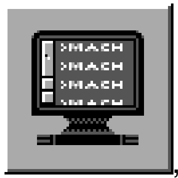
OOP Class Lab
Stack

Purpose: This lab is an exercise in basic Objective-C, creating a subclass, and writing methods. You will design a new object called Stack which performs like a stack data structure, and you will test it with a main program already provided.

1. Start up a Terminal shell , and change to the **OOPClass**/**Labs**/**Stack** folder (type "**cd OOPClass**/**Labs**/**Stack**". Type "**pwd**" to check that you are there.).

2. Run the Stack program in the Solution directory (type "**Solution**/**Stack**") to see how it works, comparing with the code in stackMain.m.

3. Use Edit  to write a subclass of Storage (an Objective-C common class) called Stack (.h and .m files). The Stack.h interface file is already provided for you, as is a skeleton Stack.m implementation file for which you will need to write the methods.

4. From studying the stackMain.m test program, you can see that a stack object must respond to the following messages:

- init (initializes the stack. This one is provided to get you started.)
- **push:** (takes number passed to it and pushes onto the stack. Returns self.)
- **(float) pop** (pops a number off the stack and returns it. Returns zero if empty.)
- **(float) top** (returns top value of stack without actually popping it. Returns zero if empty.)
- **printStack** (prints out in order all numbers on the stack. Prints "Stack is empty." if empty. Returns self.)
- empty (empties the stack. Storage's inherited method will do here.)
- free (frees up the instance. Storage's inherited method will do here.)

However, two of these methods, empty and free, are inherited from Storage and do not need to be modified. Also, the init method must perform the stack initialization, and it is provided for you. Therefore, you need only write the **push:**, **pop**, **top**, and **printStack** methods.

Read /**NextLibrary**/**Documentation**/**NextDev**/**GeneralRef**/**03_Common**/**Classes**/**Storage.rtf** in order to learn about the existing Storage class.

5. You can test the program using the stackMain.m program provided. Since the Makefile is also already provided, all you need to do to build the program with your new object is to type "**make**". Then run your program by typing "**Stack**".

6. If you have problems, you can sneak a peak at Stack.m in the **Solution** directory.

Extensions: Try a slight variation on Stack and design a Queue object. Or, design a SquareMatrix object, where each of the columns is an instance of the Storage class. Give SquareMatrix methods to set or read a particular entry, to transpose the matrix instance, or to print the values in the matrix.