```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package sq_exercises;

import stacksandqueues.*;

/**
 *
 * @author ogm
 * @param <E>
 */
public class QueueFromDequeImpl<E> implements MyQueueIF<E> {

   MyDequeIF<E> myDeque;
   int head;
   int tail;
   int size;
   E[] array;

   public QueueFromDequeImpl(E[] array) {
      myDeque = new MyDequeImpl<>(array);

   }

   @Override
   public void enqueue(E element) throws FullStructureException {
      myDeque.insertRight(element);
      //throw new UnsupportedOperationException("Not supported yet.");
   }

   @Override
   public E dequeue() throws EmptyStructureException {
      return myDeque.removeLeft();
      //throw new UnsupportedOperationException("Not supported yet.");
   }

   @Override
   public E peek() throws EmptyStructureException {
      return myDeque.peekLeft();
      //throw new UnsupportedOperationException("Not supported yet.");
   }

   @Override
```

```
    public boolean isEmpty() {
        return myDeque.isEmpty();
        //throw new UnsupportedOperationException("Not supported yet.");
    }

    @Override
    public boolean isFull() {
        return myDeque.isFull();
        //return(size== array.length);
        //throw new UnsupportedOperationException("Not supported yet.");
    }

    @Override
    public void display() {
        myDeque.display();
        //throw new UnsupportedOperationException("Not supported yet.");
    }
}
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package sq_exercises;

import java.util.logging.Level;
import java.util.logging.Logger;
import stacksandqueues.*;

/**
 *
 * @author ogm2
 * @param
 */
public class ArithmeticExpressionEvaluator {
   // myStack = new MyStackImpl<>(array);

    MyStackIF<Double> myStackVals;
    MyStackIF<Character> myStackOps;
    Double[] array1=new Double[1000];
    Character[] array= new Character[1000];
    //Character[] str=new Character[1000];
    int size;
    ////private Object myStackVals;

    public ArithmeticExpressionEvaluator()
```

```
  {
    myStackVals= new MyStackImpl<>(array1);
    myStackOps= new MyStackImpl<>(array);

  }

  public Double evaluateArithmeticExpression(String s) throws
EmptyStructureException, FullStructureException {
      char[] str=s.toCharArray();
      int i=0;
      //char el=s.charAt(i);
      while(i<str.length){


          //char el=s.charAt(i);
          //(str[i]>='1') && (str[i]<='9')

//str[i]=='1'||str[i]=='2'||str[i]=='3'||str[i]=='4'||str[i]=='5'||str[i]=='6'||str[i]=='7'||s
tr[i]=='8'||str[i]=='9'
          if((str[i]>='1') && (str[i]<='9')){
            myStackVals.push((double)str[i]-48);
            i++;
          }
          else if ((str[i]=='+')||(str[i]=='-')||(str[i]=='*')||(str[i]=='/'))
          {
            myStackOps.push(str[i]);
            i++;
          }
          else if(str[i]==')'){
            int x=i;

            while(str[x]!='('
              x--;
            while(str[x]=='('){
              //if(myStackVals.isEmpty()|| myStackOps.isEmpty())
                //throw new(EmptyStructureException)
              double num1=(myStackVals.pop());
              double num2=(myStackVals.pop());
              char op=myStackOps.pop();
              double newVal=0;
              switch (op) {
                case '+':
                  newVal=((num2)+(num1));
                  break;
                case '-':
                  newVal=((num2)-(num1));
```

```
                    break;
                case '/':
                    newVal=((num2)/(num1));
                    break;
                case '*':
                    newVal=((num2)*(num1));
                    break;
                default:
                    break;
            }
            //myStackVals.pop();
            //myStackVals.pop();
            myStackVals.push(newVal);
            str[x]='!';
            i++;
          }
        }
        else
          i++;
    }
    return myStackVals.pop();
  }
}


/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package sq_exercises;

import stacksandqueues.EmptyStructureException;
import stacksandqueues.FullStructureException;


public class DoubleStackImpl<E> implements DoubleStackIF<E> {

  E[] array;
  int sizeIn, sizeOut;
  int elIn;
  int elOut;

  public DoubleStackImpl(E[] array) {
    this.array = array;
    this.elIn=(array.length)/2;
```

```
    this.elOut=(array.length/2)+1;
}

@Override
public void pushIn(E element) throws FullStructureException {

    if(this.isFull())
        throw new FullStructureException("The array is full");
    else if(elIn==0){
        array[elIn]=element;
        elIn=array.length-1;
        sizeIn++;
    }
    else{
        array[elIn]=element;
        elIn--;
        sizeIn++;
    }

    //throw new UnsupportedOperationException("Not supported yet.");
}

@Override
public E popIn() throws EmptyStructureException {
    if(this.isEmptyIn())
        throw new EmptyStructureException("This stack is empty");
    else if(elIn==array.length-1){
        sizeIn--;
        elIn=0;
        return(array[array.length-1]);
