
* **function** Top(S : Stack) **return** Integer;  
  Return the most recent value Pushed onto S, without removing it. Precondition: S is not empty
* **procedure** Pop(S : **in out** Stack; E : **out** Integer);  
  Remove the element at the top of the stack (i.e. the one most recently Pushed), and assign it to E. Precondition: S is not empty.
* **function** Current\_Size(S : Stack) return Natural;  
  Returns the number of elements Pushed onto, and not yet Popped from, S

Define a package named Int\_Stack\_Pkg that implements the stack type Stack and these operations. Declare Stack as either a private type or a limited private type. (Which is preferable? Why?)

Declare exceptions Overflow and Underflow. Push should raise Overflow when called with a full stack (i.e., when the current size = max size). Pop and Top should raise Underflow when called with an empty stack (when current size = 0).

Test the program with the main procedure from the workshop directory.

2. “Stubbing” the Stack package body

Objective: This problem illustrates subunits.

After completing problem 1, revise the Stack package body so that one or more of the subprograms is implemented as a subunit.

3. Extending the Stack package

Objective: This problem illustrates child units

Add the following declarations in a package Int\_Stack\_Pkg.Utilities that is a child of Stack\_Pkg:

* **procedure** Flush(S : **in out** Stack);  
  Resets S so that its current length is 0
* **procedure** Display(S : **in** Stack);  
  Outputs each element of S, from bottom to top, without popping.

Test the program with the main procedure from the workshop directory.

Why do these declarations need to be in a child package versus a “client” package that “with”s Stack\_Pkg?

Why not add these declarations to the original Stack\_Pkg?