Gebze Technical University
Computer Engineering

CSE 222 - 2018 Spring

HOMEWORK 2 REPORT

STUDENT NAME
STUDENT NUMBER

SILA BENGİSU YÜKSEKLİ
1801042877

Course Assistant:

1 INTRODUCTION

1.1 Problem Definition

Creating a list -which is called ExperimentList- to keep track of some machine learning experiments and their results by using Single Linked List. Our list doesn't extends any class from Java Collections and it is iterable. It has eight methods:

1.addExp: It inserts experiments to the end of the day according to where they belong.

2.getExp: It returns you an experiment according to its day and index information.

3.setExp: It sets an experiment which has the day and index information that you entered.

4.listExp: It makes a list and show you which experiments completed.

5.removeExp: It deletes an experiment which belongs a given day and index.

6.removeDay: It deletes all experiments according to day information that you gave.

7.orderExperiments: It sorts all the experiments in the list according to the accuracy with a new ExperimentList.

8.orderDay: It sorts all experiments in a given day according to their accuracy. It doesn't change the original list.

1.2 System Requirements

General Requirements:

-File's name and class name must be same.
-Class name's first letter must be upper case.

Special Requirements for my project:

1. addExp: must take one parameter which is a Experiment object.
2.getExp: must take two parameters which are the integer.
3.setExp: must take three parameters which are the integer and Experiment object.
4.listExp: must take one parameter which is integer.
5.removeExp: must take two parameter which are integer.
6.removeDay: must take one parameter which is a integer.

7.orderExperiments: must take no parameter.
8.orderDay: must take one parameter which is a integer.

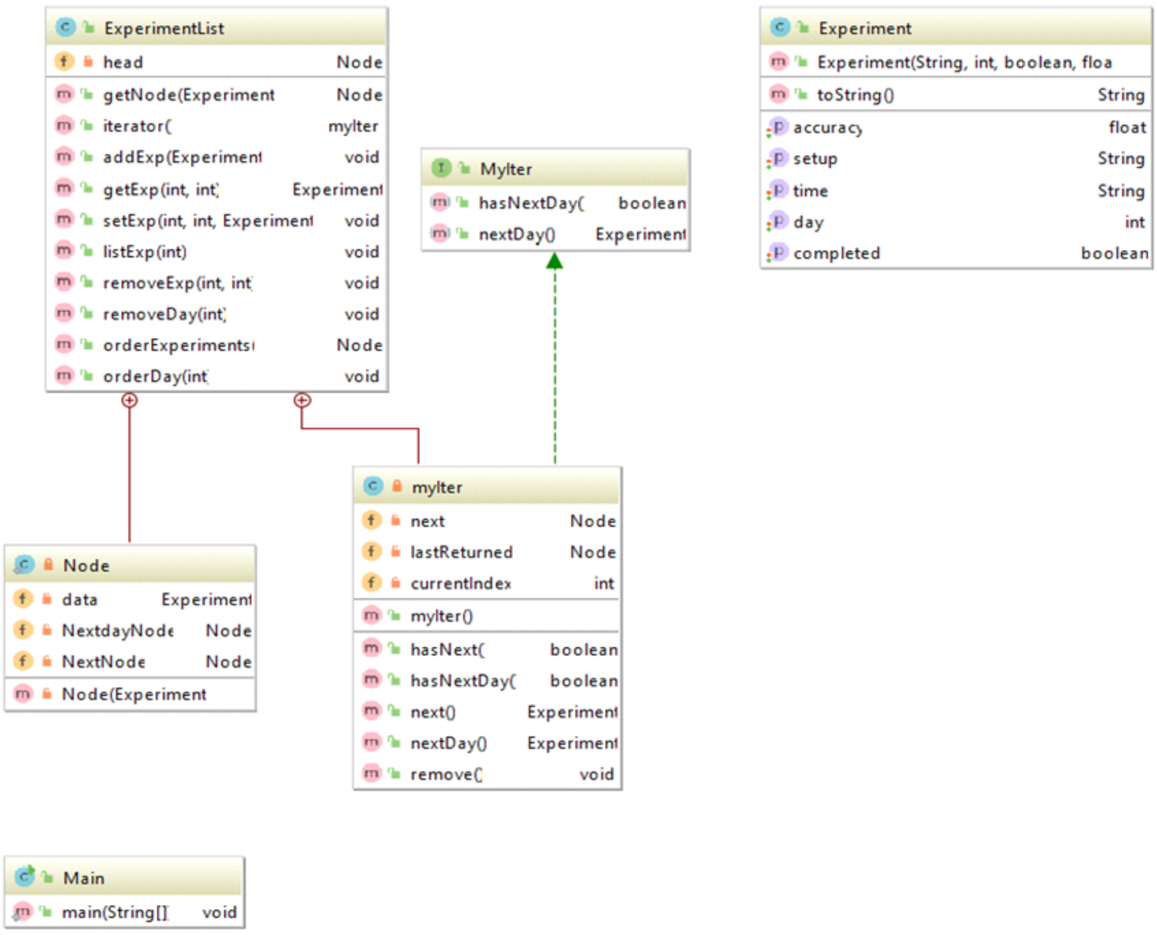
-List must have two inner classes:

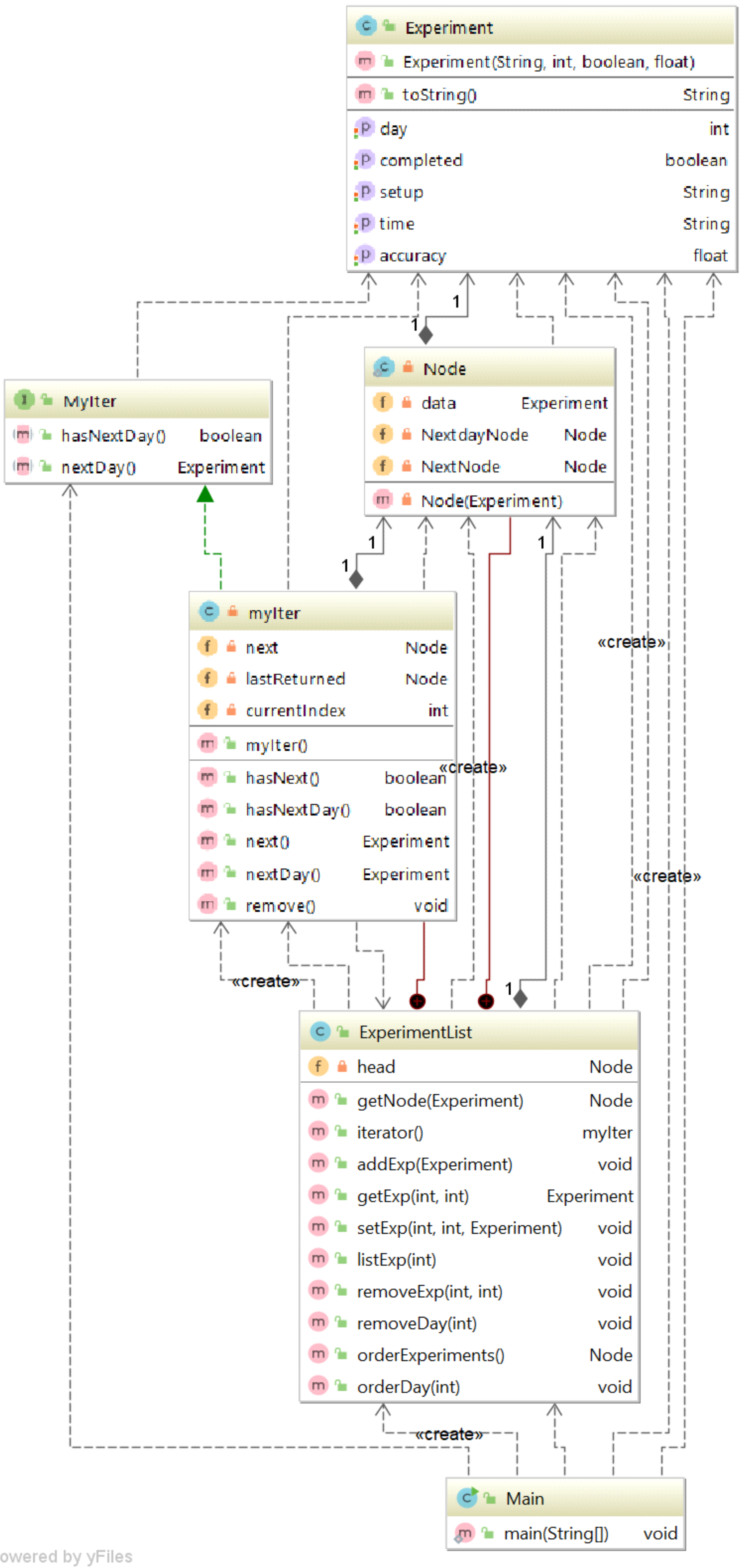
1.Node class : it is necessary to create a Linked List structure.
2.myİter class : it is necessary to use my own iterator.

-Moreover an additional list structure should be defined in the level of days.

2 METHOD

2.1 Class Diagrams





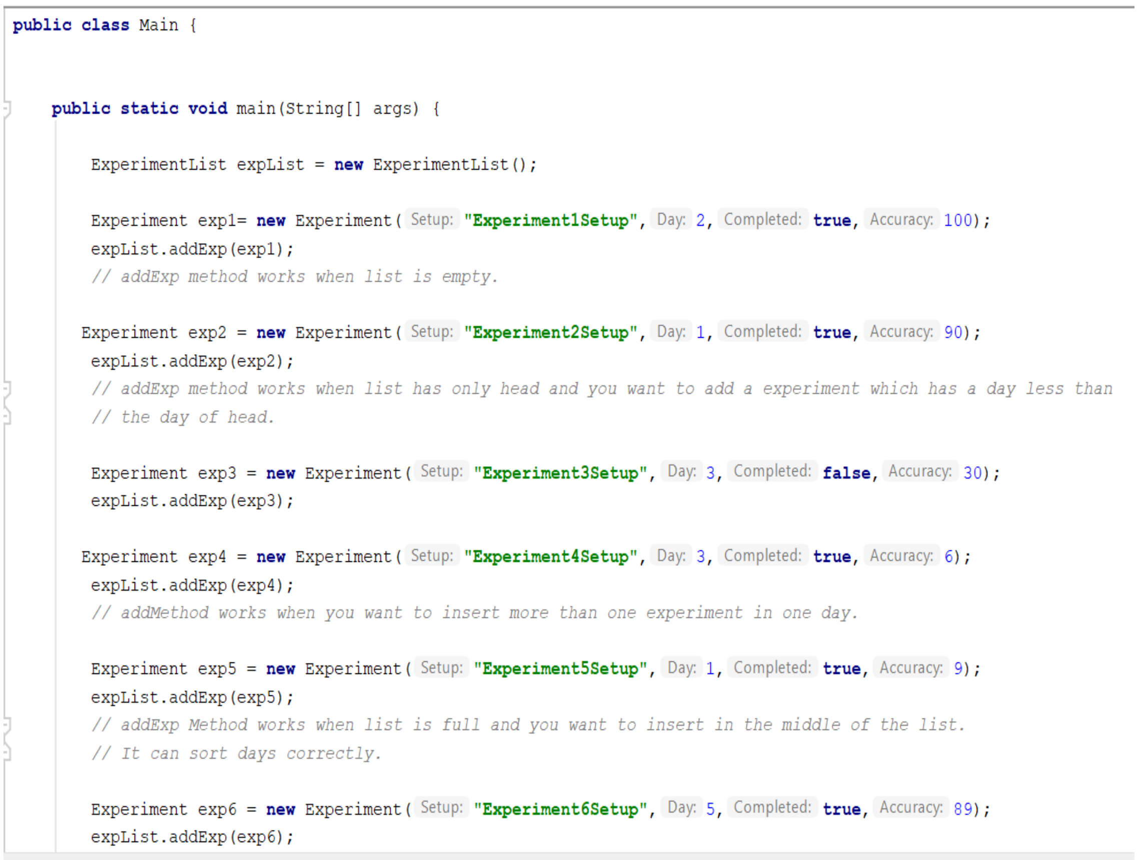
Powered by yFiles

2.2 Problem Solution Approach

I decide to implement my Experiment class first because the ExperimentList methods will be used Experiment objects. Then I went on with implementation of Experiment List. Because of ExperimentList has a Single Linked List structure and is iterable , I started to implementation with the inner classes and then I did the implementation of methods which has given. Finally I wrote a driver class which tests all the functionality of my ExperimentList class.

