Merit-Quality-Excellence

Sukkur IBA University Khairpur Campus

Data Structures

LAB No: 04

Instructor: Marina Gul

Objective of Lab No. 4:
After performing lab4, students will be able to:

- Stack
- Queue

Exercise

1. **Stack using array**: Understand provided code and implement all required methods in Stack. Stack Code is given below:

```
class Stack
{
    private int arr[];
    private int top;
    private int capacity;

    // Constructor to initialize stack
    Stack(int size)
    {
        arr = new int[size];
        capacity = size;
        top = -1;
    }

    // Utility function to add an element x in the stack and
check for stack overflow
    public void push(int x)
    {
        // Write your code here
    }

    // Utility function to pop top element from the stack and
check for stack underflow
    public int pop()
    {
        // Write your code here
    }

    // Utility function to return top element in a stack
    public int top()
    {
        // Write your code here
    }

    // Utility function to return the size of the stack
    public int size()
    {
        // Write your code here
    }

    // Utility function to check if the stack is empty or not
```

```java
    public Boolean isEmpty()
    {
        // Write your code here
    }

    // Utility function to check if the stack is full or not
    public Boolean isFull()
    {
        // Write your code here
    }

    public static void main (String[] args)
    {
        Stack stack = new Stack(3);

        stack.push(1);      // Inserting 1 in the stack
        stack.push(2);      // Inserting 2 in the stack

        stack.pop();        // removing the top 2
        stack.pop();        // removing the top 1

        stack.push(3);      // Inserting 3 in the stack

        System.out.println("Top element is: " + stack.peek());
        System.out.println("Stack size is " + stack.size());

        stack.pop();        // removing the top 3

        // check if stack is empty
        if (stack.isEmpty())
            System.out.println("Stack Is Empty");
        else
            System.out.println("Stack Is Not Empty");
    }
}
```

After implementing all the methods, run the code. Your output should be like as follows:

```
Inserting 1
Inserting 2
Removing 2
Removing 1
Inserting 3
Top element is: 3
Stack size is 1
Removing 3
Stack Is Empty
```

2. **<u>Stack using Linked list</u>:** Understand provided code and implement all required methods in Stack. Stack Code is given below:

```
// A linked list node
class Node
{
    int data;        // integer data
    Node next;       // pointer to the next node
};

class Stack
{
    private Node top;

    public Stack() {
        this.top = null;
    }

    // Utility function to add an element x in the stack
    public void push(int x) // insert at the beginning
    {
        // Write your code here
    }

    // Utility function to check if the stack is empty or not
    public boolean isEmpty()
    {
        // Write your code here
    }
```

```java
    // Utility function to return top element in a stack
    public int top()
    {
        // Write your code here
    }

    // Utility function to pop top element from the stack and
check for Stack underflow
    public void pop() // remove at the beginning
    {
        // Write your code here
    }
}

class StackImpl
{
    public static void main(String[] args)
    {
        Stack stack = new Stack();

        stack.push(1);
        stack.push(2);
        stack.push(3);

        System.out.println("Top element is " +  stack.peek());

        stack.pop();
        stack.pop();
        stack.pop();

        if (stack.isEmpty()) {
            System.out.print("Stack is empty");
        } else {
            System.out.print("Stack is not empty");
        }
    }
}
```

After implementing all the methods, run the code. Your output should be like as follows:

```
Inserting 1
Inserting 2
Inserting 3
Top element is 3
Removing 3
Removing 2
Removing 1
Stack is empty
```

