EX 3.4 Hand trace a stack X through the following operations:

```
X.push(new Integer(4));
X.push(new Integer(3));
Integer Y = X.pop();
X.push(new Integer(7));
X.push(new Integer(2));
X.push(new Integer(5));
X.push(new Integer(9));
Integer Y = X.pop();
X.push(new Integer(3));
X.push(new Integer(9));
```

EX 3.4 Answer

4
4, 3
4
4, 7
4, 7, 2
4, 7, 2, 5
4, 7, 2, 5, 9
4, 7, 2, 5
4, 7, 2, 5, 3
4, 7, 2, 5, 3, 9

EX 3.5 Given the resulting stack X from the previous exercise, what would be the result of each of the following?

```
a. Y = X.peek();
b. Y = X.pop();
   Z = X.peek();
c. Y = X.pop();
   Z = X.peek();
```

EX 3.5 Answer

X = 4, 7, 2, 5
Y = 3
Z = 5

EX 4.1 Explain what will happen if the steps depicted in Figure 4.4 are reversed.

EX 4.1 Answer

If you reset the front reference before setting to point the added node to the current first node, you would lose the reference to the existing list, so that you wouldn't be able to retrieve elements from the list.

EX 4.2 Explain what will happen if the steps depicted in Figure 4.5 are reversed.

EX 4.2 Answer

If you reset the reference of the current node before setting to point the added node to the following node which refers current, you would lose the reference to the following nodes, so that you wouldn't be able to retrieve the elements.

EX 4.4 Write an algorithm for the add method that will add at the end of the list instead of the beginning. What is the time complexity of this algorithm?

EX 4.4 Answer

```
add(E o) {
   // enqueue
   count the nodes in the list // O(n)
   find the last node of which index is the number of the list minus 1 // O(1)
   add the method parameter o to the rear of which index is the number of the list // O(1)
   // dequeue
   get the value of the front node in the list and store it in an variable // O(1)
   remove the front node // O(1)
   reassign all the indices of nodes // O(n)
   return the variable // O(1)
}
```

Time complexity: O(n)

EX 4.5 Modify the algorithm from the previous exercise so that it makes use of a rear reference. How does this affect the time complexity of this and the other operations?

EX 4.5 Answer

```
add(E o) {
   // enqueue
   find the rear node of which the reference is null // O(n)
   the next reference of the rear is set to point to the method parameter o // O(1)
   // dequeue
   get the value of the front node in the list and store it in an variable // O(1)
   the reference to the fiont node is reset to point to the second node // O(1)
   remove the first node // O(1)
```

```
    return the variable // O(1)
}
```

Time complexity: O(n)