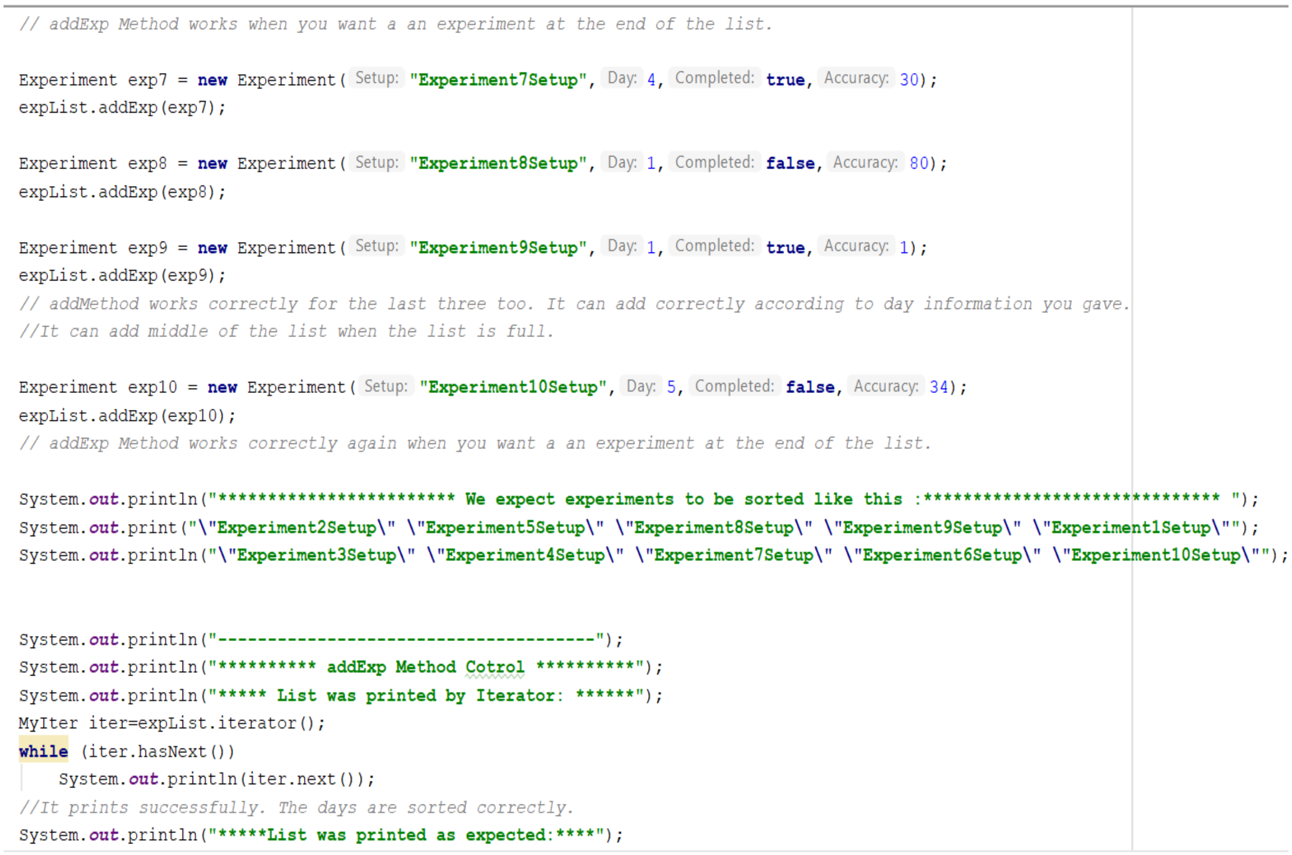
3 RESULT

3.1 Test Cases



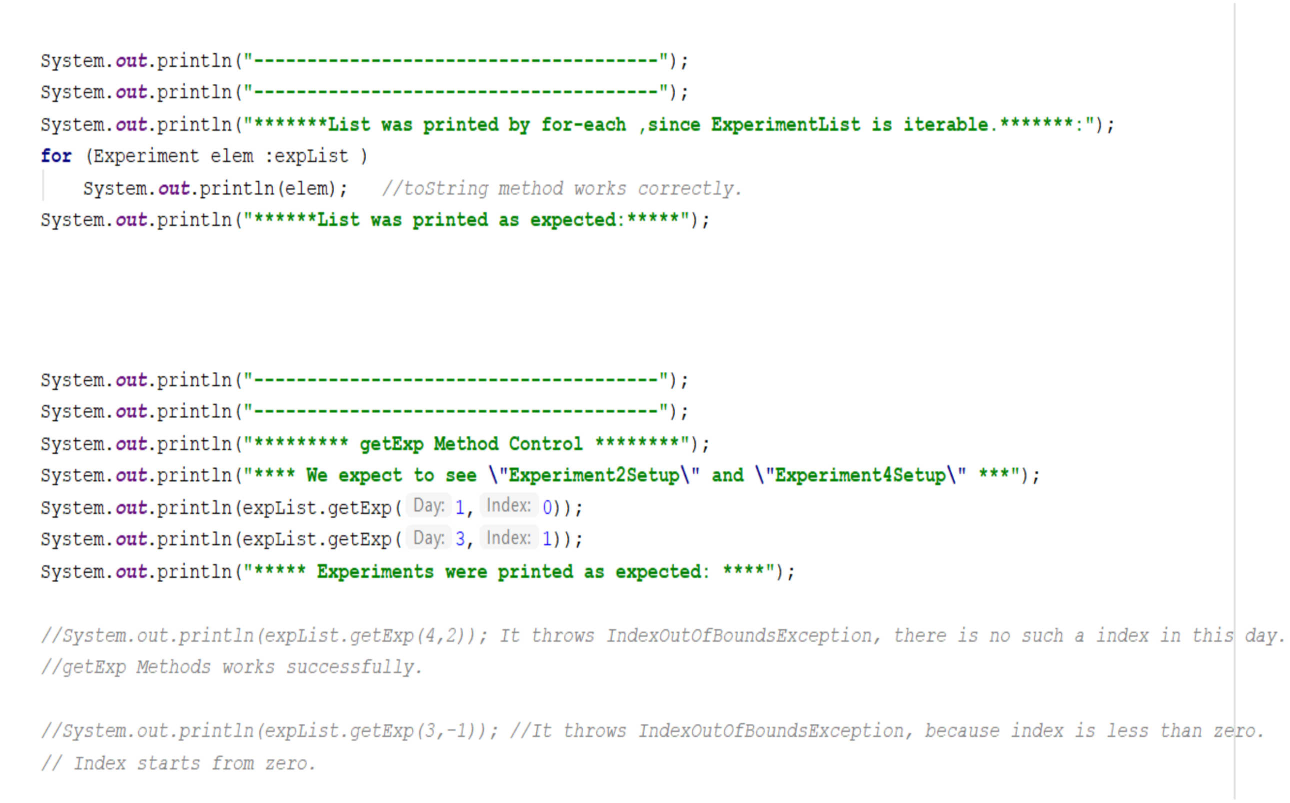
```
public class Main {


    public static void main(String[] args) {

        ExperimentList expList = new ExperimentList();

        Experiment exp1= new Experiment( Setup: "Experiment1Setup", Day: 2, Completed: true, Accuracy: 100);
        expList.addExp(exp1);
        // addExp method works when list is empty.

       Experiment exp2 = new Experiment( Setup: "Experiment2Setup", Day: 1, Completed: true, Accuracy: 90);
        expList.addExp(exp2);
        // addExp method works when list has only head and you want to add a experiment which has a day less than
        // the day of head.

        Experiment exp3 = new Experiment( Setup: "Experiment3Setup", Day: 3, Completed: false, Accuracy: 30);
        expList.addExp(exp3);

       Experiment exp4 = new Experiment( Setup: "Experiment4Setup", Day: 3, Completed: true, Accuracy: 6);
        expList.addExp(exp4);
        // addMethod works when you want to insert more than one experiment in one day.

        Experiment exp5 = new Experiment( Setup: "Experiment5Setup", Day: 1, Completed: true, Accuracy: 9);
        expList.addExp(exp5);
        // addExp Method works when list is full and you want to insert in the middle of the list.
        // It can sort days correctly.

        Experiment exp6 = new Experiment( Setup: "Experiment6Setup", Day: 5, Completed: true, Accuracy: 89);
        expList.addExp(exp6);
```

-addExp Method test



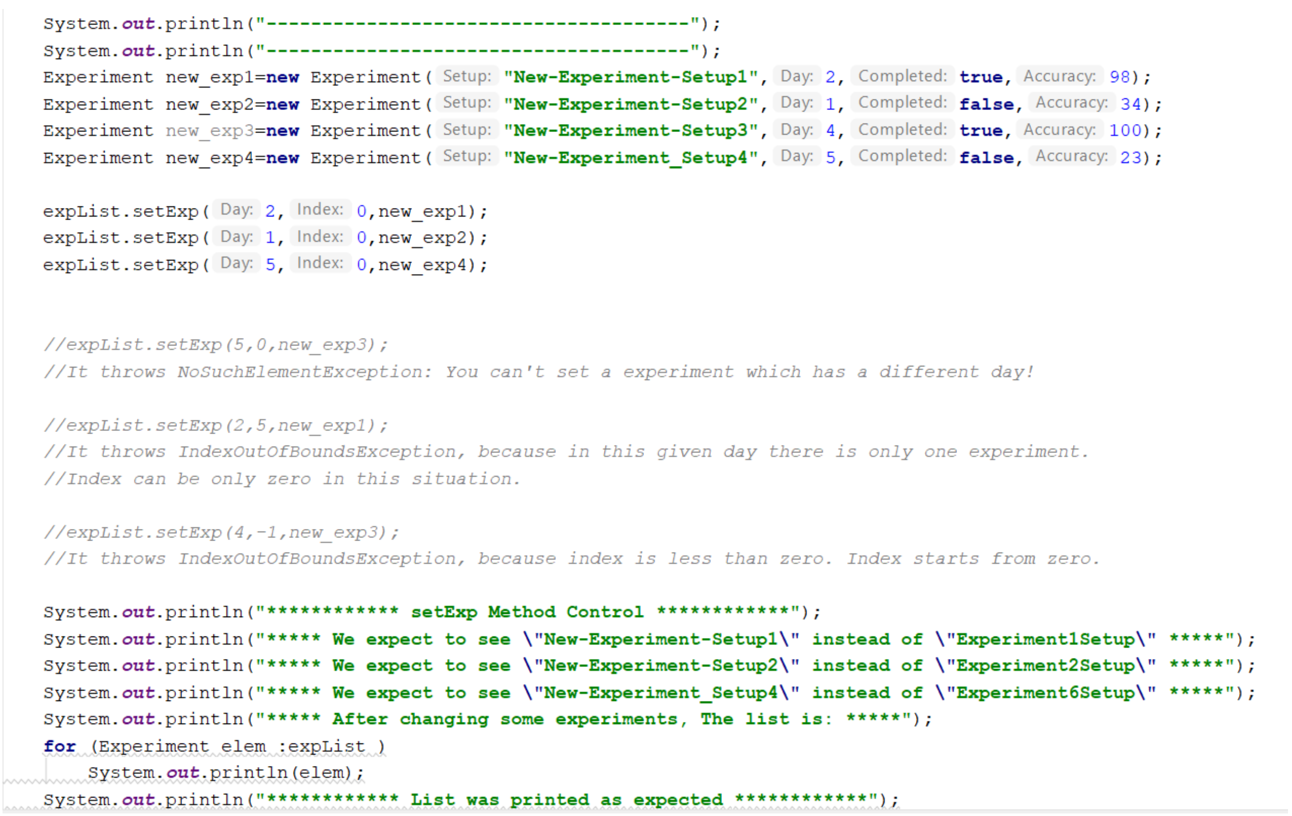
```
// addExp Method works when you want a an experiment at the end of the list.

Experiment exp7 = new Experiment( Setup: "Experiment7Setup", Day: 4, Completed: true, Accuracy: 30);
expList.addExp(exp7);

Experiment exp8 = new Experiment( Setup: "Experiment8Setup", Day: 1, Completed: false, Accuracy: 80);
expList.addExp(exp8);

Experiment exp9 = new Experiment( Setup: "Experiment9Setup", Day: 1, Completed: true, Accuracy: 1);
expList.addExp(exp9);
// addMethod works correctly for the last three too. It can add correctly according to day information you gave.
//It can add middle of the list when the list is full.

Experiment exp10 = new Experiment( Setup: "Experiment10Setup", Day: 5, Completed: false, Accuracy: 34);
expList.addExp(exp10);
// addExp Method works correctly again when you want a an experiment at the end of the list.

System.out.println("*********************** We expect experiments to be sorted like this :***************************** ");
System.out.print("\"Experiment2Setup\" \"Experiment5Setup\" \"Experiment8Setup\" \"Experiment9Setup\" \"Experiment1Setup\"");
System.out.println("\"Experiment3Setup\" \"Experiment4Setup\" \"Experiment7Setup\" \"Experiment6Setup\" \"Experiment10Setup\"");


System.out.println("--------------------------------------");
System.out.println("********** addExp Method Cotrol **********");
System.out.println("***** List was printed by Iterator: ******");
MyIter iter=expList.iterator();
while (iter.hasNext())
    System.out.println(iter.next());
//It prints successfully. The days are sorted correctly.
System.out.println("*****List was printed as expected:****");
```

-addExp Method Test



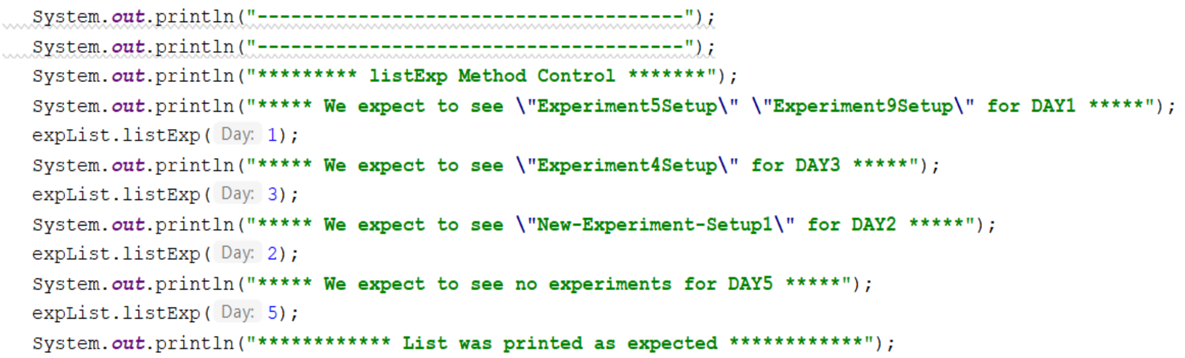
```
System.out.println("----------------------------------------");
System.out.println("----------------------------------------");
System.out.println("*******List was printed by for-each ,since ExperimentList is iterable.*******:");
for (Experiment elem :expList )
    System.out.println(elem);   //toString method works correctly.
System.out.println("******List was printed as expected:*****");




System.out.println("----------------------------------------");
System.out.println("----------------------------------------");
System.out.println("********* getExp Method Control ********");
System.out.println("**** We expect to see \"Experiment2Setup\" and \"Experiment4Setup\" ***");
System.out.println(expList.getExp( Day: 1, Index: 0));
System.out.println(expList.getExp( Day: 3, Index: 1));
System.out.println("***** Experiments were printed as expected: ****");

//System.out.println(expList.getExp(4,2)); It throws IndexOutOfBoundsException, there is no such a index in this day.
//getExp Methods works successfully.

//System.out.println(expList.getExp(3,-1)); //It throws IndexOutOfBoundsException, because index is less than zero.
// Index starts from zero.
```

-getExp Method Test

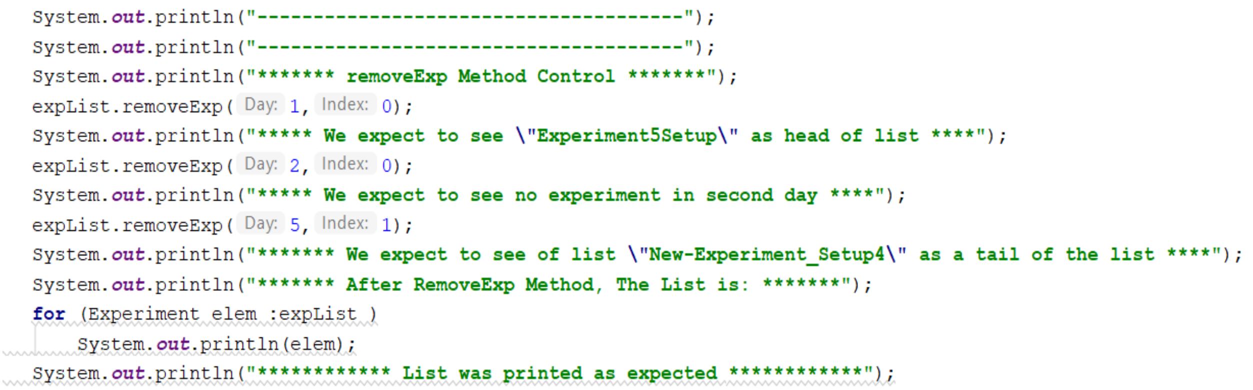


```
System.out.println("----------------------------------------");
System.out.println("----------------------------------------");
Experiment new_exp1=new Experiment( Setup: "New-Experiment-Setup1", Day: 2, Completed: true, Accuracy: 98);
Experiment new_exp2=new Experiment( Setup: "New-Experiment-Setup2", Day: 1, Completed: false, Accuracy: 34);
Experiment new_exp3=new Experiment( Setup: "New-Experiment-Setup3", Day: 4, Completed: true, Accuracy: 100);
Experiment new_exp4=new Experiment( Setup: "New-Experiment_Setup4", Day: 5, Completed: false, Accuracy: 23);

expList.setExp( Day: 2, Index: 0,new_exp1);
expList.setExp( Day: 1, Index: 0,new_exp2);
expList.setExp( Day: 5, Index: 0,new_exp4);


//expList.setExp(5,0,new_exp3);
//It throws NoSuchElementException: You can't set a experiment which has a different day!

//expList.setExp(2,5,new_exp1);
//It throws IndexOutOfBoundsException, because in this given day there is only one experiment.
//Index can be only zero in this situation.

//expList.setExp(4,-1,new_exp3);
//It throws IndexOutOfBoundsException, because index is less than zero. Index starts from zero.

System.out.println("************ setExp Method Control ************");
System.out.println("***** We expect to see \"New-Experiment-Setup1\" instead of \"Experiment1Setup\" *****");
System.out.println("***** We expect to see \"New-Experiment-Setup2\" instead of \"Experiment2Setup\" *****");
System.out.println("***** We expect to see \"New-Experiment_Setup4\" instead of \"Experiment6Setup\" *****");
System.out.println("***** After changing some experiments, The list is: *****");
for (Experiment elem :expList )
    System.out.println(elem);
System.out.println("************ List was printed as expected ************");
```

