Objective: data abstraction.

Assignment: The goal of this assignment is to implement the methods in the class `MyNode` according to the given specification. The class `MyNode` implements the link list data structure. A link list is a possibly empty sequence of nodes. In our implementation of a link list, each node contains data (an integer) as well as a reference to the next node in the sequence. A pictorial representation of our implementation of a link list of [1, 2, 3] is shown below.

Part of the code for the class `MyNode` is given below. The specification for each of the method which you have to complete is given in the comments.

```java
class MyNode {
    private int data;
    private MyNode next;

    MyNode(int a) {
        data = a;
        next = null;
    }

    MyNode(int a, MyNode n) {
        data = a;
        next = n;
    }

    // prints out the contents of this list from here onwards
```
public String toString() {
    // to fill in
}

// set the value of data to a
public void setData(int a) {
    // to fill in
}

// set the value of next to l
public void setNext(MyNode l) {
    // to fill in
}

// returns the value of data
public int getData() {
    // to fill in
}

// returns the next node in the sequence
public MyNode getNext() {
    // to fill in
}

// adds the link list l to the end of this list
public void add(MyNode l) {
    return next;
}

// returns the size of the current list.
// For example, a list [1,2,3] has size 3.
public int size() {
    // to fill in
}

// reverses the current list.
// The first node of the reversed list is returned.
public MyNode reverse() {
    // to fill in
}
}

There are two constructors for MyNode. For example, the following code fragment will create the link list [1,2,3].

MyNode l1 = new MyNode(1, new MyNode(2, new MyNode(3)));
The method **toString** returns a string representation of the linklist. For example,

```java
System.out.println(l1.toString());
```

or

```java
System.out.println(l1);
```

prints `1 2 3`. The methods **setData**(*a*) and **setNext**(*l*) are used to set **data** and **next** to the values *a* and *l* respectively. The methods **getData** and **getNext** are used to get the values of **data** and **next** respectively. For example,

```java
l1.setData(5);
System.out.println(l1);
MyNode l2 = new MyNode(5, new MyNode(6));
l1.setNext(l2);
System.out.println(l1);
System.out.println(l1.getNext());
```

will print `5 2 3, 5 5 6` and `5 6` respectively. The method **add**(*l*) appends *l* to the end of the current list. For example,

```java
MyNode l3 = new MyNode(7, new MyNode(8));
l1.add(l3);
System.out.println(l1);
```

will print `5 5 6 7 8`. The method **size** returns the size of the currently list. For example,

```java
System.out.println(l1.size());
System.out.println(l3.size());
```

will print `5` and `2` respectively. The method **reverse** reverses the order of elements in the current list by reversing **next** of each Node. For example,

```java
MyNode l4 = l3.reverse();
System.out.println(l3);
System.out.println(l4);
System.out.println(l1);
```

will print `7 8, 8 7` and `5 5 6 7 8` respectively. Note that reversing the list `l3` will not cause a side-effect on the lists `l3` or `l1`.

Fill in the missing code for the class **MyNode** and test the methods of **MyNode** using another class **TestMyNode** with a main method. For example, the above code fragments could be found in the main method of **TestMyNode**.
class TestMyNode {
    public static void main(String[] args) {
        MyNode l1 = new MyNode(1, new MyNode(2, new MyNode(3)));

        System.out.println(l1);
        System.out.println(l1.toString());

        l1.setData(5);
        System.out.println(l1);
        MyNode l2 = new MyNode(5, new MyNode(6));
        l1.setNext(l2);
        System.out.println(l1);
        System.out.println(l1.getNext());

        MyNode l3 = new MyNode(7, new MyNode(8));
        l1.add(l3);
        System.out.println(l1);

        System.out.println(l1.size());
        System.out.println(l3.size());

        MyNode l4 = l3.reverse();
        System.out.println(l3);
        System.out.println(l4);
        System.out.println(l1);
    }
}

os-prompt> java TestMyNode
1 2 3
1 2 3
5 2 3
5 5 6
5 6
5 5 6 7 8
5
2
7 8
8 7
5 5 6 7 8
os-prompt>

Follow these instructions carefully and you will score some easy points:

1. Make sure the text output of your program matches that of the sample run above exactly.
2. Submit your program to the assignment labeled “Program 5: My Node” through your account on WebCT.

3. Submit exactly the following files:
   - MyNode.java - your source code, which should contain a class called MyNode.
   - MyNode.class - your compiled program.

4. If you are working with a partner, only one of you should submit the assignment. But your source code should list both of you as authors.