

Stacks & Queues Competency #6

Stacks	Mastered 1 point	Progressing 0.001 points	Novice 0 points	Criterion Score
Can design and implement a stack using an array				/ 1
Can design and implement a stack using a linked list				/ 1

Queues	Mastered 1 point	Progressing 0.001 points	Novice 0 points	Criterion Score
Can design and implement a queue using a circular array				/ 1
Can design and implement a queue using a linked list				/ 1
Can design and implement a priority queue using a heap				/ 1

LinkQueue.java X

```
1 package project;
2
3 public class LinkQueue<E> {
4     private Node<E> head;
5     private Node<E> tail;
6     private int manyNodes;
7
8     public LinkQueue() {}
9
10    public void enqueue(E e) {
11        if(manyNodes == 0) {
12            head = new Node<E>(e, null);
13            tail = head;
14        } else {
15            tail.setLink(head);
16            tail = tail.getLink();
17        }
18        manyNodes++;
19    }
20
21    public E dequeue() {
22        E element = head.getData();
23        head = head.getLink();
24        manyNodes--;
25        if(manyNodes == 0) {
26            tail = null;
27        }
28        return element;
29    }
30
31    public E peek() {
32        if(isEmpty()) {
33            System.out.println("Is empty!");
34        }
35        return head.getData();
36    }
37
38    public boolean isEmpty() {
39        if(head == null) {
40            return true;
41        } else {
42            return false;
43        }
44    }
45 }
46
```

LinkStack.java X

```
1 package project;
2
3 public class LinkStack<E> {
4     private Node<E> top;
5     private int manyNodes;
6
7     public LinkStack() {
8         top = null;
9         manyNodes = 0;
10    }
11
12    public void push (E e) {
13        top = new Node<E>( e, top);
14        manyNodes++;
15    }
16
17    public E pop() {
18        E element = top.getData();
19        top = top.getLink();
20        manyNodes--;
21        return element;
22    }
23
24    public E peek() {
25        if(isEmpty()) {
26            System.out.println("Is empty!");
27        }
28        return top.getData();
29    }
30
31    public boolean isEmpty() {
32        if(top == null) {
33            return true;
34        } else {
35            return false;
36        }
37    }
38
39
40
41 }
42
```

ArrayQueue.java X

```
1 package project;
2
3 public class ArrayQueue<E> {
4     private Object[] data;
5     private int head;
6     private int manyItems;
7
8     public ArrayQueue(int num) {
9         data = new Object[num];
10        manyItems = 0;
11    }
12
13    public void enqueue(E e) {
14        data[(head+manyItems) % data.length] = e;
15        manyItems++;
16    }
17
18    public E dequeue() {
19        E temp = (E) data[head];
20        head = (head + 1) % data.length;
21        manyItems--;
22        return temp;
23    }
24
25    public E peek() {
26        if(isEmpty()) {
27            System.out.println("Is empty!");
28        }
29        return (E) data[head];
30    }
31
32    public boolean isEmpty() {
33        if(data == null) {
34            return true;
35        } else {
36            return false;
37        }
38    }
39 }
40
```

```
ArrayStack.java X
1 package project;
2
3 public class ArrayStack<E> {
4     private Object[] data;
5     private int top;
6
7     public ArrayStack(int num) {
8         data = new Object[num];
9         top = 0;
10    }
11
12    public void push (E e) {
13        data[top++] = e;
14    }
15
16    public E pop() {
17        if(isEmpty()) {
18            System.out.println("Is empty!");
19        }
20        return (E) data[--top];
21    }
22
23    public E peek() {
24        if(isEmpty()) {
25            System.out.println("Is empty!");
26        }
27        return (E) data[top];
28    }
29
30    public boolean isEmpty() {
31        if(data == null) {
32            return true;
33        } else {
34            return false;
35        }
36    }
37 }
38
```

QueueDriver.java X

```
1 package project;
2
3 public class QueueDriver {
4
5     public QueueDriver() {}
6
7     public static void main(String[] args) {
8         ArrayQueue<Integer> numbers = new ArrayQueue<>(5);
9         numbers.enqueue(34);
10        numbers.enqueue(56);
11        numbers.dequeue();
12        System.out.println(numbers.peek());
13        System.out.println(numbers.isEmpty());
14
15        LinkQueue<String> messages = new LinkQueue<>();
16        messages.enqueue("Hello");
17        messages.enqueue("World");
18        messages.dequeue();
19        System.out.println(messages.peek());
20        System.out.println(messages.isEmpty());
21    }
22
23 }
24
```

Console X

<terminated> QueueDriver [Java Application] /Users/cindy/.p2/pool/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.macosx.x

```
56
false
Hello
false
```

```
Run Last Tool X
1 package project;
2
3 public class StackDriver {
4
5     public StackDriver() {
6         // TODO Auto-generated constructor stub
7     }
8
9     public static void main(String[] args) {
10         ArrayStack<Integer> numbers = new ArrayStack<Integer>(5);
11         numbers.push(1);
12         numbers.push(2);
13         numbers.pop();
14         System.out.println(numbers.peek());
15         System.out.println(numbers.isEmpty());
16
17         LinkStack<String> messages = new LinkStack<String>();
18         messages.push("Hello");
19         messages.push("World");
20         messages.pop();
21         System.out.println(messages.peek());
22         System.out.println(messages.isEmpty());
23
24     }
25
26 }
27
```

```
Console X
<terminated> StackDriver [Java Application] /Users/cindy/.p2/pool/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.macosx.x86_6
2
false
Hello
false
```