-setExp Method Test

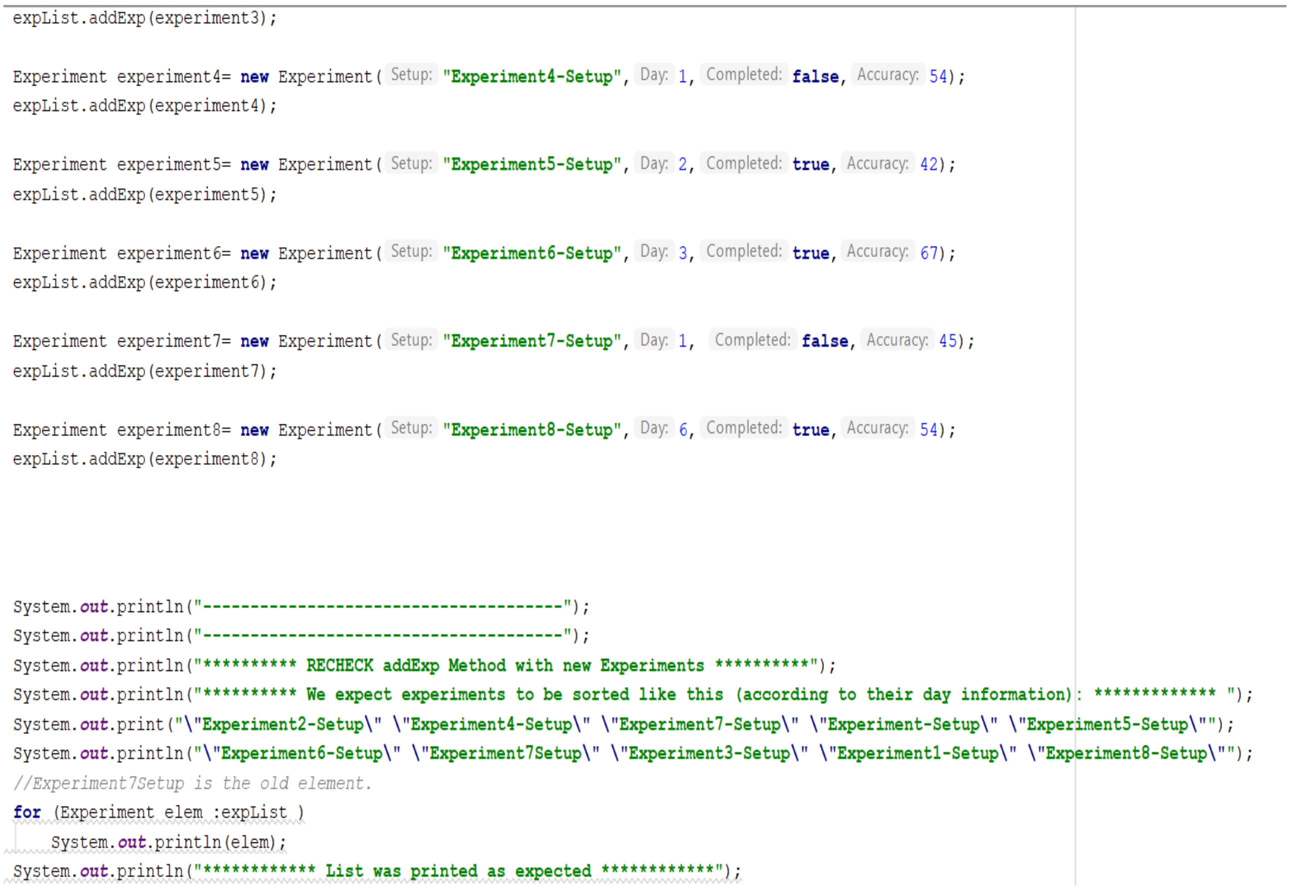


```
System.out.println("----------------------------------------");
System.out.println("----------------------------------------");
System.out.println("********* listExp Method Control *******");
System.out.println("***** We expect to see \"Experiment5Setup\" \"Experiment9Setup\" for DAY1 *****");
expList.listExp( Day: 1);
System.out.println("***** We expect to see \"Experiment4Setup\" for DAY3 *****");
expList.listExp( Day: 3);
System.out.println("***** We expect to see \"New-Experiment-Setup1\" for DAY2 *****");
expList.listExp( Day: 2);
System.out.println("***** We expect to see no experiments for DAY5 *****");
expList.listExp( Day: 5);
System.out.println("************ List was printed as expected ************");
```



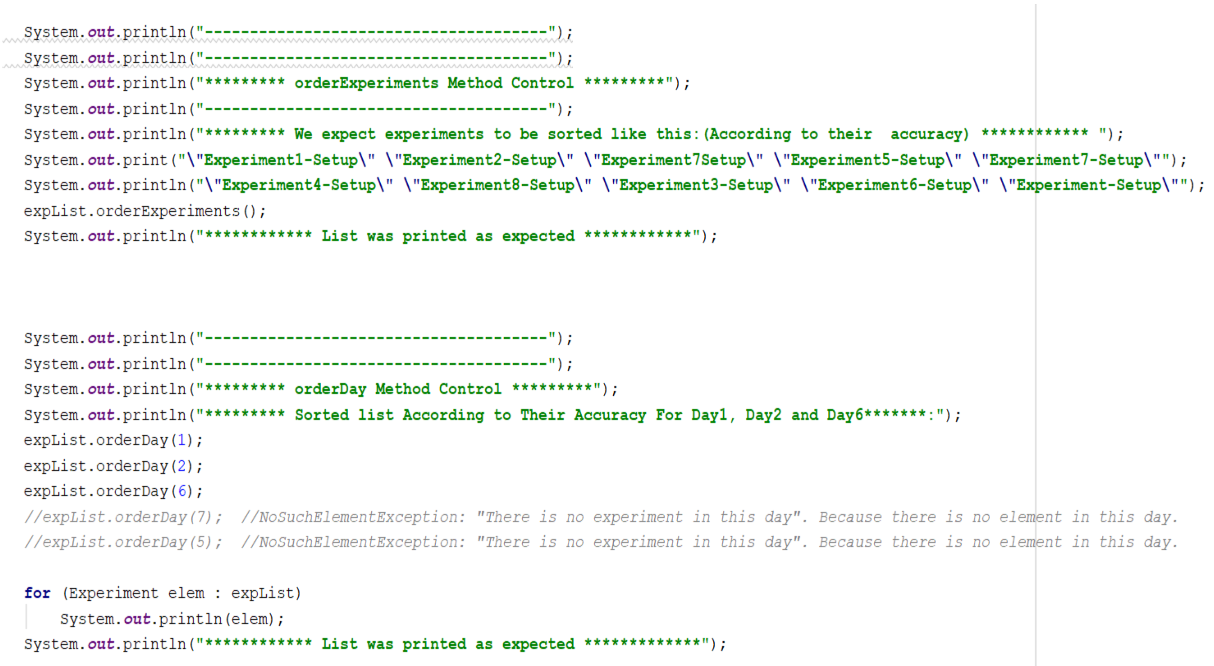
```
System.out.println("----------------------------------------");
System.out.println("----------------------------------------");
System.out.println("******* removeExp Method Control *******");
expList.removeExp( Day: 1, Index: 0);
System.out.println("***** We expect to see \"Experiment5Setup\" as head of list ****");
expList.removeExp( Day: 2, Index: 0);
System.out.println("***** We expect to see no experiment in second day ****");
expList.removeExp( Day: 5, Index: 1);
System.out.println("******* We expect to see of list \"New-Experiment_Setup4\" as a tail of the list ****");
System.out.println("******* After RemoveExp Method, The List is: *******");
for (Experiment elem :expList )
    System.out.println(elem);
System.out.println("************ List was printed as expected ************");
```

-listExp Method Test



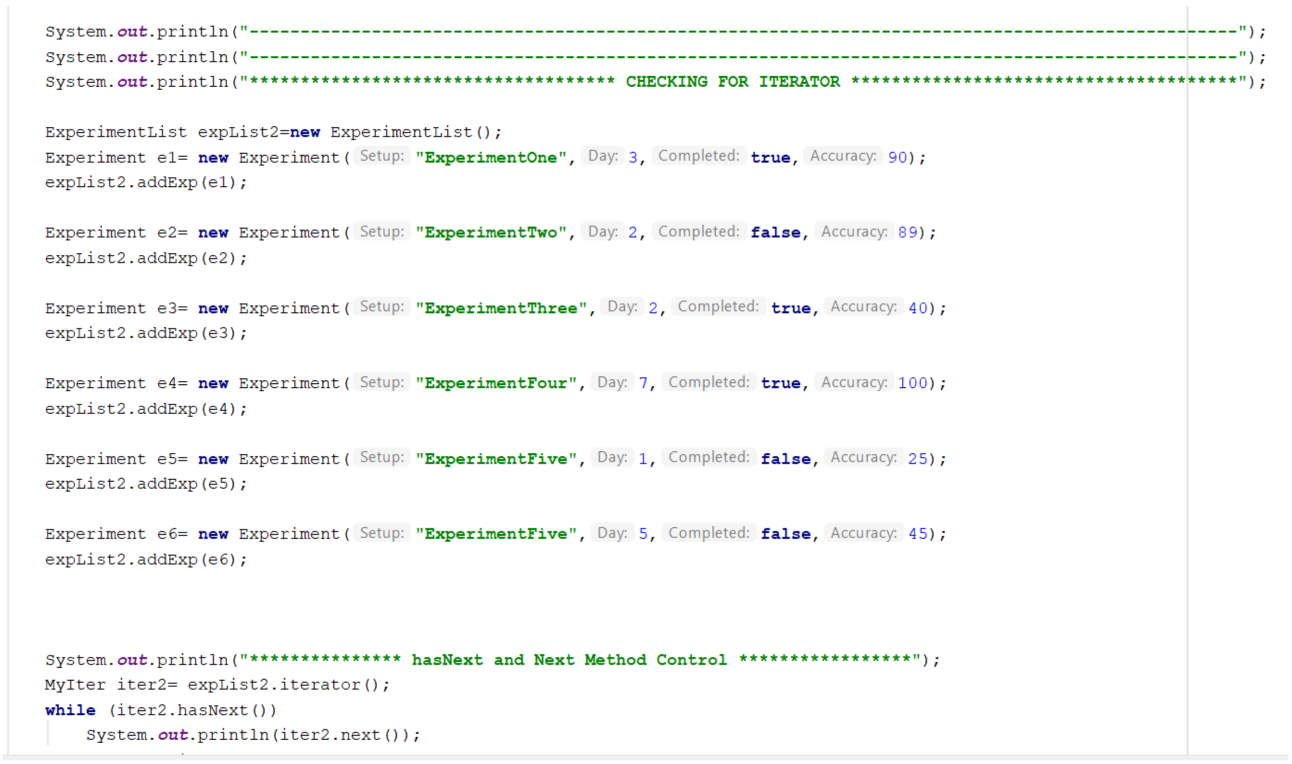
```java
expList.addExp(experiment3);

Experiment experiment4= new Experiment( Setup: "Experiment4-Setup", Day: 1, Completed: false, Accuracy: 54);
expList.addExp(experiment4);

Experiment experiment5= new Experiment( Setup: "Experiment5-Setup", Day: 2, Completed: true, Accuracy: 42);
expList.addExp(experiment5);

Experiment experiment6= new Experiment( Setup: "Experiment6-Setup", Day: 3, Completed: true, Accuracy: 67);
expList.addExp(experiment6);

Experiment experiment7= new Experiment( Setup: "Experiment7-Setup", Day: 1,  Completed: false, Accuracy: 45);
expList.addExp(experiment7);

Experiment experiment8= new Experiment( Setup: "Experiment8-Setup", Day: 6, Completed: true, Accuracy: 54);
expList.addExp(experiment8);




System.out.println("--------------------------------------");
System.out.println("--------------------------------------");
System.out.println("********** RECHECK addExp Method with new Experiments **********");
System.out.println("********** We expect experiments to be sorted like this (according to their day information): ************* ");
System.out.print("\"Experiment2-Setup\" \"Experiment4-Setup\" \"Experiment7-Setup\" \"Experiment-Setup\" \"Experiment5-Setup\"");
System.out.println("\"Experiment6-Setup\" \"Experiment7Setup\" \"Experiment3-Setup\" \"Experiment1-Setup\" \"Experiment8-Setup\"");
//Experiment7Setup is the old element.
for (Experiment elem :expList )
    System.out.println(elem);
System.out.println("************ List was printed as expected ************");
```

