Name: Sarvagnya H. Purohit **Registration Number**: 201070056
Date: 16-10-2021 **Branch:** Computer Engineering
Course: DSA **Course Instructor:** Dr. Mahesh Shirole

Lab Assignment 3

Aim: To perform array-based implementation of Stack and Queue Java API using generic classes

Theory:

Java Interface: An interface in Java is a blueprint of a class. It has static constants and abstract methods. The interface in Java is a mechanism to achieve abstraction. There can be only abstract methods in the Java interface, not method body. It is used to achieve abstraction and multiple inheritance in Java. We can have multiple classes which "implement" this interface, based on the abstract methods declared in it.

Java API: Application programming interface (API) in the form of a Java interface, which describes the names of the of the methods that the ADT supports and how they are to be declared and used. An API can consist of interfaces, classes and user interfaces. They enable developers to integrate various applications and websites and offer real-time information.

Generic classes in Java: A generic type is a generic class or interface that is parameterized over types. The idea is to allow any reference type (Integer, String, … etc., and user-defined types) to be a parameter to methods, classes, and interfaces. Using Generics, it is possible to create classes that work with different data types. An entity such as class, interface, or method that operates on a parameterized type is called a generic entity. Java includes support for writing generic classes and methods that can operate on a variety of data types while often avoiding the need for explicit casts.

To create an instance of a generic class, the syntax is:

```
BaseType <Type> obj = new BaseType <Type> ()
```

Stack data structure: A stack is a collection of objects that are inserted and removed according to the **last-in, first-out (LIFO)** principle. A stack allows access to only one data item: the last item inserted. If we remove this item, we can access the next-to-last item inserted, and so on. A user may insert objects into a stack at any time, but may only access or remove the most recently inserted object that remains (at the "top" of the stack). The fundamental operations involve the "pushing" and "popping" of a data-element on the stack.

Stack ADT: Formally, a stack is an abstract data type (ADT) that supports the following two update methods:

- push(e): Adds element 'e' to the top of the stack.
- pop(): Removes and returns the top element from the stack (or null if the stack is empty).

Additionally, stack ADT supports the following accessor methods:

- top(): Returns the top element of the stack, without removing it (or null if the stack is empty).
- size(): Returns the number of elements in the stack.
- isEmpty(): Returns a boolean indicating whether the stack is empty

Queue data structure: A queue defines a collection that keeps objects in a sequence, where element access and deletion are restricted to the first element in the queue, and element insertion is restricted to the back of the sequence. This restriction enforces the rule that items are inserted and deleted in a queue according to the **first-in, first-out (FIFO)** principle. Insert is also called put or add or "enqueue", while remove may be called delete or "dequeue"

Queue ADT: The queue abstract data type (ADT) supports the following two update methods:

- enqueue(e): Adds element e to the back of queue.
- dequeue(): Removes and returns the first element from the queue (or null if the queue is empty).

The queue ADT also includes the following accessor methods

- first(): Returns the first element of the queue, without removing it (or null if the queue is empty).
- size(): Returns the number of elements in the queue.
- isEmpty(): Returns a boolean indicating whether the queue is empty.

Test Data and Output of the program:

Stack:

```
"C:\Users\Sarvagnya Purohit9\.jdks\openjdk-17\bin\java.exe" "-javaagent:C:\Users\Sarvagnya
 Purohit9\AppData\Local\JetBrains\Toolbox\apps\IDEA-U\ch-0\212.5284.40\lib\idea_rt.jar=56268:C:\Users\Sarvagnya
 Purohit9\AppData\Local\JetBrains\Toolbox\apps\IDEA-U\ch-0\212.5284.40\bin" -Dfile.encoding=UTF-8 -classpath "D:\College
 Resources\Submissions\Assignment\DSA\Lab Assignment 3 Stack\out\production\Lab Assignment 3 Stack" com.company.Main
Enter size of the stack:
5
Enter any of the following options:
1.Push, 2. Pop, 3. Size, 4. isEmpty(), 5. Top, 6. Display, 0. Exit
4
Stack is empty
Enter any of the following options:
1.Push, 2. Pop, 3. Size, 4. isEmpty(), 5. Top, 6. Display, 0. Exit
1
Enter value to push:
5
Value has been pushed
Enter any of the following options:
1.Push, 2. Pop, 3. Size, 4. isEmpty(), 5. Top, 6. Display, 0. Exit
1
Enter value to push:
3
Value has been pushed
```

```
Enter any of the following options:
1.Push, 2. Pop, 3. Size, 4. isEmpty(), 5. Top, 6. Display, 0. Exit
3
Size of the stack is: 2
Enter any of the following options:
1.Push, 2. Pop, 3. Size, 4. isEmpty(), 5. Top, 6. Display, 0. Exit
6
3
5
Enter any of the following options:
1.Push, 2. Pop, 3. Size, 4. isEmpty(), 5. Top, 6. Display, 0. Exit
1
Enter value to push:
22
Value has been pushed
Enter any of the following options:
1.Push, 2. Pop, 3. Size, 4. isEmpty(), 5. Top, 6. Display, 0. Exit
1
Enter value to push:
33
Value has been pushed
Enter any of the following options:
1.Push, 2. Pop, 3. Size, 4. isEmpty(), 5. Top, 6. Display, 0. Exit
1
Enter value to push:
44
Value has been pushed
Enter any of the following options:
1.Push, 2. Pop, 3. Size, 4. isEmpty(), 5. Top, 6. Display, 0. Exit
1
Enter value to push:
55
Stack overflow error
Enter any of the following options:
1.Push, 2. Pop, 3. Size, 4. isEmpty(), 5. Top, 6. Display, 0. Exit
3
Size of the stack is: 5
Enter any of the following options:
1.Push, 2. Pop, 3. Size, 4. isEmpty(), 5. Top, 6. Display, 0. Exit
6
44
33
22
```

```
3
5
Enter any of the following options:
1.Push, 2. Pop, 3. Size, 4. isEmpty(), 5. Top, 6. Display, 0. Exit
5
Top element of the stack is: 44
Enter any of the following options:
1.Push, 2. Pop, 3. Size, 4. isEmpty(), 5. Top, 6. Display, 0. Exit
2
Popped element was: 44
Enter any of the following options:
1.Push, 2. Pop, 3. Size, 4. isEmpty(), 5. Top, 6. Display, 0. Exit
2
Popped element was: 33
Enter any of the following options:
1.Push, 2. Pop, 3. Size, 4. isEmpty(), 5. Top, 6. Display, 0. Exit
6
22
3
5
Enter any of the following options:
1.Push, 2. Pop, 3. Size, 4. isEmpty(), 5. Top, 6. Display, 0. Exit
3
Size of the stack is: 3
Enter any of the following options:
1.Push, 2. Pop, 3. Size, 4. isEmpty(), 5. Top, 6. Display, 0. Exit
2
Popped element was: 22
Enter any of the following options:
1.Push, 2. Pop, 3. Size, 4. isEmpty(), 5. Top, 6. Display, 0. Exit
2
Popped element was: 3
Enter any of the following options:
1.Push, 2. Pop, 3. Size, 4. isEmpty(), 5. Top, 6. Display, 0. Exit
2
Popped element was: 5
Enter any of the following options:
1.Push, 2. Pop, 3. Size, 4. isEmpty(), 5. Top, 6. Display, 0. Exit
4
Stack is empty
Enter any of the following options:
1.Push, 2. Pop, 3. Size, 4. isEmpty(), 5. Top, 6. Display, 0. Exit
0


Process finished with exit code 0
```