3: **Queue using array:** Understand provided code and implement all required methods in Queue. Queue Code is given below:

```java
// Class for queue
class Queue
{
    private int arr[];
    private int front;
    private int rear;
    private int capacity;
    private int count;

    // Constructor to initialize queue
    Queue(int size)
    {
        arr = new int[size];
        capacity = size;
        front = 0;
        rear = 0;
        count = 0;
    }

    // Utility function to remove front element from the queue
and check for Queue Underflow
    public void dequeue()
    {
        // Write your code here
    }

    // Utility function to add an item to the queue and check
for queue overflow
    public void enqueue(int item)
    {
        // Write your code here
    }

    // Utility function to return front element in the queue and
check for Queue Underflow
    public int peek()
    {
        // Write your code here
    }

    // Utility function to return the size of the queue
    public int size()
    {
        // Write your code here
    }

    // Utility function to check if the queue is empty or not
    public Boolean isEmpty()
    {
        // Write your code here
    }

    // Utility function to check if the queue is empty or not
    public Boolean isFull()
    {
        // Write your code here
    }
}

class Main
{
    // main function
    public static void main (String[] args)
    {
        // create a queue of capacity 5
        Queue q = new Queue(5);

        q.enqueue(1);
        q.enqueue(2);
        q.enqueue(3);

        System.out.println("Front element is: " + q.peek());
        q.dequeue();
        System.out.println("Front element is: " + q.peek());

        System.out.println("Queue size is " + q.size());
        q.dequeue();
        q.dequeue();

        if (q.isEmpty())
            System.out.println("Queue Is Empty");
        else
            System.out.println("Queue Is Not Empty");
    }
}
```

After implementing all the methods, run the code. Your output should be like as follows:

```
Inserting 1
Inserting 2
Inserting 3
Front element is: 1
Removing 1
Front element is: 2
Queue size is 2
Removing 2
Removing 3
Queue Is Empty
```

4. **Queue using Linked list:** Understand provided code and implement all required methods in Queue. Queue Code is given below:

```
// A linked list node
class Node
{
    int data;       // integer data
    Node next;      // pointer to the next node

    public Node(int data)
    {
        // set the data in allocated node and return the node
        this.data = data;
        this.next = null;
    }
}

class Queue
{
    private static Node rear = null, front = null;
    // Utility function to remove front element from the queue
and check for Queue Underflow
    public static int dequeue()      // delete at the beginning
    {
        // Write your code here
    }

    // Utility function to add an item in the queue
    public static void enqueue(int item)     // insertion at the
end
    {
        // Write your code here
    }

    // Utility function to return top element in a queue
    public static int peek()
    {
        // Write your code here
    }

    // Utility function to check if the queue is empty or not
    public static boolean isEmpty()
    {
        // Write your code here
    }
}

class Main {
    public static void main(String[] args)
    {
        Queue q = new Queue();
        q.enqueue(1);
        q.enqueue(2);
        q.enqueue(3);
        q.enqueue(4);

        System.out.printf("Front element is %d\n", q.peek());

        q.dequeue();
        q.dequeue();
        q.dequeue();
        q.dequeue();

        if (q.isEmpty()) {
            System.out.print("Queue is empty");
        } else {
            System.out.print("Queue is not empty");
            }
        }
}
```

After implementing all the methods, run the code. Your output should be like as follows:

```
Inserting 1
Inserting 2
Inserting 3
Inserting 4
Front element is 1
Removing 1
Removing 2
Removing 3
Removing 4
Queue is empty
```

5. **Queue using two Stacks:** Understand provided code and implement all required methods in Queue Class. Sample Code is given below:

```
// Implement Queue using two stacks
class Queue {
    private Stack s1, s2;

    // Constructor
    Queue() {
        s1 = new Stack();
        s2 = new Stack();
    }

    // Enqueue an item to the queue
    public void enqueue(int data)
    {
        // Write your code here
    }

    // Dequeue an item from the queue
    public int dequeue()
    {
        // Write your code here
    }
```

```java
    public static void main(String[] args) {
        int[] keys = { 1, 2, 3, 4, 5 };
        Queue q = new Queue();

        // insert above keys
        for (int key : keys) {
            q.enqueue(key);
        }

        System.out.println(q.dequeue());    // print 1
        System.out.println(q.dequeue());    // print 2
    }
}
```

6. Think about the inverse of task 05 (Stack using queue) and implement all the required methods.