-addExp Method Test

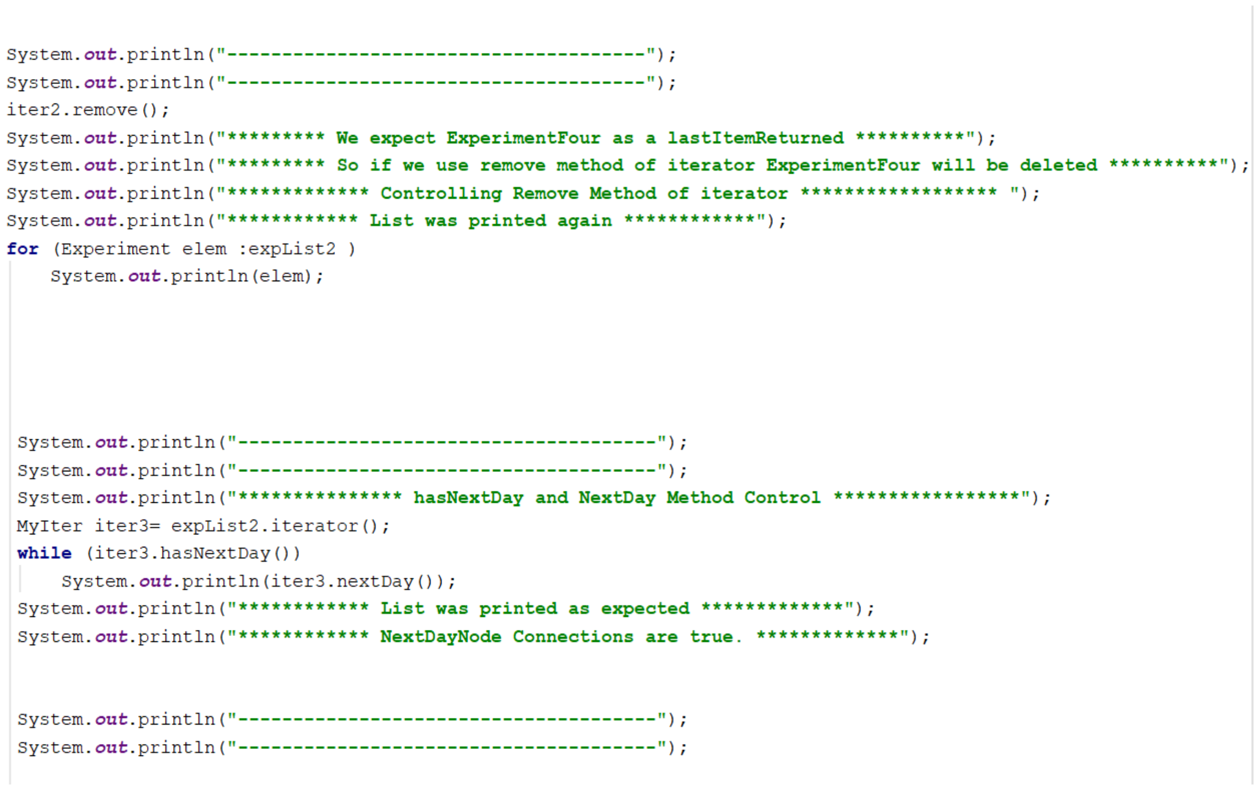


```java
System.out.println("--------------------------------------");
System.out.println("--------------------------------------");
System.out.println("********* orderExperiments Method Control *********");
System.out.println("--------------------------------------");
System.out.println("********* We expect experiments to be sorted like this:(According to their  accuracy) ************ ");
System.out.print("\"Experiment1-Setup\" \"Experiment2-Setup\" \"Experiment7Setup\" \"Experiment5-Setup\" \"Experiment7-Setup\"");
System.out.println("\"Experiment4-Setup\" \"Experiment8-Setup\" \"Experiment3-Setup\" \"Experiment6-Setup\" \"Experiment-Setup\"");
expList.orderExperiments();
System.out.println("************ List was printed as expected ************");



System.out.println("--------------------------------------");
System.out.println("--------------------------------------");
System.out.println("********* orderDay Method Control *********");
System.out.println("********* Sorted list According to Their Accuracy For Day1, Day2 and Day6*******:");
expList.orderDay(1);
expList.orderDay(2);
expList.orderDay(6);
//expList.orderDay(7);  //NoSuchElementException: "There is no experiment in this day". Because there is no element in this day.
//expList.orderDay(5);  //NoSuchElementException: "There is no experiment in this day". Because there is no element in this day.

for (Experiment elem : expList)
    System.out.println(elem);
System.out.println("************ List was printed as expected *************");
```

-orderExperiments and orderDay Methods Test



```
System.out.println("-------------------------------------------------------------------------------------------------");
System.out.println("-------------------------------------------------------------------------------------------------");
System.out.println("************************************ CHECKING FOR ITERATOR ***************************************");

ExperimentList expList2=new ExperimentList();
Experiment e1= new Experiment( Setup: "ExperimentOne", Day: 3, Completed: true, Accuracy: 90);
expList2.addExp(e1);

Experiment e2= new Experiment( Setup: "ExperimentTwo", Day: 2, Completed: false, Accuracy: 89);
expList2.addExp(e2);

Experiment e3= new Experiment( Setup: "ExperimentThree", Day: 2, Completed: true, Accuracy: 40);
expList2.addExp(e3);

Experiment e4= new Experiment( Setup: "ExperimentFour", Day: 7, Completed: true, Accuracy: 100);
expList2.addExp(e4);

Experiment e5= new Experiment( Setup: "ExperimentFive", Day: 1, Completed: false, Accuracy: 25);
expList2.addExp(e5);

Experiment e6= new Experiment( Setup: "ExperimentFive", Day: 5, Completed: false, Accuracy: 45);
expList2.addExp(e6);



System.out.println("*************** hasNext and Next Method Control *****************");
MyIter iter2= expList2.iterator();
while (iter2.hasNext())
    System.out.println(iter2.next());
```

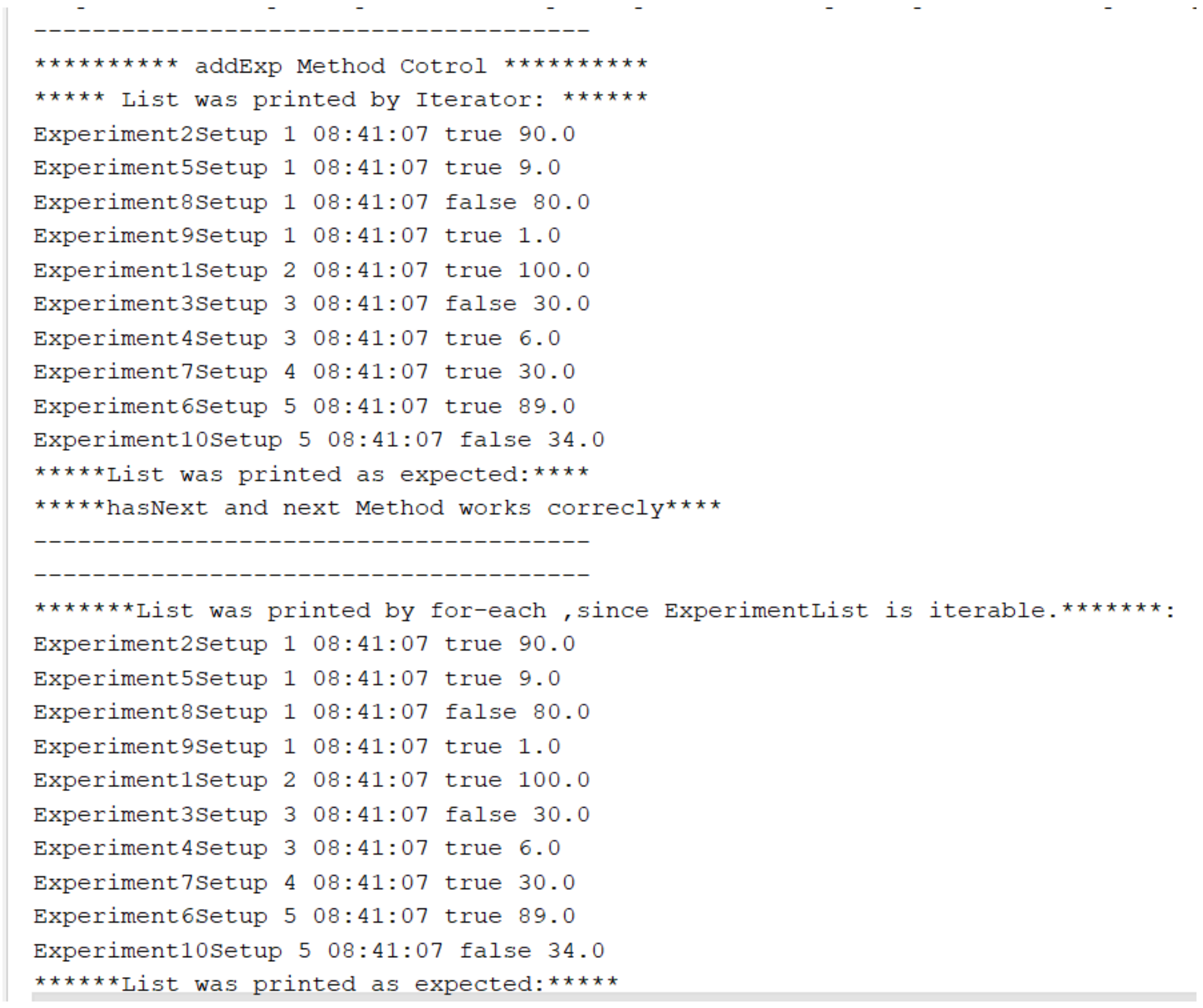


```
System.out.println("---------------------------------------");
System.out.println("---------------------------------------");
iter2.remove();
System.out.println("********* We expect ExperimentFour as a lastItemReturned **********");
System.out.println("********* So if we use remove method of iterator ExperimentFour will be deleted **********");
System.out.println("************* Controlling Remove Method of iterator ****************** ");
System.out.println("************ List was printed again ************");
for (Experiment elem :expList2 )
    System.out.println(elem);




System.out.println("---------------------------------------");
System.out.println("---------------------------------------");
System.out.println("*************** hasNextDay and NextDay Method Control *****************");
MyIter iter3= expList2.iterator();
while (iter3.hasNextDay())
    System.out.println(iter3.nextDay());
System.out.println("************ List was printed as expected *************");
System.out.println("************ NextDayNode Connections are true. *************");


System.out.println("---------------------------------------");
System.out.println("---------------------------------------");
```

-iterator test

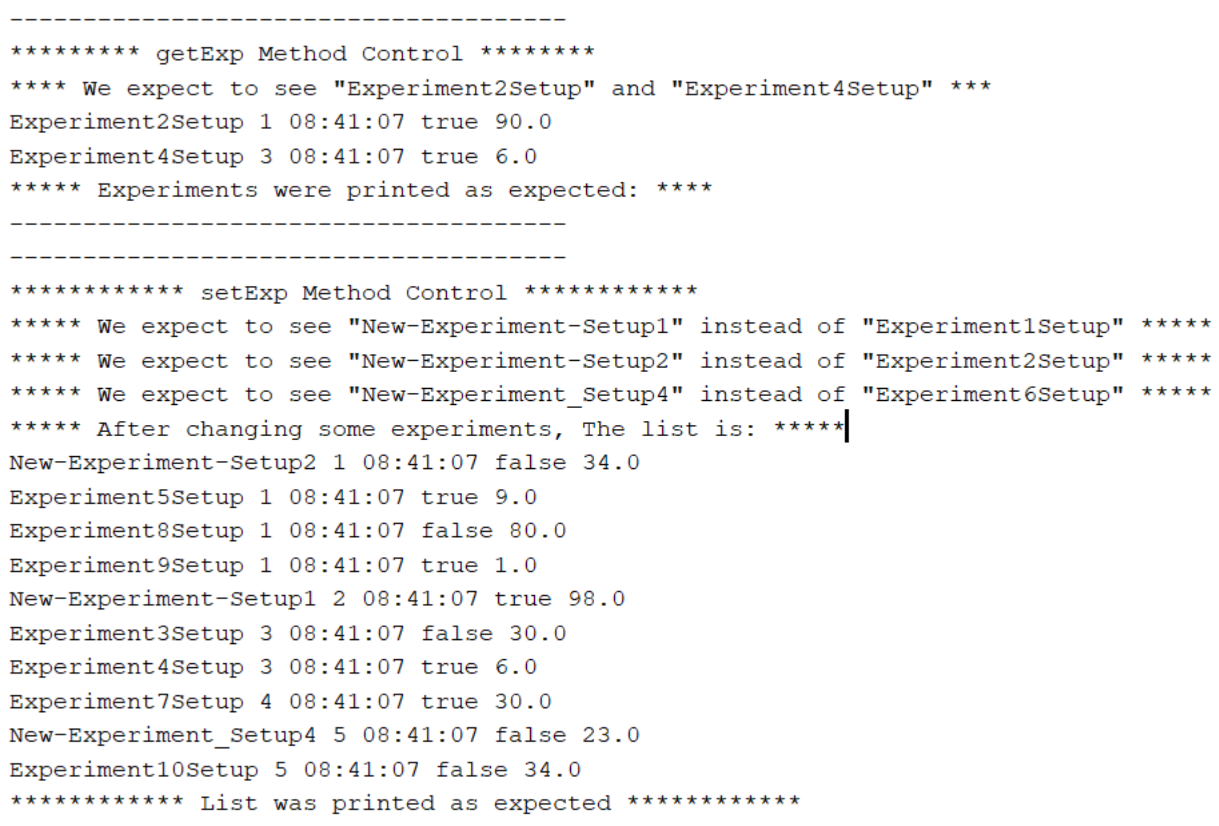
3.2 Running Results

The result of driver function is:



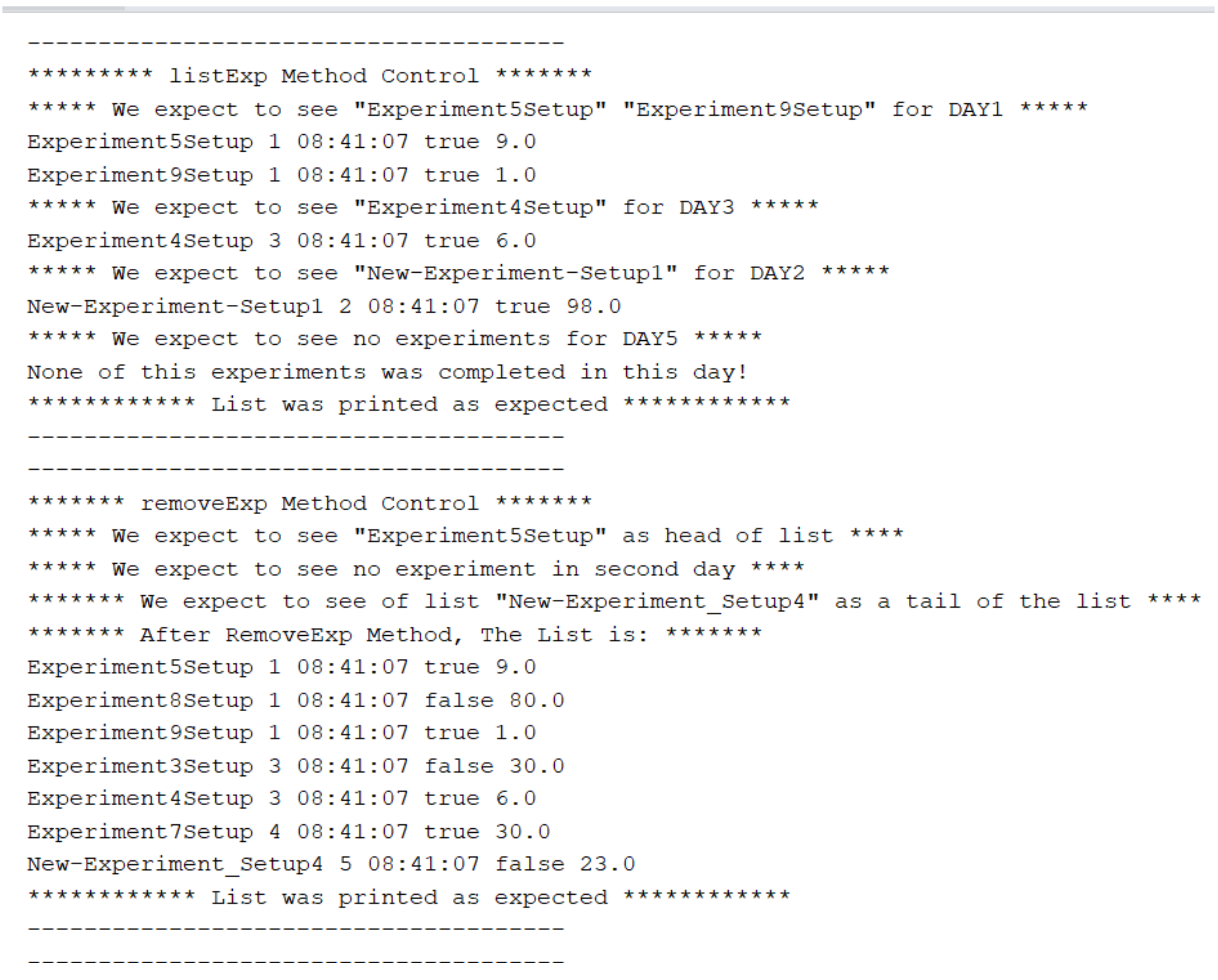
```
--------------------------------------
********** addExp Method Cotrol **********
***** List was printed by Iterator: ******
Experiment2Setup 1 08:41:07 true 90.0
Experiment5Setup 1 08:41:07 true 9.0
Experiment8Setup 1 08:41:07 false 80.0
Experiment9Setup 1 08:41:07 true 1.0
Experiment1Setup 2 08:41:07 true 100.0
Experiment3Setup 3 08:41:07 false 30.0
Experiment4Setup 3 08:41:07 true 6.0
Experiment7Setup 4 08:41:07 true 30.0
Experiment6Setup 5 08:41:07 true 89.0
Experiment10Setup 5 08:41:07 false 34.0
*****List was printed as expected:****
*****hasNext and next Method works correcly****
--------------------------------------
--------------------------------------
*******List was printed by for-each ,since ExperimentList is iterable.*******:
Experiment2Setup 1 08:41:07 true 90.0
Experiment5Setup 1 08:41:07 true 9.0
Experiment8Setup 1 08:41:07 false 80.0
Experiment9Setup 1 08:41:07 true 1.0
Experiment1Setup 2 08:41:07 true 100.0
Experiment3Setup 3 08:41:07 false 30.0
Experiment4Setup 3 08:41:07 true 6.0
Experiment7Setup 4 08:41:07 true 30.0
Experiment6Setup 5 08:41:07 true 89.0
Experiment10Setup 5 08:41:07 false 34.0
******List was printed as expected:*****
```

It is for the addExp Method. It prints correctly. Days are ordered as expected.



