Lab Exercise 5

Implement Linear Queue using user defined exception handling (also use 'throw' and 'throws' keyword)

Theory: In java user can create their own exception class and throw that exception using throw keyword. These exceptions are known as **user-defined** or **custom** exceptions. Exception classes can be created by extending the Exception class.The extended class contains constructors, data members and methods like any other class. The throw and throws keywords are used while implementing the user-defined exceptions.

The **throw** keyword in Java is used to explicitly throw an exception from a method or any block of code. We can throw either checked or unchecked exception. The throw keyword is mainly used to throw custom exceptions.

Syntax: **throw *Instance***

Example:

throw new ArithmeticException("/ by zero");

The **throws** is a keyword in Java which is used in the signature of method to indicate that this method might throw one of the listed type exceptions. The caller to these methods has to handle the exception using a try-catch block.

Syntax: **type method_name(parameters) throws exception_list**

exception_list is a comma separated list of all the exceptions which a method might throw.

Program:

```
class LinearQueue extends Exception        //user-defined exception class "LinearQueue"
{
   int front,rear;
   final int size=5;
   int q[]=new int[size];

   LinearQueue()
   {
      front=0;
      rear=-1;
   }

   LinearQueue(String s)
```

```
{
   super(s);     //calling constructor of super class Exception
}

void enqueue(int ele) throws LinearQueue
{
   try {
      if (rear == -1 || rear < size-1) {
         q[++rear] = ele;
         System.out.println("Q has "+q[rear]+ " at pos "+rear);
      }
      else {
         throw new LinearQueue("Queue is full\n");
      }
   }
  catch(LinearQueue lq) {
      System.out.println(lq.getMessage());
   }
}

void dequeue() throws LinearQueue
{
   try {
      if (front < size) {
         System.out.println("Deleted element is " + q[front++]);
      }
      else {
         if(front==size)
            front=-1;
         throw new LinearQueue("No elements to delete");
      }
   }
   catch(LinearQueue lq)
   {
      System.out.println(lq.getMessage());
   }
}
```

```
    void display()
    {
        if(front>-1)
        {
            System.out.println("Elements are");
            int k=front;
            for(int i=k;i<=rear;i++)
                System.out.println(q[k++]);
        }
    }     }

public class Six
{
    public static void main(String args[]) throws LinearQueue
    {
        LinearQueue q=new LinearQueue();
        q.enqueue(10);
        q.enqueue(20);
        q.enqueue(30);
        q.enqueue(40);
        q.enqueue(50);
        q.display();
        System.out.println("When tried to put sixth element to full queue");
        q.enqueue(60);
        q.dequeue();
        q.dequeue();
        q.dequeue();
        q.display();
        q.dequeue();
        q.dequeue();
        System.out.println("When tried to remove sixth element from empty queue");
        q.dequeue();
        q.display();
    }
}
```