# Assignment 2
# Stacks and Queues

| Release Time | Due Date |
|---|---|
| 9/30/2016 | 10/17/2016 |

## Objectives

- Creating stacks and queues.
- To practice add, delete, search, change data using stacks and queues.
- Practice developing high-performance computing solutions
- Compare the differences between stacks and queues.

## Problem Specification

In reality, we always use various data structures to store data, and different data structures have different advantages and disadvantages. For instance, some data structures are good at adding/deleting new data, but they may have deficiency on searching. We already learned comparisons of linked-lists and arrays from our previous assignment. Now lets manipulate two more data structures, namely stacks and queues. After this assignment is completed, you will learn more and have a good understanding of these two linear data structures. Here we go:

Write a JAVA program to solve the following problem:

1) Design and implement a Java class for **stacks** using linked-lists (instead of the ones from the Java library), i this class should have at least two functions push() and pop();

2) Design and implement a Java class for **queues** using linked-lists (don’t use built-in queue related Java classes), this class should have at least two functions add() and delete();;

3) Now, design and implement a strategy of using two stacks to implement the functionality of a queue;

4) Use two queues to implement a stack;

5) Implement a function, minValue() for stack, which returns the minimum value with a time complexity O(1); (Hint: can two stacks help solve this problem?) (Note: this is an example of a tradeoff between space and time!)

6) Your implementations should wprk easily on int, float, double, and String type values – use Java generics.

## Design Requirements

### Code Documentation

For this assignment, you must include documentation for your code as generated by JavaDoc. You should have JavaDoc comments for every class, constructor, and method. By default, JavaDoc should output html documentation to a subfolder within your project (/dist/javadoc). Make sure this folder is included when you zip your files for submission. You do not need to submit a hard copy of this documentation.

Hint: http://stackoverflow.com/questions/4468669/how-to-generate-javadoc-html-in-eclipse

### Coding Standards

You must adhere to all conventions in the CS 3310 Java coding standard. This includes the use of white spaces for readability and the use of comments to explain the meaning of various methods and attributes. Be sure to follow the conventions for naming files, packages, classes, variables, method parameters and methods.

### Testing

Make sure you test your application with several different values, to make sure it works.

A data file may be provided later – if we do, it will just include a number of strings one per line.

### Assignment Submission

- Generate a .zip file that contains all your files, including:
    - Source code files
    - Including any input or output files
- Javadocs