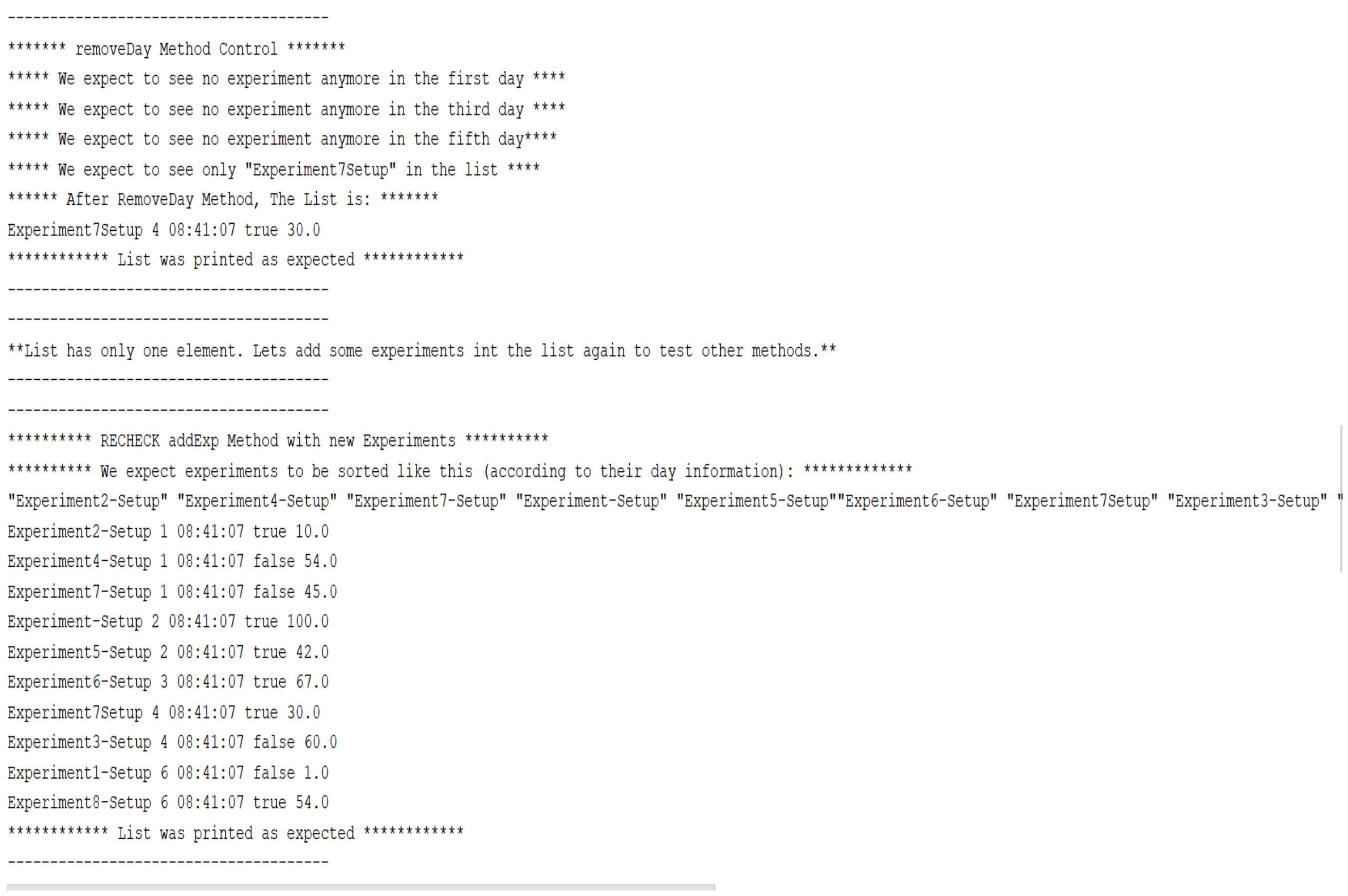
```
---------------------------------------
********* getExp Method Control ********
**** We expect to see "Experiment2Setup" and "Experiment4Setup" ***
Experiment2Setup 1 08:41:07 true 90.0
Experiment4Setup 3 08:41:07 true 6.0
***** Experiments were printed as expected: ****
---------------------------------------
---------------------------------------
************ setExp Method Control ************
***** We expect to see "New-Experiment-Setup1" instead of "Experiment1Setup" *****
***** We expect to see "New-Experiment-Setup2" instead of "Experiment2Setup" *****
***** We expect to see "New-Experiment_Setup4" instead of "Experiment6Setup" *****
***** After changing some experiments, The list is: *****
New-Experiment-Setup2 1 08:41:07 false 34.0
Experiment5Setup 1 08:41:07 true 9.0
Experiment8Setup 1 08:41:07 false 80.0
Experiment9Setup 1 08:41:07 true 1.0
New-Experiment-Setup1 2 08:41:07 true 98.0
Experiment3Setup 3 08:41:07 false 30.0
Experiment4Setup 3 08:41:07 true 6.0
Experiment7Setup 4 08:41:07 true 30.0
New-Experiment_Setup4 5 08:41:07 false 23.0
Experiment10Setup 5 08:41:07 false 34.0
************ List was printed as expected ************
```

getExp and setExp Methods are works coreectly. The results are as expected.



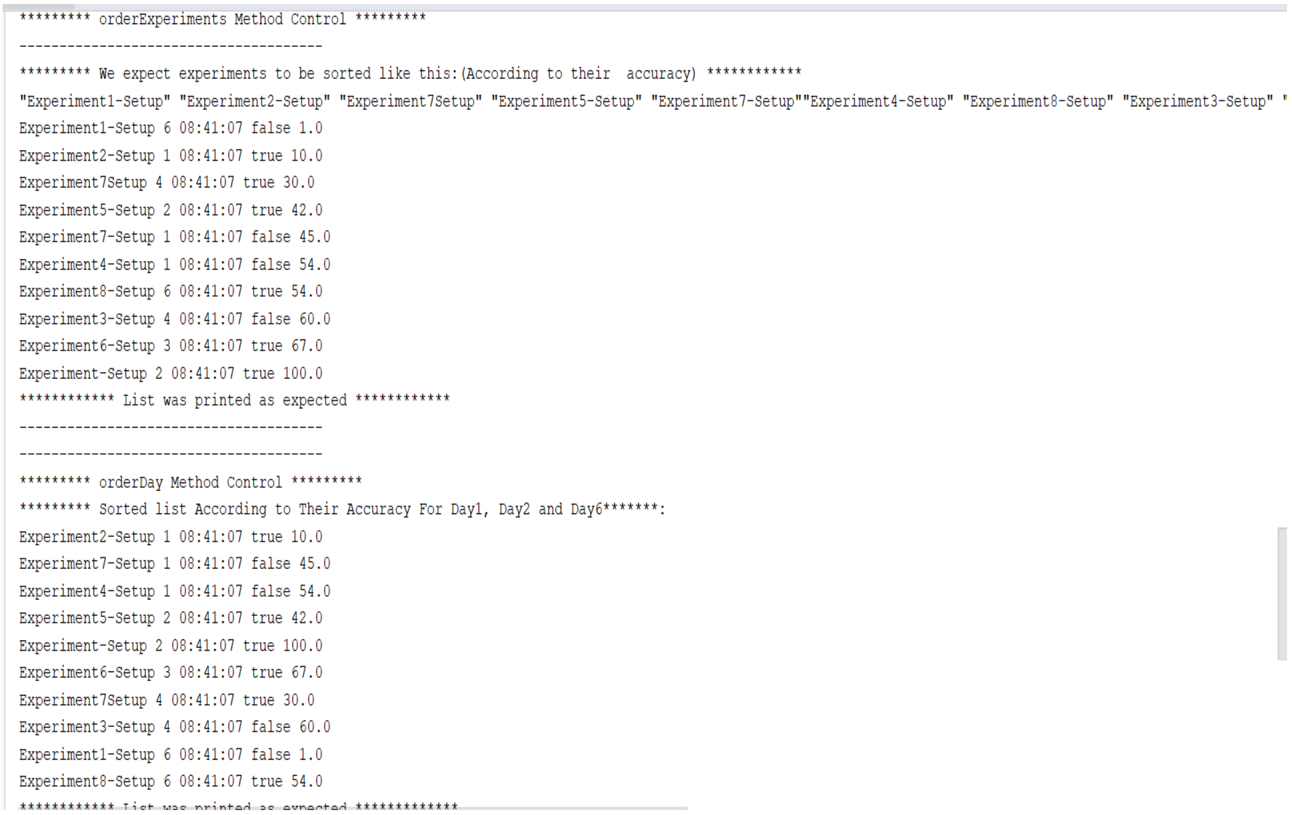
```
---------------------------------------
********* listExp Method Control *******
***** We expect to see "Experiment5Setup" "Experiment9Setup" for DAY1 *****
Experiment5Setup 1 08:41:07 true 9.0
Experiment9Setup 1 08:41:07 true 1.0
***** We expect to see "Experiment4Setup" for DAY3 *****
Experiment4Setup 3 08:41:07 true 6.0
***** We expect to see "New-Experiment-Setup1" for DAY2 *****
New-Experiment-Setup1 2 08:41:07 true 98.0
***** We expect to see no experiments for DAY5 *****
None of this experiments was completed in this day!
************ List was printed as expected ************
---------------------------------------
---------------------------------------
******* removeExp Method Control *******
***** We expect to see "Experiment5Setup" as head of list ****
***** We expect to see no experiment in second day ****
******* We expect to see of list "New-Experiment_Setup4" as a tail of the list ****
******* After RemoveExp Method, The List is: *******
Experiment5Setup 1 08:41:07 true 9.0
Experiment8Setup 1 08:41:07 false 80.0
Experiment9Setup 1 08:41:07 true 1.0
Experiment3Setup 3 08:41:07 false 30.0
Experiment4Setup 3 08:41:07 true 6.0
Experiment7Setup 4 08:41:07 true 30.0
New-Experiment_Setup4 5 08:41:07 false 23.0
************ List was printed as expected ************
---------------------------------------
---------------------------------------
```

listExp and removeExp Methods are works coreectly. The results are as expected. All experiments which were printed, are completed.(true). So listExp Works ok and removeExp method delete the right experiments.



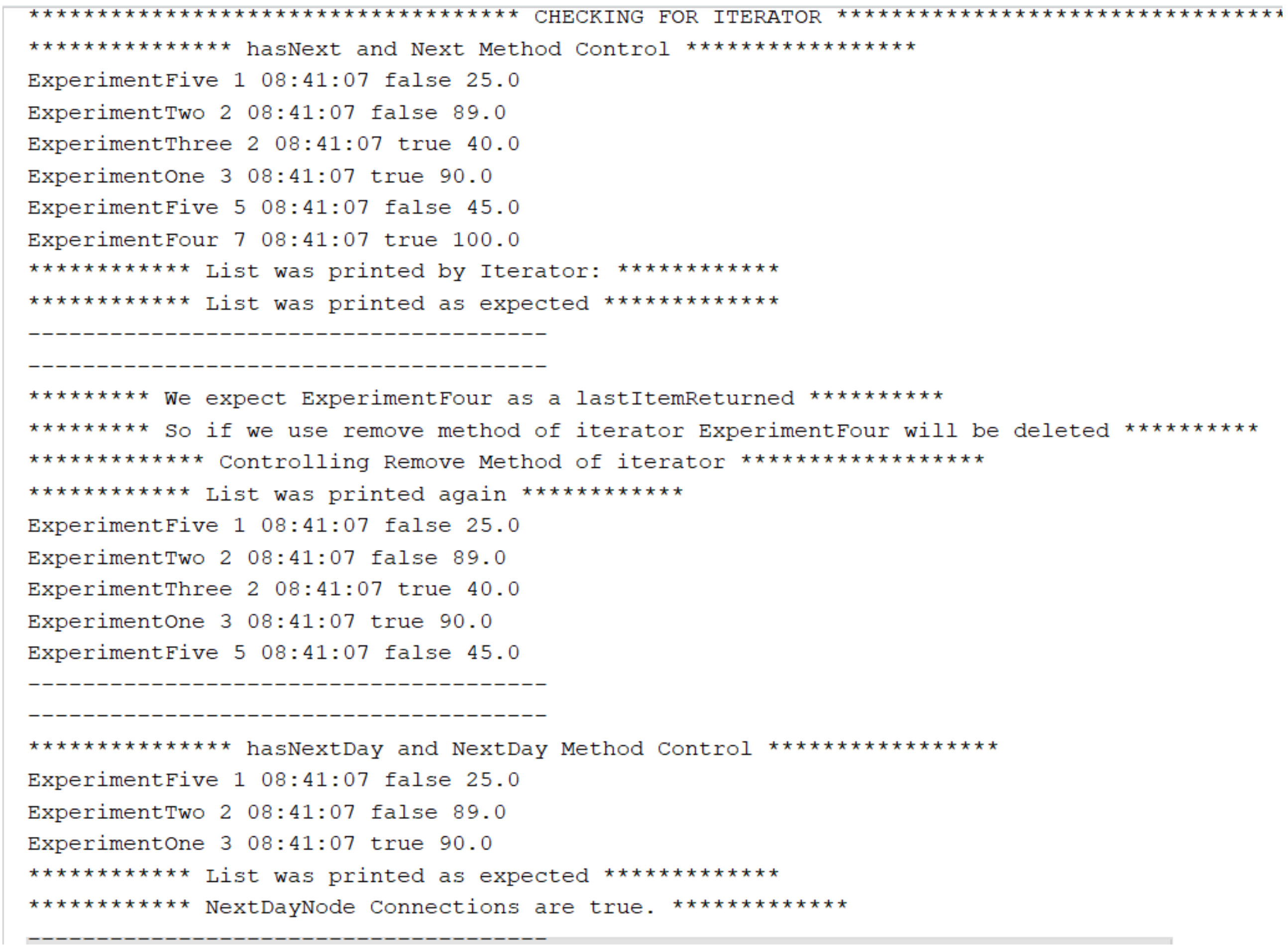
```
--------------------------------------
******* removeDay Method Control *******
***** We expect to see no experiment anymore in the first day ****
***** We expect to see no experiment anymore in the third day ****
***** We expect to see no experiment anymore in the fifth day****
***** We expect to see only "Experiment7Setup" in the list ****
****** After RemoveDay Method, The List is: *******
Experiment7Setup 4 08:41:07 true 30.0
************ List was printed as expected ************
--------------------------------------
--------------------------------------
**List has only one element. Lets add some experiments int the list again to test other methods.**
--------------------------------------
--------------------------------------
********** RECHECK addExp Method with new Experiments **********
********** We expect experiments to be sorted like this (according to their day information): *************
"Experiment2-Setup" "Experiment4-Setup" "Experiment7-Setup" "Experiment-Setup" "Experiment5-Setup""Experiment6-Setup" "Experiment7Setup" "Experiment3-Setup" "
Experiment2-Setup 1 08:41:07 true 10.0
Experiment4-Setup 1 08:41:07 false 54.0
Experiment7-Setup 1 08:41:07 false 45.0
Experiment-Setup 2 08:41:07 true 100.0
Experiment5-Setup 2 08:41:07 true 42.0
Experiment6-Setup 3 08:41:07 true 67.0
Experiment7Setup 4 08:41:07 true 30.0
Experiment3-Setup 4 08:41:07 false 60.0
Experiment1-Setup 6 08:41:07 false 1.0
Experiment8-Setup 6 08:41:07 true 54.0
************ List was printed as expected ************
--------------------------------------
```