    }
    else{
        sizeIn--;
        elIn++;
        return(array[elIn-1]);
    }
    //throw new UnsupportedOperationException("Not supported yet.");
}

@Override
public void pushOut(E element) throws FullStructureException {
    if(this.isFull())
        throw new FullStructureException ("This stack is full");
    else if(elOut==array.length-1){
        array[elOut]=element;
        elOut=0;
```

```
        sizeOut++;
    }
    else{
        array[elOut]=element;
        elOut++;
        sizeOut++;
    }
    //throw new UnsupportedOperationException("Not supported yet.");
}

@Override
public E popOut() throws EmptyStructureException {
    if(this.isEmptyOut())
        throw new EmptyStructureException("The stack is empty");
    else if(elOut==0){
        elOut=array.length-1;
        sizeOut--;
        return(array[0]);
    }
    else{
        elOut--;
        sizeOut--;
        return(array[elOut+1]);
    }
    //throw new UnsupportedOperationException("Not supported yet.");
}

public boolean isEmpty() {
    return ((sizeOut+sizeIn)==0);
    //throw new UnsupportedOperationException("Not supported yet.");
}

@Override
public boolean isFull() {
    return (sizeOut+sizeIn==array.length);
    //throw new UnsupportedOperationException("Not supported yet.");
}

@Override
public void display() {
    System.out.println("Stack Out: ");
    int i=0;
    int x=0;
    int elPrintO=(array.length/2)+1;
    int elPrintI=(array.length/2);
    while(i<=sizeOut){
```

```
        if(elPrintO==array.length-1){
            System.out.print(array[elPrintO]);
            elPrintO=0;
            i++;
        }
        else{
            System.out.print(array[elPrintO]);
            elPrintO++;
            i++;
        }
    }
    System.out.println();
    System.out.println("Stack In: ");
    while(x<=sizeIn){
        if(elPrintI==0){
            System.out.print(array[elPrintI]);
            elPrintI=array.length-1;
            x++;
        }
        else{
            System.out.print(array[elPrintI]);
            elPrintI--;
            x++;
        }
    }
    System.out.println();
    //throw new UnsupportedOperationException("Not supported yet.");
}

@Override
public E peekIn() throws EmptyStructureException {
    if(this.isEmptyIn())
        throw new EmptyStructureException("There are no elements");
    else
        return this.array[elIn];
    //throw new UnsupportedOperationException("Not supported yet.");
}

@Override
public boolean isEmptyIn() {
    return(sizeIn==0);
    //throw new UnsupportedOperationException("Not supported yet.");
}

@Override
public E peekOut() throws EmptyStructureException {
```

```
        if(this.isEmptyOut())
            throw new EmptyStructureException("There are no elements");
        else
            return this.array[elOut];
        //throw new UnsupportedOperationException("Not supported yet.");
    }

    @Override
    public boolean isEmptyOut() {
        return(sizeOut==0);
        //throw new UnsupportedOperationException("Not supported yet.");
    }

}

package sq_exercises;

import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;
import java.util.logging.Level;
import java.util.logging.Logger;
import stacksandqueues.*;

/**
 *
 * @author ogm2
 */
public class AgesOfHollywood {

    String[] array1;
    String[] array = new String[1000];
    HollywoodCelebrity holly=new HollywoodCelebrity();
    MyPriorityQueueIF<String> queue;


    public AgesOfHollywood(){
        queue=new MyPriorityQueueImpl<>(array);
        //throw new UnsupportedOperationException("Not supported yet.");
    }

    public void parseTextFile(String pathname) throws FileNotFoundException {
        int count=0;
        String l="";
        String m="";
        int n=0;
```

```
        try{
           Scanner sc= new Scanner(new File(pathname));
              while(sc.hasNext()){
                 array[count]=sc.next();
                 count++;
               //sc.nextLine();
               }
        }
         catch (FileNotFoundException e){

//Logger.getLogger(StacksAndQueuesLauncher.class.getName()).log(Level.SEVERE,
null, ex);
        }
        //Scanner scan=new Scanner(new File(pathname));
        //array1=pathname.split(" ");
        for(int x=0; x<(count/3); x++){
           int i=1;

           //while(i<3)

              if(i%3==1)
              {
                 holly.setFirstName(array[x]);
                 i++;
                 l=holly.getFirstName();
              }
              if(i%3==2)
              {
                 holly.setLastName(array[x]);
                 i++;
                 m=holly.getLastName();
              }
              if(i%3==0)
              {
                 holly.setYearOfBirth(Integer.parseInt(array[x]));
                 i++;
                 n=holly.getYearOfBirth();
              }
           try {
                 queue.insert((l+m),n);
              } catch (FullStructureException ex) {}
           x++;
          queue.display();
        }

        queue.display();
```

```
    }
}
```