"UML" for a second **stack** class (revised **2-3-05**)

NOW: for a FIXED CAPACITY stack

adapted from Ch. 7, Savitch and Main, "Data Structures and Other Objects Using C++"

Template Class: `stack`

/* a collection of items such that entries can be inserted and removed at only one end (called the **top**). */

Member data and related details:

/* contains elements of the type set to be the value of template parameter **`Item`** */

/* has a fixed capacity */

Constructors:

/* postcondition: creates an empty **stack** instance */

```
stack( );
```

Accessors and other constant member functions:

/* postcondition: returns **true** if stack is empty, and returns **false** otherwise */

```
bool     is_empty( ) const;
```

/* postcondition: returns **true** if stack is full (if it contains the number of items equal to its capacity), and returns **false** otherwise */

```
bool     is_full( ) const;
```

/* precondition: is_empty() == false */
/* postcondition: returns the value of the top item of the stack, BUT the stack is unchanged. */

```
Item     get_top( ) const;
```

/* postcondition: returns the capacity of the stack (how many items it CAN hold) */

```
int      get_capacity( ) const;
```

/* postcondition: returns the number of elements currently in the stack */

```
int      get_size( ) const;
```

Modifiers and other modifying member functions:

/* precondition: is_full() == false */
/* postcondition: a new copy of **entry** has been pushed onto the (top of the) stack */

```
void     push(const Item& entry);
```

/* precondition: is_empty() == false */
/* postcondition: the top item of the stack has been removed, and its value is returned */

```
Item     pop( );
```