-removeDay method and addExp Method again, they are work correctly. After removeExp and removeDay method list was empty. So I use addExp method again to test other methods.



```
********* orderExperiments Method Control *********
-------------------------------------
********* We expect experiments to be sorted like this:(According to their  accuracy) ************
"Experiment1-Setup" "Experiment2-Setup" "Experiment7Setup" "Experiment5-Setup" "Experiment7-Setup""Experiment4-Setup" "Experiment8-Setup" "Experiment3-Setup" "
Experiment1-Setup 6 08:41:07 false 1.0
Experiment2-Setup 1 08:41:07 true 10.0
Experiment7Setup 4 08:41:07 true 30.0
Experiment5-Setup 2 08:41:07 true 42.0
Experiment7-Setup 1 08:41:07 false 45.0
Experiment4-Setup 1 08:41:07 false 54.0
Experiment8-Setup 6 08:41:07 true 54.0
Experiment3-Setup 4 08:41:07 false 60.0
Experiment6-Setup 3 08:41:07 true 67.0
Experiment-Setup 2 08:41:07 true 100.0
************ List was printed as expected ************
-------------------------------------
-------------------------------------
********* orderDay Method Control *********
********* Sorted list According to Their Accuracy For Day1, Day2 and Day6*******:
Experiment2-Setup 1 08:41:07 true 10.0
Experiment7-Setup 1 08:41:07 false 45.0
Experiment4-Setup 1 08:41:07 false 54.0
Experiment5-Setup 2 08:41:07 true 42.0
Experiment-Setup 2 08:41:07 true 100.0
Experiment6-Setup 3 08:41:07 true 67.0
Experiment7Setup 4 08:41:07 true 30.0
Experiment3-Setup 4 08:41:07 false 60.0
Experiment1-Setup 6 08:41:07 false 1.0
Experiment8-Setup 6 08:41:07 true 54.0
************ List was printed as expected *************
```

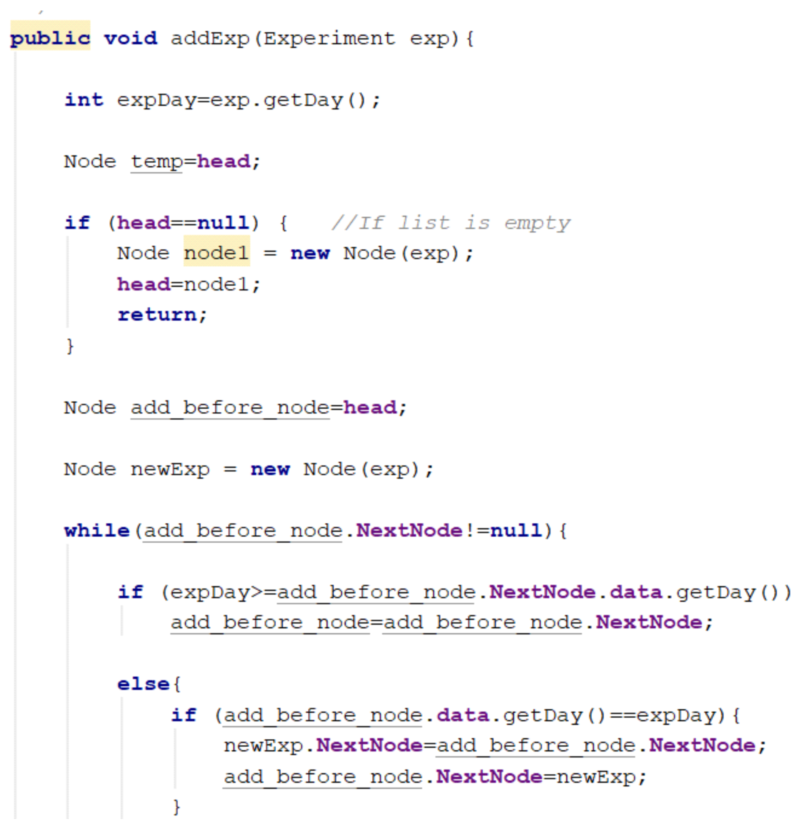
orderExperiments and orderDay methods work correctly. They list experiments according to their accuracy. As you can see list was printed as expected.



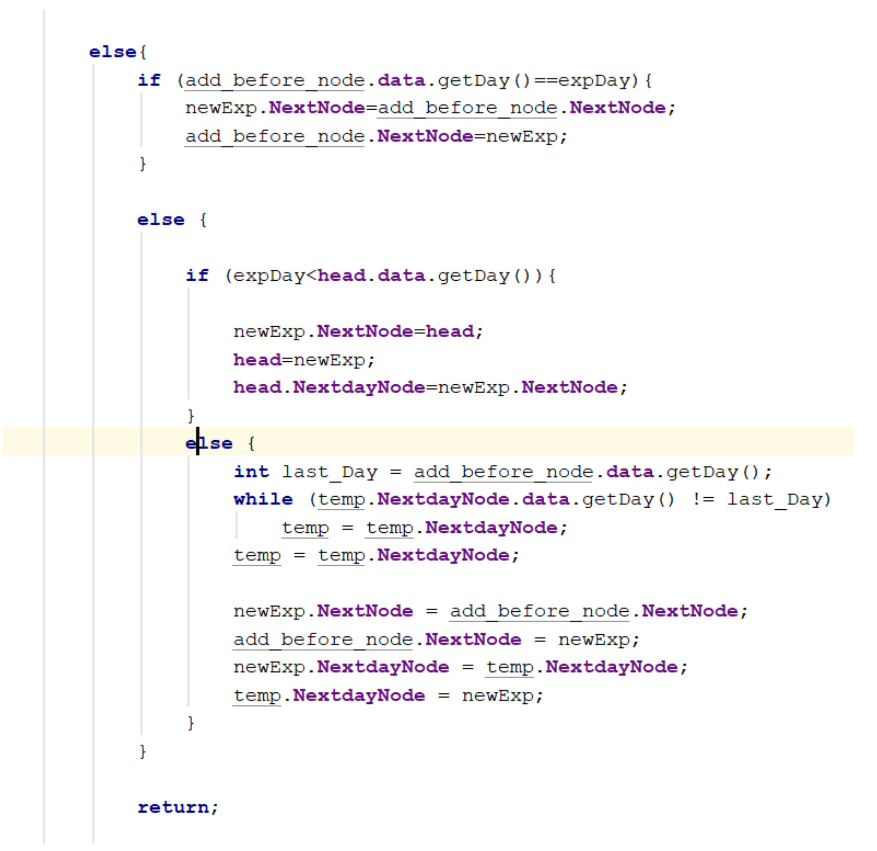
```
************************************ CHECKING FOR ITERATOR **********************************
*************** hasNext and Next Method Control *****************
ExperimentFive 1 08:41:07 false 25.0
ExperimentTwo 2 08:41:07 false 89.0
ExperimentThree 2 08:41:07 true 40.0
ExperimentOne 3 08:41:07 true 90.0
ExperimentFive 5 08:41:07 false 45.0
ExperimentFour 7 08:41:07 true 100.0
************ List was printed by Iterator: ************
************ List was printed as expected *************
--------------------------------------
--------------------------------------
********* We expect ExperimentFour as a lastItemReturned **********
********* So if we use remove method of iterator ExperimentFour will be deleted **********
************* Controlling Remove Method of iterator ******************
************ List was printed again ************
ExperimentFive 1 08:41:07 false 25.0
ExperimentTwo 2 08:41:07 false 89.0
ExperimentThree 2 08:41:07 true 40.0
ExperimentOne 3 08:41:07 true 90.0
ExperimentFive 5 08:41:07 false 45.0
--------------------------------------
--------------------------------------
*************** hasNextDay and NextDay Method Control *****************
ExperimentFive 1 08:41:07 false 25.0
ExperimentTwo 2 08:41:07 false 89.0
ExperimentOne 3 08:41:07 true 90.0
************ List was printed as expected *************
************ NextDayNode Connections are true. *************
--------------------------------------
```

Iterator Works correctly. It has. hasNext, hasNextDay, next, nextDay and remove methods and all of them printed the list as we expect.

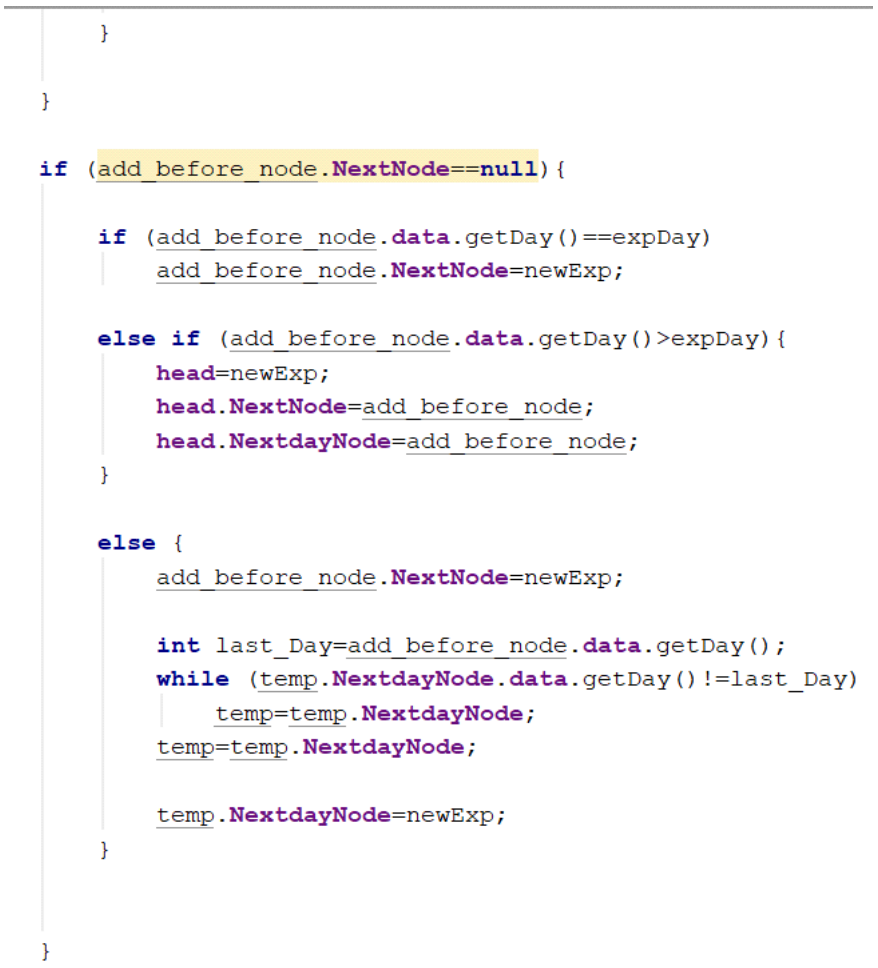
TIME COMPLEXITY ANALYSIS



```java
public void addExp(Experiment exp){

    int expDay=exp.getDay();

    Node temp=head;

    if (head==null) {    //If list is empty
        Node node1 = new Node(exp);
        head=node1;
        return;
    }

    Node add_before_node=head;

    Node newExp = new Node(exp);

    while(add_before_node.NextNode!=null){

        if (expDay>=add_before_node.NextNode.data.getDay())
            add_before_node=add_before_node.NextNode;

        else{
            if (add_before_node.data.getDay()==expDay){
                newExp.NextNode=add_before_node.NextNode;
                add_before_node.NextNode=newExp;
            }
```

