project 2 - Stacks

Comp2000 – data structures Fall 2018

# Topic(s)

Stack Applications and Implementations

# Readings & review

* Carrano, *Data Structures and Abstractions with Java* Chapters 5, 6 (prelude and Appendices if you need a Java review).
* Stack ADT & implementation lectures

# Objectives

Be able to:

* Implement a Stack ADT (Abstract Data Type)
* Use your Stack ADT in an application

# instructions

1. Please let instructor know **BEFORE YOU START** who you will work with.
2. Read over the assignment and ASK QUESTIONS about anything that you don’t understand (before you start).
3. Do the [problems](#_Problems) listed in the next section.
   1. Be sure to follow [Good Programming Practices](#_Good_programming_practices),
   2. Keep in mind the [guidelines on plagiarism](#_guidelines_on_plaigarism).
4. Do a [Write-up (Analysis /](#_Analysis_/_Summary:) Summary) for your project.
5. Follow instructions for [submitting your work](#_Submitting_your_work:).

# Problems

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Title** | **Ch** | **Project** | | **Page** | **Notes** |
| **Stack Implementation** | 6 | **Specific for the group** | | 193-194 (4th e)  194 (5th e) | Remember: Your implementation should be **your own**, written from scratch (so **you learn** the concepts) according to the project description. |
| **Stack Application** | 5 | **Select one** | **#7** | Lisp: 178 | * You may also propose your own project as an application of Stacks. * **Start early**, ask questions, relax and **have fun**! (challenges should be fun) * The tasks are described in the provided word documents. * Do #9 if you are familiar with Java GUI |
| **#9** | Calculator:179 |
| **#100** | PostfixToInfix |

# Good programming practices

**For all classes**

* Make your instance variables **private**
* Include **constructors** to initialize your instance variables.
* Derived class constructors should **leave initialization of super class instance variables** to the super classes’ constructors:
  + Remember the call to the super classes constructor is: **super( <init1>, <init2>,..)**
* Include **Accessor** and **Mutator** methods for all instance variables *(please ask if you’re not sure what these are)*
* Include a **main** method for testing (unless it’s an abstract class) and test before you proceed…
* Add comments to your code, not just so it’s easier for other readers, but also so it’s easier for you to remember your logic.

# wrIte-up analysis

1. In general, how useful is the Stack data structure? Can you think of other uses aside from those discussed in the applications chapter?

2. If we didn’t have the Stack data structure can you think of another way to implement the LIFO (last in first out) concept?

## summary

1. If you worked in groups:
   1. How did you “divide up” the work so that each student still met the objectives for the assignment (i.e., learned, understood and applied the concepts).
   2. What was your contribution?
   3. How did you coordinate code changes/testing?
   4. Other observations about working with a partner?
2. Where did you have trouble with this assignment? How did you move forward? What topics still confuse you?
3. What did you learn from this assignment? (Please be specific)
4. How could this assignment be improved in the future?

# Submitting your work:

1. Make sure that your name(s) are in all of your project files.
2. If you have more than one file for your solution, make a .zip file for your project
3. In Blackboard, attach your solution file to the submission for this assignment.

# guidelines on plaigarism in computer science

## Clear Plagiarism

A clear case of plagiarism exists if a student **passes off someone else's work as his or her own**. Examples of such behavior include (but are not limited to) the following:

* A student making cosmetic alterations to another's work and then handing it in as his or her own.
* A student having another person complete an assignment and then handing it in as his or her own.
* A student handing in (as his or her own) a solution to an assignment performed by someone else from a previous offering of the course.

These are all cases of indisputable plagiarism and are characterized by the submission of work, performed by another, under one's own name.

## Permitted Collaboration

Collaboration and research, where they do not constitute plagiarism, should be generally encouraged. These efforts constitute honest attempts at obtaining a deeper understanding of the problem at hand. They can also serve as a validation of a student's approach to a problem. Some examples are as follows:

* A student researching existing, general public approaches (but not source code solutions) to a problem presented as an assignment.
* A group of students discussing existing, public approaches to a problem presented as an assignment.
* A group of students discussing the requirements of an assignment.
* A student discussing the merits of his or her proposed solution to an assignment with the instructor or teaching assistant.

Provided that no plagiarism is committed, these are all valid (and encouraged) activities.

## The Gray Area

The differences between the examples of plagiarism and encouraged collaboration above are those of originality and acknowledgment. In general, any work submitted without acknowledgment which is not the original work of the indicated author constitutes plagiarism. Some examples which illustrate this point follow. It should be noted that while some of the examples do not constitute academic misconduct, they may be of questionable academic value. It should also be noted that anyone (knowingly or through negligence) contributing to someone else's academic misconduct is also guilty of the same.

* It is acceptable to use solution proposals presented by the instructor or teaching assistant. The acknowledgment is implicit. Explicit acknowledgment is not usually required.
* It is not acceptable to use publicly available work without acknowledgment.
* It is not acceptable to use a full or partial solution obtained from or by another through any means (verbal, written, electronic, etc.), without that person's permission.
* It is not acceptable to collaborate on a single solution without acknowledgment.
* It is not acceptable to discuss solutions or partial solutions to a problem and then incorporate them without acknowledgment.
* It is acceptable to implement a standard solution to a problem without acknowledgment, but it is not acceptable to incorporate someone else's implementation without acknowledgment. Here standard solution means a commonly used data structure or algorithm.
* It is not acceptable to re-submit an assignment from another course or a previous offering of the same course without acknowledgment, regardless of authorship.
* It is not acceptable to make a solution available as an aid to others.

This set of examples helps define the bounds between encouraged behavior and misconduct but does not constitute an exhaustive set.