Tutorial 09: Abstract Data Types (ADTs) (1): Stacks and Vector

Related Objectives from Unit Outline:

- Describe the concept, application, and specification of an abstract data type (ADT) and employ Java classes to encapsulate ADTs

Objectives:

1. To become familiar with the concepts and applications of Stacks, their implementation using alternative data structures, and existing implementation in Java classes;
2. To demonstrate the awareness of the principles of algorithms behind the Java implementations of Stacks.

Tasks:

Complete the following.

Task 1: The following is a simplified algorithm that tests whether a *phrase* is well-bracketed.

1. Make *bracket-stack* empty.
2. For each symbol *sym* in phrase (scanning from left to right), repeat:
   2.1. If *sym* is a left bracket:
      2.1.1. Add *sym* to the top of *bracket-stack*.
   2.2.If *sym* is a right bracket:
      2.2.1. If *bracket-stack* is empty, terminate with answer false.
      2.2.2. Remove a bracket from the top of *bracket-stack* into *left*.
      2.2.3. if *left* and *sym* are not matched brackets, terminate with answer false.
3. Terminate with answer true if *bracket-stack* is empty, or false if otherwise

Hand-test this algorithm with the following phrases:

```
a. main(String[] args) {System.out.print (arg[0];}
b. [(a + b) – (c – d)
```

Task 2: Test the `push()`and `pop()` methods of the `Stack Class` implemented using Java `Vector Class` (Download the Java code WS0901 from Blackboard)

a. Explain what will be resulted from each of the statements in the code;
b. Check with your explanation by running the Java Program.

Task 3: Test the Vector class using WS0902 (download the Java code from Blackboard)

a. Note how to construct vector objects;

```
private static Vector v = new Vector();
private static Vector w = new Vector();
```

b. Observe the operation of some vector methods by analyzing the executed results.