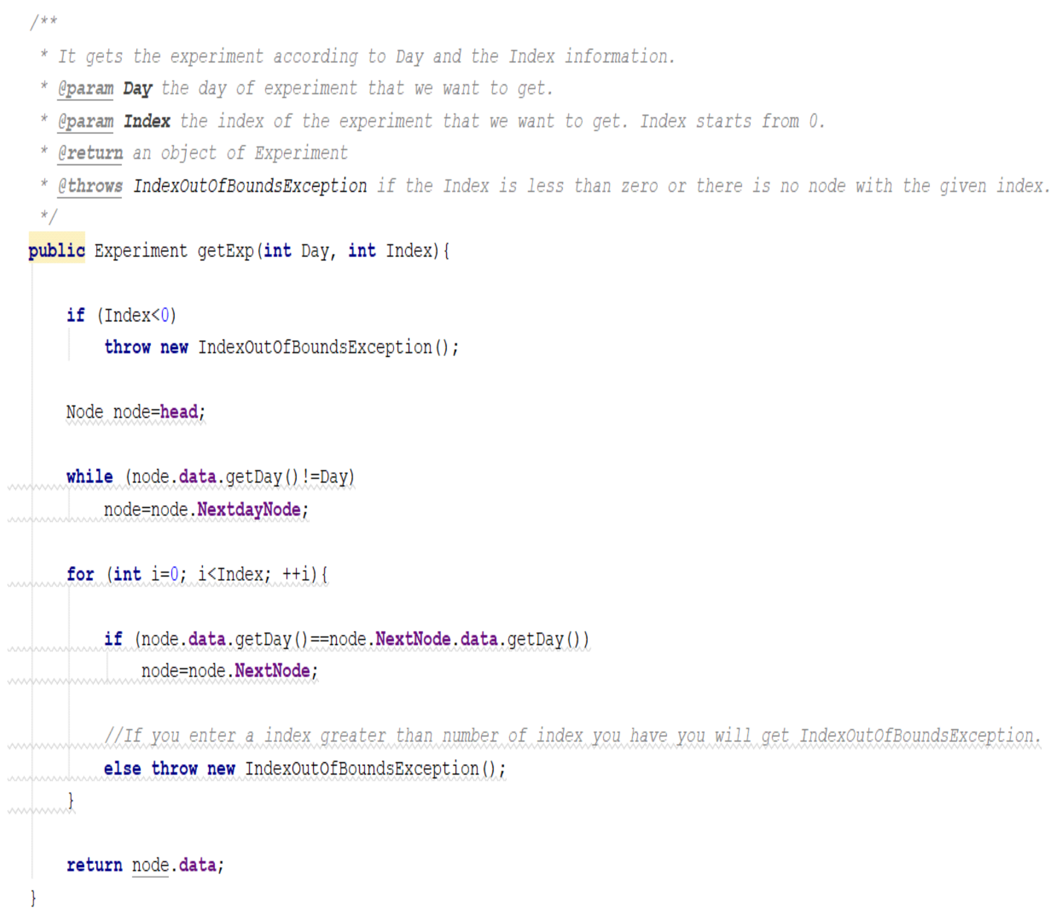


```
        else{
            if (add_before_node.data.getDay()==expDay){
                newExp.NextNode=add_before_node.NextNode;
                add_before_node.NextNode=newExp;
            }

            else {

                if (expDay<head.data.getDay()){

                    newExp.NextNode=head;
                    head=newExp;
                    head.NextdayNode=newExp.NextNode;
                }
                else {
                    int last_Day = add_before_node.data.getDay();
                    while (temp.NextdayNode.data.getDay() != last_Day)
                        temp = temp.NextdayNode;
                    temp = temp.NextdayNode;

                    newExp.NextNode = add_before_node.NextNode;
                    add_before_node.NextNode = newExp;
                    newExp.NextdayNode = temp.NextdayNode;
                    temp.NextdayNode = newExp;
                }
            }

            return;
```



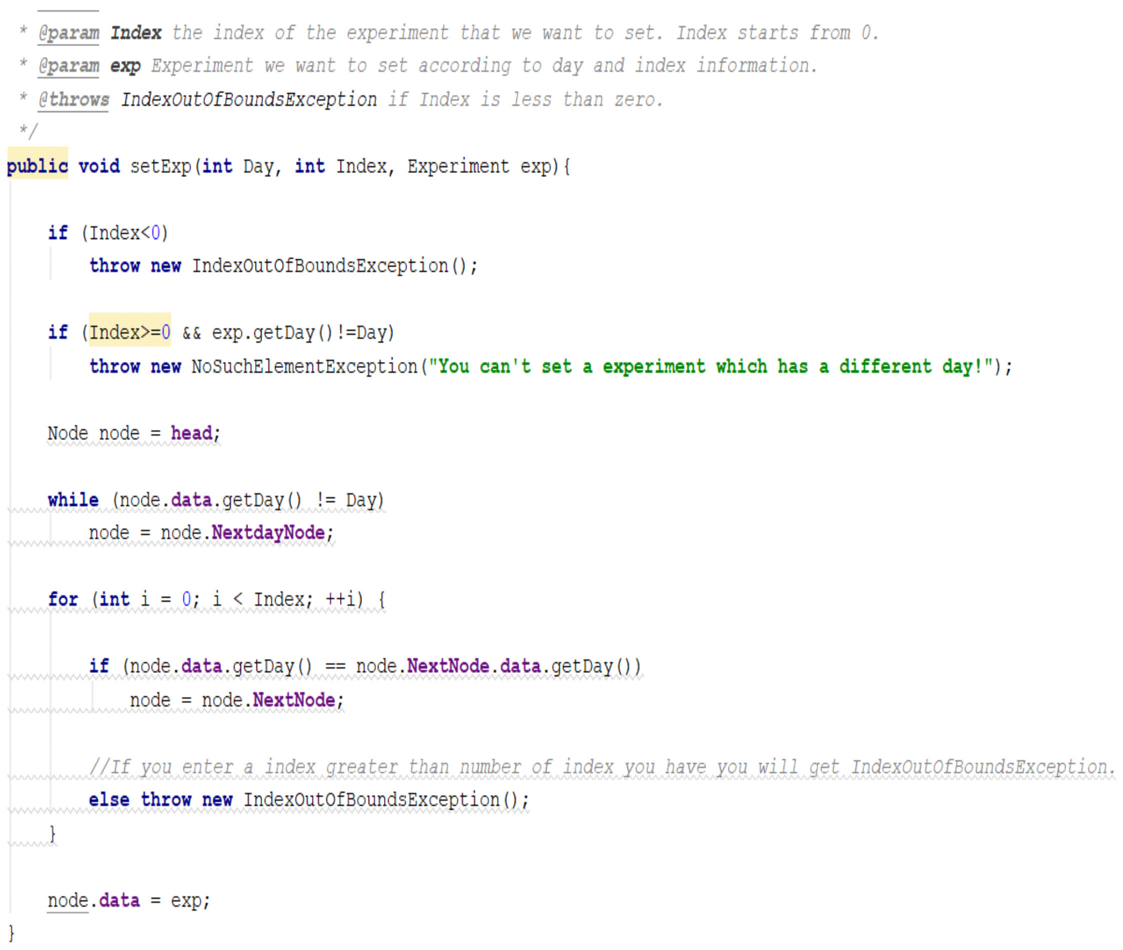
```
        }

    }

    if (add_before_node.NextNode==null){

        if (add_before_node.data.getDay()==expDay)
            add_before_node.NextNode=newExp;

        else if (add_before_node.data.getDay()>expDay){
            head=newExp;
            head.NextNode=add_before_node;
            head.NextdayNode=add_before_node;
        }

        else {
            add_before_node.NextNode=newExp;

            int last_Day=add_before_node.data.getDay();
            while (temp.NextdayNode.data.getDay()!=last_Day)
                temp=temp.NextdayNode;
            temp=temp.NextdayNode;

            temp.NextdayNode=newExp;
        }

    }
```

Time Complexity for addExp Method: For the best case I can say, List is empty. So addExp method will work in O(1) (constant time). But for the worst case it will enter in while loop two times and loop depends on the number of the NextDayNode. So the funciton will works in O(n^2).



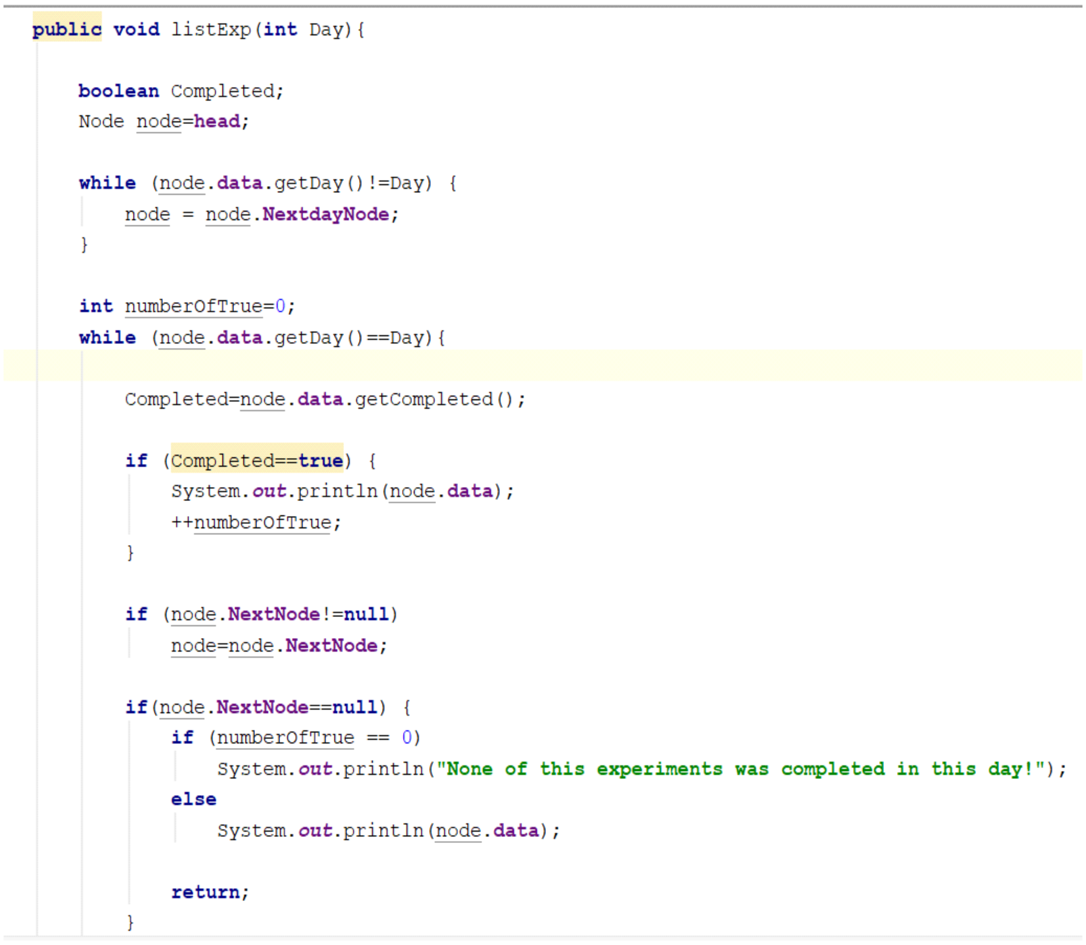
```java
/**
 * It gets the experiment according to Day and the Index information.
 * @param Day the day of experiment that we want to get.
 * @param Index the index of the experiment that we want to get. Index starts from 0.
 * @return an object of Experiment
 * @throws IndexOutOfBoundsException if the Index is less than zero or there is no node with the given index.
 */
public Experiment getExp(int Day, int Index){

    if (Index<0)
        throw new IndexOutOfBoundsException();

    Node node=head;

    while (node.data.getDay()!=Day)
        node=node.NextdayNode;

    for (int i=0; i<Index; ++i){

        if (node.data.getDay()==node.NextNode.data.getDay())
            node=node.NextNode;

        //If you enter a index greater than number of index you have you will get IndexOutOfBoundsException.
        else throw new IndexOutOfBoundsException();
    }

    return node.data;
}
```

Time Complexity for getExp Method: It will enter in while loop and works n times because it depends on the number of the nextDayNode and then it will enter for loop and works n times again because of value of index. So time complexity of this method is O(n).



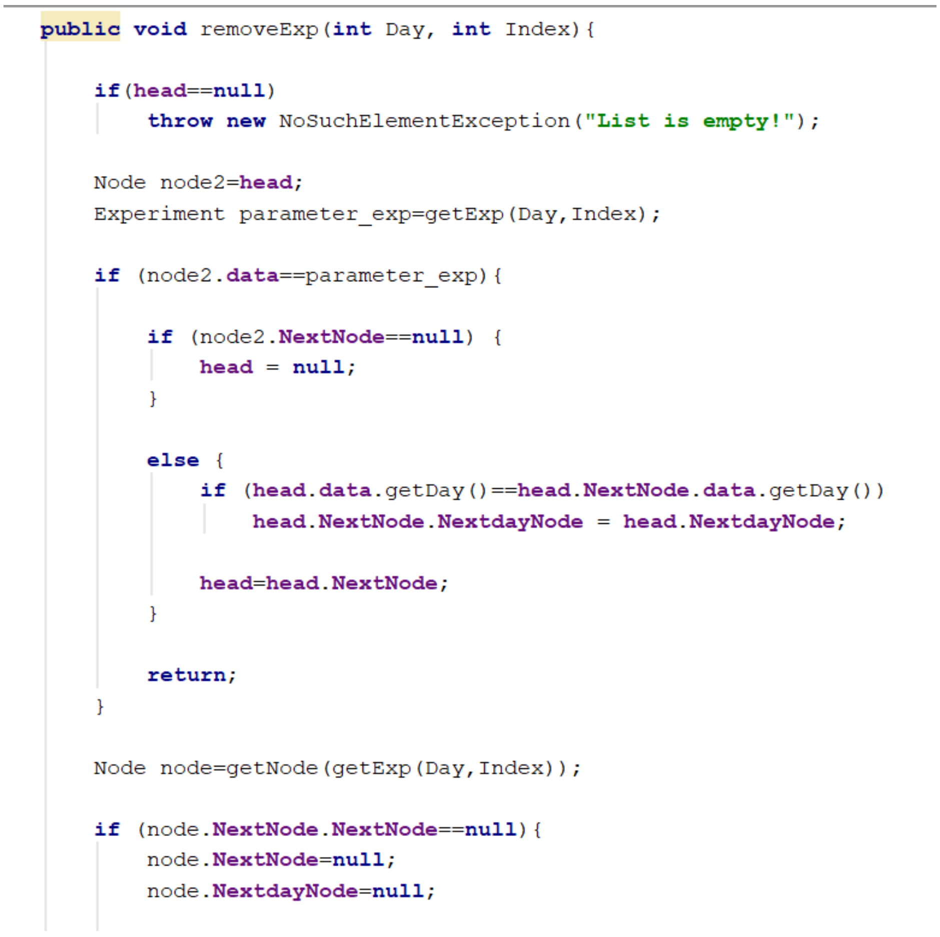
```
 * @param Index the index of the experiment that we want to set. Index starts from 0.
 * @param exp Experiment we want to set according to day and index information.
 * @throws IndexOutOfBoundsException if Index is less than zero.
 */
public void setExp(int Day, int Index, Experiment exp){

    if (Index<0)
        throw new IndexOutOfBoundsException();

    if (Index>=0 && exp.getDay()!=Day)
        throw new NoSuchElementException("You can't set a experiment which has a different day!");

    Node node = head;

    while (node.data.getDay() != Day)
        node = node.NextdayNode;

    for (int i = 0; i < Index; ++i) {

        if (node.data.getDay() == node.NextNode.data.getDay())
            node = node.NextNode;

        //If you enter a index greater than number of index you have you will get IndexOutOfBoundsException.
        else throw new IndexOutOfBoundsException();
    }

    node.data = exp;
}
```

Time Complexity for setExp Method: : It will enter in while loop and works n times because it depends on the number of the nextDayNode and then it will enter for loop and works n times again because of value of index. So time complexity of this method is O(n).