Queue:

```
"C:\Users\Sarvagnya Purohit9\.jdks\openjdk-17\bin\java.exe" "-javaagent:C:\Users\Sarvagnya
 Purohit9\AppData\Local\JetBrains\Toolbox\apps\IDEA-U\ch-0\212.5284.40\lib\idea_rt.jar=56296:C:\Users\Sarvagnya
 Purohit9\AppData\Local\JetBrains\Toolbox\apps\IDEA-U\ch-0\212.5284.40\bin" -Dfile.encoding=UTF-8 -classpath "D:\College
 Resources\Submissions\Assignment\DSA\Lab Assignment 3 Queue\out\production\untitled104" com.company.Main
Enter size of the queue:
6
1.Size, 2.isEmpty(), 3.Enqueue, 4.First, 5.Dequeue, 6.Display, 0.Exit
2
Queue is empty
1.Size, 2.isEmpty(), 3.Enqueue, 4.First, 5.Dequeue, 6.Display, 0.Exit
3
Enter element to enqueue
22
1.Size, 2.isEmpty(), 3.Enqueue, 4.First, 5.Dequeue, 6.Display, 0.Exit
3
Enter element to enqueue
33
1.Size, 2.isEmpty(), 3.Enqueue, 4.First, 5.Dequeue, 6.Display, 0.Exit
3
Enter element to enqueue
44
1.Size, 2.isEmpty(), 3.Enqueue, 4.First, 5.Dequeue, 6.Display, 0.Exit
```

```
6
22  33  44
1.Size, 2.isEmpty(), 3.Enqueue, 4.First, 5.Dequeue, 6.Display, 0.Exit
1
Size of the queue is: 3
1.Size, 2.isEmpty(), 3.Enqueue, 4.First, 5.Dequeue, 6.Display, 0.Exit
4
First element is 22
1.Size, 2.isEmpty(), 3.Enqueue, 4.First, 5.Dequeue, 6.Display, 0.Exit
3
Enter element to enqueue
55
1.Size, 2.isEmpty(), 3.Enqueue, 4.First, 5.Dequeue, 6.Display, 0.Exit
3
Enter element to enqueue
66
1.Size, 2.isEmpty(), 3.Enqueue, 4.First, 5.Dequeue, 6.Display, 0.Exit
3
Enter element to enqueue
77
1.Size, 2.isEmpty(), 3.Enqueue, 4.First, 5.Dequeue, 6.Display, 0.Exit
3
```

```
Enter element to enqueue
88
The queue is full
1.Size, 2.isEmpty(), 3.Enqueue, 4.First, 5.Dequeue, 6.Display, 0.Exit
1
Size of the queue is: 6
1.Size, 2.isEmpty(), 3.Enqueue, 4.First, 5.Dequeue, 6.Display, 0.Exit
6
22  33  44  55  66  77
1.Size, 2.isEmpty(), 3.Enqueue, 4.First, 5.Dequeue, 6.Display, 0.Exit
4
First element is 22
1.Size, 2.isEmpty(), 3.Enqueue, 4.First, 5.Dequeue, 6.Display, 0.Exit
3
Enter element to enqueue
2
The queue is full
1.Size, 2.isEmpty(), 3.Enqueue, 4.First, 5.Dequeue, 6.Display, 0.Exit
5
Dequeued element is 22
1.Size, 2.isEmpty(), 3.Enqueue, 4.First, 5.Dequeue, 6.Display, 0.Exit
5
```

```
Dequeued element is 33
1.Size, 2.isEmpty(), 3.Enqueue, 4.First, 5.Dequeue, 6.Display, 0.Exit
6
44  55  66  77
1.Size, 2.isEmpty(), 3.Enqueue, 4.First, 5.Dequeue, 6.Display, 0.Exit
5
Dequeued element is 44
1.Size, 2.isEmpty(), 3.Enqueue, 4.First, 5.Dequeue, 6.Display, 0.Exit
5
Dequeued element is 55
1.Size, 2.isEmpty(), 3.Enqueue, 4.First, 5.Dequeue, 6.Display, 0.Exit
1
Size of the queue is: 2
1.Size, 2.isEmpty(), 3.Enqueue, 4.First, 5.Dequeue, 6.Display, 0.Exit
6
66  77
1.Size, 2.isEmpty(), 3.Enqueue, 4.First, 5.Dequeue, 6.Display, 0.Exit
4
First element is 66
1.Size, 2.isEmpty(), 3.Enqueue, 4.First, 5.Dequeue, 6.Display, 0.Exit
5
```

```
Dequeued element is 66
1.Size, 2.isEmpty(), 3.Enqueue, 4.First, 5.Dequeue, 6.Display, 0.Exit
5
Dequeued element is 77
1.Size, 2.isEmpty(), 3.Enqueue, 4.First, 5.Dequeue, 6.Display, 0.Exit
2
Queue is empty
1.Size, 2.isEmpty(), 3.Enqueue, 4.First, 5.Dequeue, 6.Display, 0.Exit
0

Process finished with exit code 0
```

Conclusion:

In this lab assignment, we learnt about making Java APIs by bundling interfaces and classes implementing those interfaces. We also learnt about how we can make generic classes and extend the same class to work for all reference types. Most importantly, we learnt about the 2 powerful linear data structures – stack and queue. The LIFO nature of the stack makes it a natural choice for being used in many real-world applications (like bracket pair matching which is crucial in verifying syntax of any code). The basic operations related to stack like push() and pop() were also studied. Similarly, the FIFO nature of the queue allows us to mimic queue-based and first-come priority-based applications (like the robin round process scheduler). Basic queue operations on enqueue() and dequeue() were implemented. We also now understand the limitations of a queue and how a circular queue can improve the performance as opposed to a normal queue, by using the available size efficiently. We can now use the API created in this assignment and apply it to any application based on stacks and queues.