Introduction to Computer Programming (Java A)
Lab 13

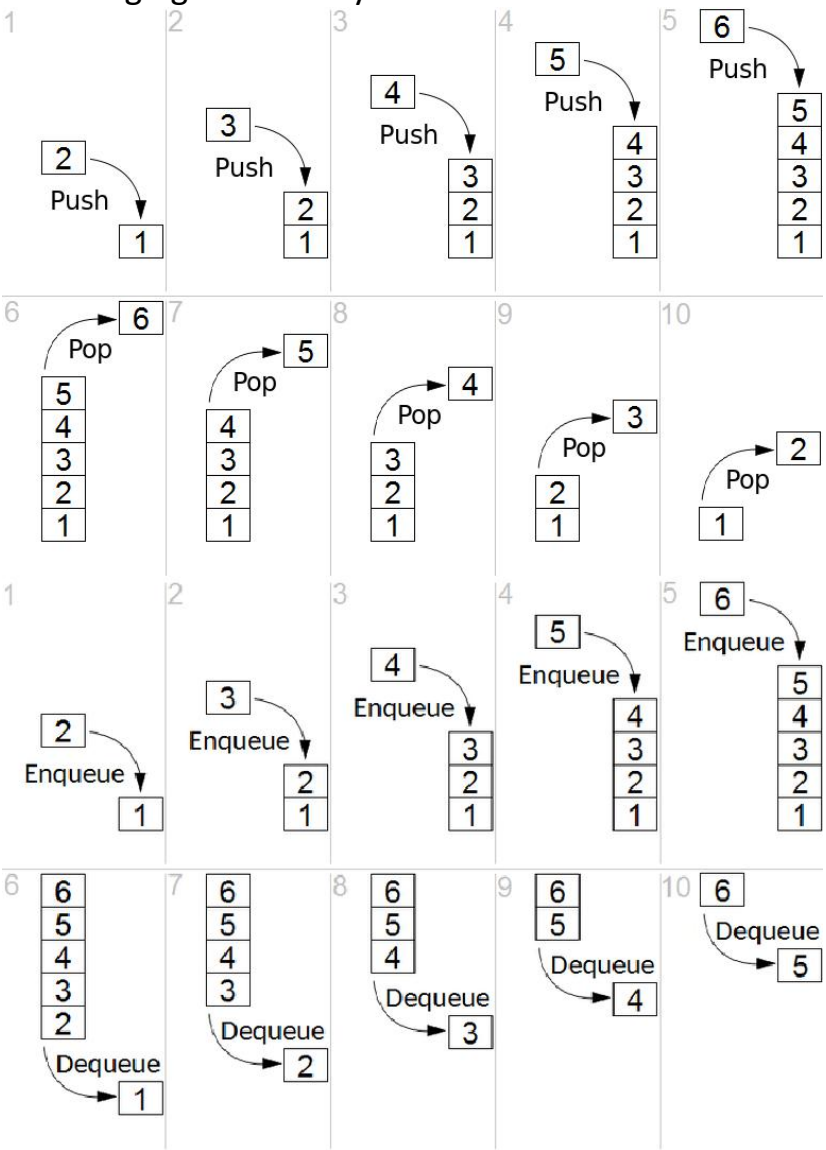
[Objective]

- Learn implementing generic class.
- Learn recursion

[Exercises]

1. Generic Classes

Stack and Queue are two widely used data structures. Stack has the property LIFO (last in first out), while Queue has the property FIFO (first in first out). In this lab, you are required to implement two generic classes: Stack<E> and Queue<E>. You should define the operations ***push*** and ***pop*** in the Stack class and ***enqueue*** and ***dequeue*** in the Queue class. Under the hood, you can use an ArrayList to store data items. The following figures show you the data structures.



Requirements:

The generic class Stack<E> should contain the following methods:

- ***push***: this method pushes a data item onto the stack
- ***pop***: this method pops the latest added data item from the stack. The method may throw runtime exceptions when it is invoked on an empty stack.
- ***hasItems***: this method returns true if the stack is not empty and false otherwise.

The generic class Queue<E> should contain the following methods:

- ***enqueue***: this method adds a new data item into the queue.
- ***dequeue***: this method returns the oldest data item from the queue and returns it.
- ***hasItems***: this method returns true if the queue is not empty and false otherwise.

