```
/**
 * MyQueue Class Run-time Analysis
 * @author Tan Pham
 * Created 01/10/2015
 * ********************************************************************************
 *
 */

Note: constant = cost a constant time.
    amotized constant = cost an amotized constant time.
    linear = cost linear time.



**********************************************************************************
import java.util.ArrayList;


public class MyQueue {

    public ArrayList <Student> queue;

    boolean open = false;



**********************************************************************************
    public MyQueue(){
        queue = new ArrayList <Student>(); // constant
    }

    Total time = constant




**********************************************************************************
    public boolean isEmpty(){
        return queue.isEmpty();  // constant
    }

    Total time = constant




**********************************************************************************
```

```
public void offer(Student std){ // constant
      queue.add(std);        // amortized constant
}
```

Total time = constant + amortized constant time
= amortized constant time

****

```
public void offer(int index, Student std){  // constant
      queue.add(index,std);             // amortized constant
}
```

Total time = constant + amortized constant time
= amortized constant time

****

```
public Student poll(){
      return queue.remove(0);        //  linear
}
```

Total time = linear time

****

```
public Student peek(){
      return queue.get(0);          // constant
}
```

Total time = constant time

****

```
public MyQueue split(){
      MyQueue newQueue = new MyQueue();                  //constant1
```

```
        for (int i = queue.size()-1; i >= 0; i--){              //constant2
            if (i%2==1){                                //constant3
                newQueue.offer(0,queue.remove(i));      //amotized constant + linear
            }
        }
        return newQueue;                                //constant4
    }
```

Since the for loop goes for n times, so:
Total time = constant1 +n*(constant2 + constant3 + amotized constant + linear) + constant4
= constant5 + n*constant6 + n*(amotized constant) + n*linear
= constant5 + O(n)*constant6 + O(n)*(amotized constant) + O(n)*O(n)
//Since linear = O(n)
= O(n^2)

So total time for this method is O(n^2)

****

```
    public int size(){
        return queue.size();                //constant
    }
```

Total time = constant

****

```
    public String toString(){
        if(queue.size()==0) return "[]";                    // constant1
        String str = "[";                               // constant2

        for (int i = 0; i < queue.size(); i++){             // constant3
            str += queue.get(i).toString() + ", ";          // constant4
        }

        str = str.substring(0, str.length()-2) + "]";       // linear
        return str;
    }
}
```

Since the for loop goes n times,so:
Total time = constant1 + constant2 + n*(constant3+constant4) + linear
= constant5 + O(n)*constant6 + O(n)
= O(n)

So the total time for this method is O(n)