We provide two classes for testing: TestStudent and Student. Please run the main method in the TestStudent class and see if it prints the expected result. If yes, your implementation is likely correct.

Copy the following code to Student.java

```
public class Student
   { private String
   firstName; private String
   lastName; private Gender
   gender;
   public Student(String firstName, String lastName, Gender gender)
      { this.firstName = firstName;
      this.lastName = lastName;
      this.gender = gender;
   }

   public String toString() {
      return String.format("%s %s, %s", firstName, lastName, gender);
   }
}
enum Gender
   { MALE,
```

Copy the following code to TestStudent.java

```
public class TestStudent {
   public static void main(String[] args) throws Exception
      { Student s1 = new Student("Harry", "Potter",
      Gender.MALE); Student s2 = new Student("Ron", "Weasley",
      Gender.MALE);
      Student s3 = new Student("Hermione", "Granger", Gender.FEMALE);
      // test queue implementation
      Queue<Student> queue = new Queue<Student>();
      queue.enqueue(s1);
      queue.enqueue(s2);
      queue.enqueue(s3);
```

```
        while(queue.hasItems())
          { System.out.println(queue.dequeue());
        }

        // test stack implementation
        Stack<Student> stack = newStack<Student>();
        stack.push(s1);
        stack.push(s2);
        stack.push(s3);
        System.out.println("---Stack: last in first out---");
        while(stack.hasItems()) {
          System.out.println(stack.pop());
        }
    }
}
```

Expected output:

```
---Queue: first in first out---
Harry Potter, MALE
Ron Weasley, MALE
Hermione Granger, FEMALE
---Stack: last in first out---
Hermione Granger, FEMALE
Ron Weasley, MALE
Harry Potter, MALE
```

To ease your task, the template code for Stack.java is given to you:

```
import java.util.ArrayList;

public class Stack<E> {
  private ArrayList<E> items = new ArrayList<E>();

  public void push(E item) {
    // fill in your code here
  }

  public E pop() {
    // fill in your code here
  }

  public boolean hasItems() {
    // fill in your code here
  }
}
```

2. Recursion

In linear algebra, the **determinant** is a scalar value that can be computed from the elements of a square matrix and encodes certain properties of the linear transformation described by the matrix. The determinant of a matrix A is denoted $\det(A)$, $\det A$, or $|A|$. In the case of a 2×2 matrix the determinant may be defined as

$$|A| = \begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$$

Similarly, for a 3 × 3 matrix A, its determinant is

$$|A| = \begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} = a \begin{vmatrix} \square & \square & \square \\ \square & e & f \\ \square & h & i \end{vmatrix} - b \begin{vmatrix} \square & \square & \square \\ d & \square & f \\ g & \square & i \end{vmatrix} + c \begin{vmatrix} \square & \square & \square \\ d & e & \square \\ g & h & \square \end{vmatrix}$$

$$= a \begin{vmatrix} e & f \\ h & i \end{vmatrix} - b \begin{vmatrix} d & f \\ g & i \end{vmatrix} + c \begin{vmatrix} d & e \\ g & h \end{vmatrix}$$

$$= aei + bfg + cdh - ceg - bdi - afh.$$

For a 4 × 4 matrix, its determinant can be decomposed as

$$\begin{vmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{vmatrix} = a \begin{vmatrix} f & g & h \\ j & k & l \\ n & o & p \end{vmatrix} - b \begin{vmatrix} e & g & h \\ i & k & l \\ m & o & p \end{vmatrix} + c \begin{vmatrix} e & f & h \\ i & j & l \\ m & n & p \end{vmatrix} - d \begin{vmatrix} e & f & g \\ i & j & k \\ m & n & o \end{vmatrix}$$

Implement the following function where the input argument is a N x N square matrix (N >= 1) represented by a two-dimensional array:

```
public int determinant (int[][] a) {
    // write your code here
    // return the determinant to the caller of this method

}
```

To test your code, the determinant of the following 5 x 5 matrix is 28

$$\begin{vmatrix} 1 & 2 & 3 & 4 & 1 \\ 0 & -1 & 2 & 4 & 2 \\ 0 & 0 & 4 & 0 & 0 \\ -3 & -6 & -9 & -12 & 4 \\ 0 & 0 & 1 & 1 & 1 \end{vmatrix} = 28$$