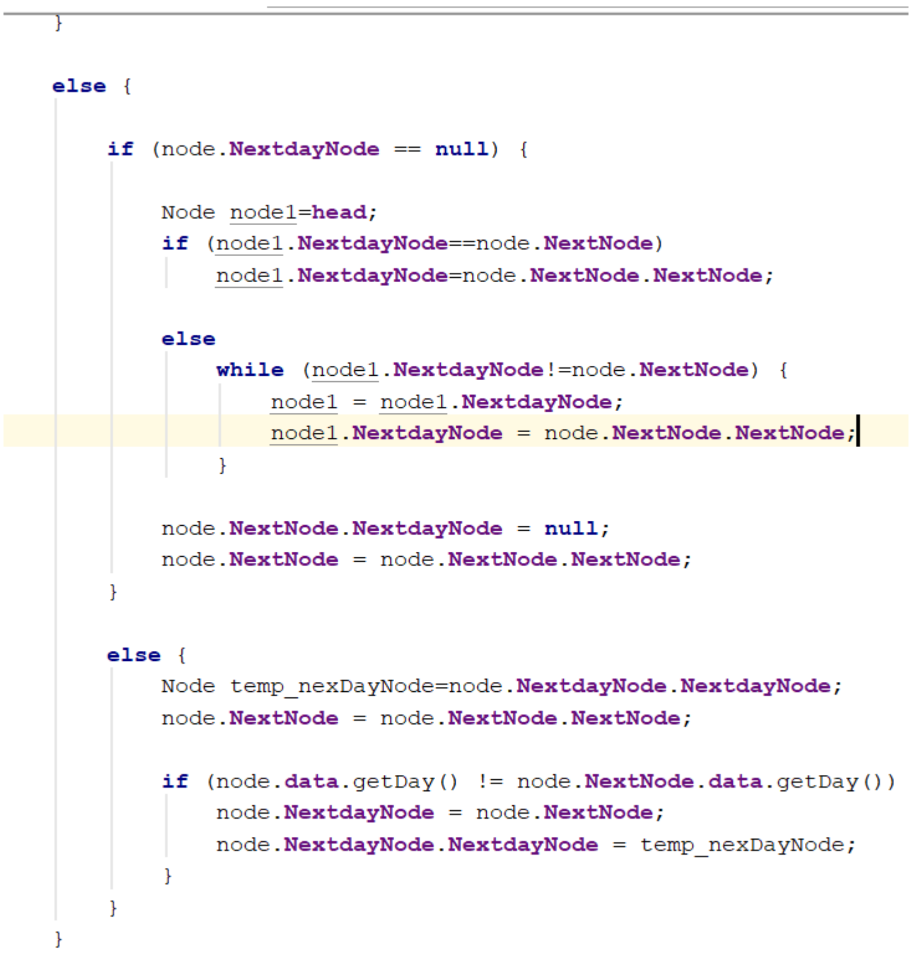


```
public void listExp(int Day){

    boolean Completed;
    Node node=head;

    while (node.data.getDay()!=Day) {
        node = node.NextdayNode;
    }

    int numberOfTrue=0;
    while (node.data.getDay()==Day){

        Completed=node.data.getCompleted();

        if (Completed==true) {
            System.out.println(node.data);
            ++numberOfTrue;
        }

        if (node.NextNode!=null)
            node=node.NextNode;

        if(node.NextNode==null) {
            if (numberOfTrue == 0)
                System.out.println("None of this experiments was completed in this day!");
            else
                System.out.println(node.data);

            return;
        }
```

Time Complexity for listExp Method: It will enter in while loop and works n times because it depends on the number of the nextDayNode and then it will enter while loop again and work n times again . So time complexity of this method is O(n).

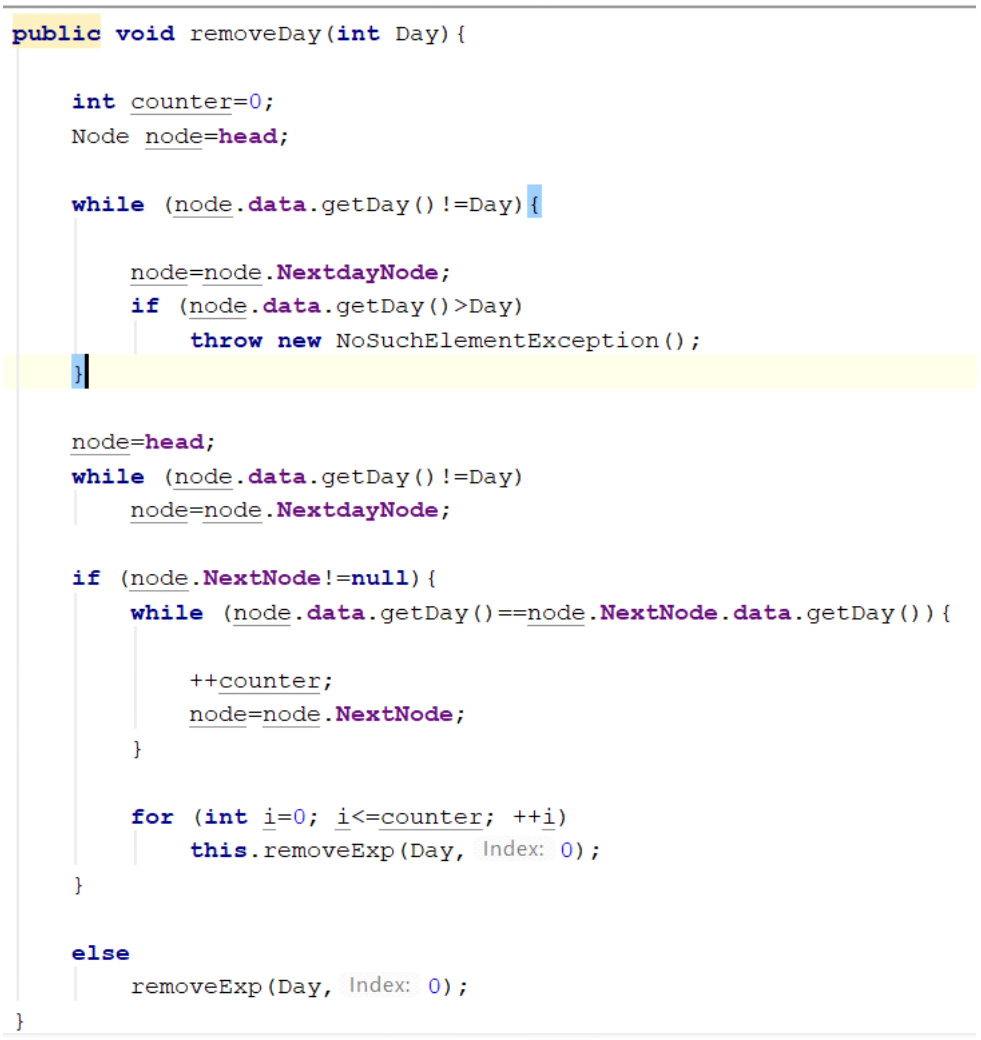


```java
public void removeExp(int Day, int Index){

    if(head==null)
        throw new NoSuchElementException("List is empty!");

    Node node2=head;
    Experiment parameter_exp=getExp(Day,Index);

    if (node2.data==parameter_exp){

        if (node2.NextNode==null) {
            head = null;
        }

        else {
            if (head.data.getDay()==head.NextNode.data.getDay())
                head.NextNode.NextdayNode = head.NextdayNode;

            head=head.NextNode;
        }

        return;
    }

    Node node=getNode(getExp(Day,Index));

    if (node.NextNode.NextNode==null){
        node.NextNode=null;
        node.NextdayNode=null;
```



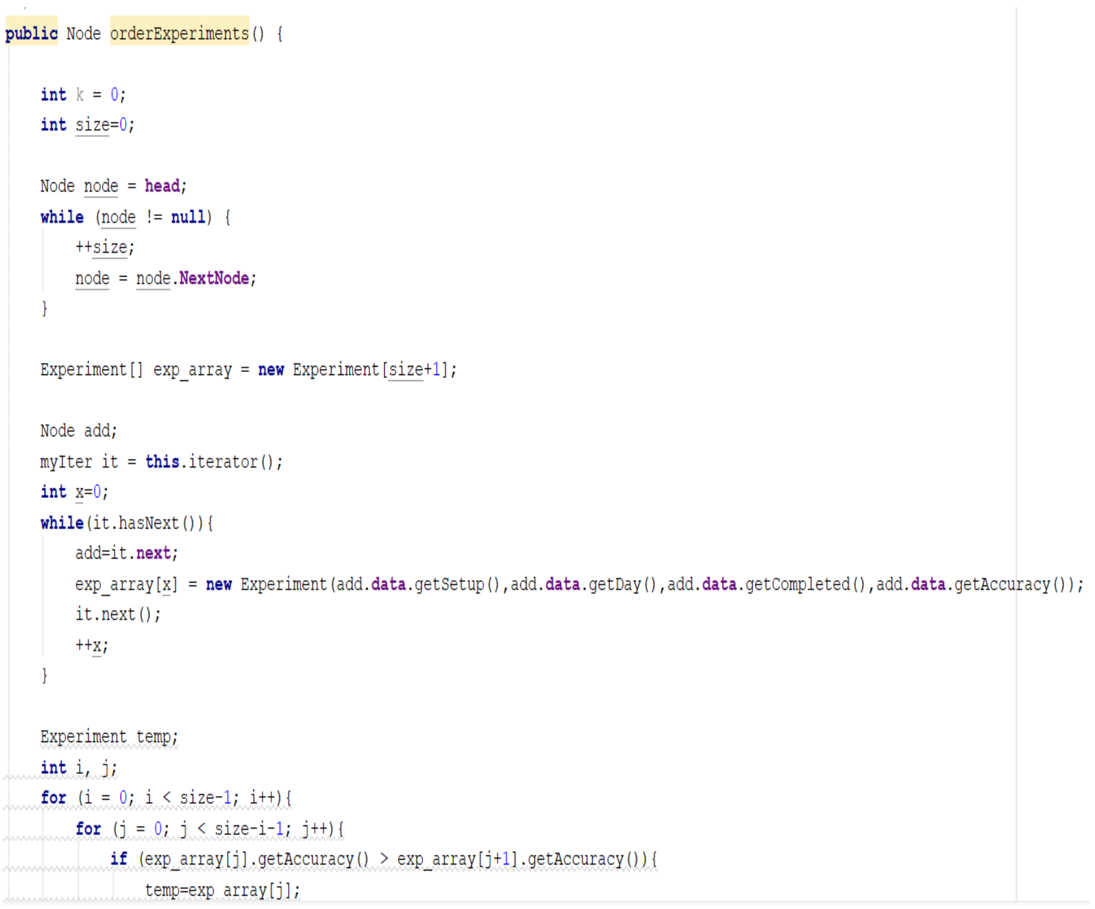
```java
}

else {

    if (node.NextdayNode == null) {

        Node node1=head;
        if (node1.NextdayNode==node.NextNode)
            node1.NextdayNode=node.NextNode.NextNode;

        else
            while (node1.NextdayNode!=node.NextNode) {
                node1 = node1.NextdayNode;
                node1.NextdayNode = node.NextNode.NextNode;
            }

        node.NextNode.NextdayNode = null;
        node.NextNode = node.NextNode.NextNode;
    }

    else {
        Node temp_nexDayNode=node.NextdayNode.NextdayNode;
        node.NextNode = node.NextNode.NextNode;

        if (node.data.getDay() != node.NextNode.data.getDay())
            node.NextdayNode = node.NextNode;
            node.NextdayNode.NextdayNode = temp_nexDayNode;
        }
    }
}
```

Time Complexity for removeExp Method: For the best case I can say, it will enter in the first if statement and whatever it enters , else or if, it doent' matter, the time complexity of this situation is constant time, O(1). For the worst case, it will enter first else statement and while loop. So it works in linear time. O(n) Because while loop depends on some number, so it will work more than once.

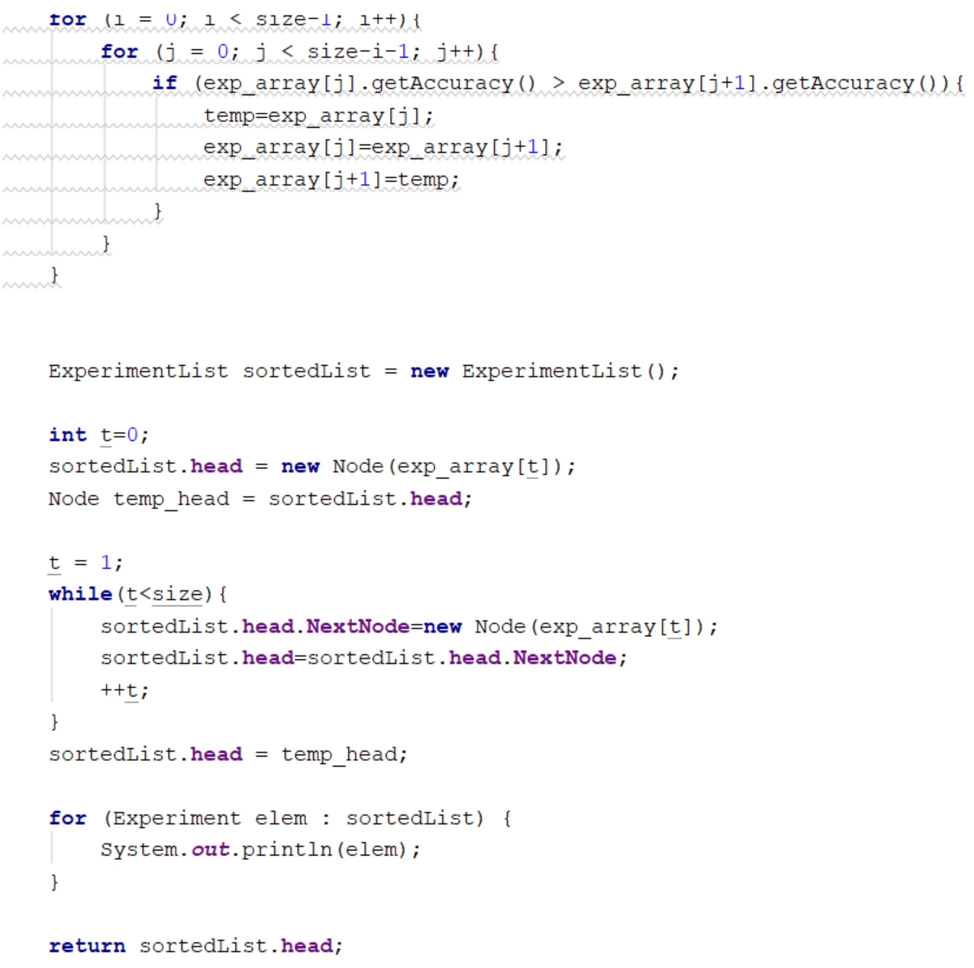


```java
public void removeDay(int Day){

    int counter=0;
    Node node=head;

    while (node.data.getDay()!=Day){

        node=node.NextdayNode;
        if (node.data.getDay()>Day)
            throw new NoSuchElementException();
    }

    node=head;
    while (node.data.getDay()!=Day)
        node=node.NextdayNode;

    if (node.NextNode!=null){
        while (node.data.getDay()==node.NextNode.data.getDay()){

            ++counter;
            node=node.NextNode;
        }

        for (int i=0; i<=counter; ++i)
            this.removeExp(Day, Index: 0);
    }

    else
        removeExp(Day, Index: 0);
}
```

Time Complexity for removeDay Method: It will enter while loop three times but the constants are insignificant. While loop depends on some number, so it will work more than once. So time complexity of this funciton is O(n).

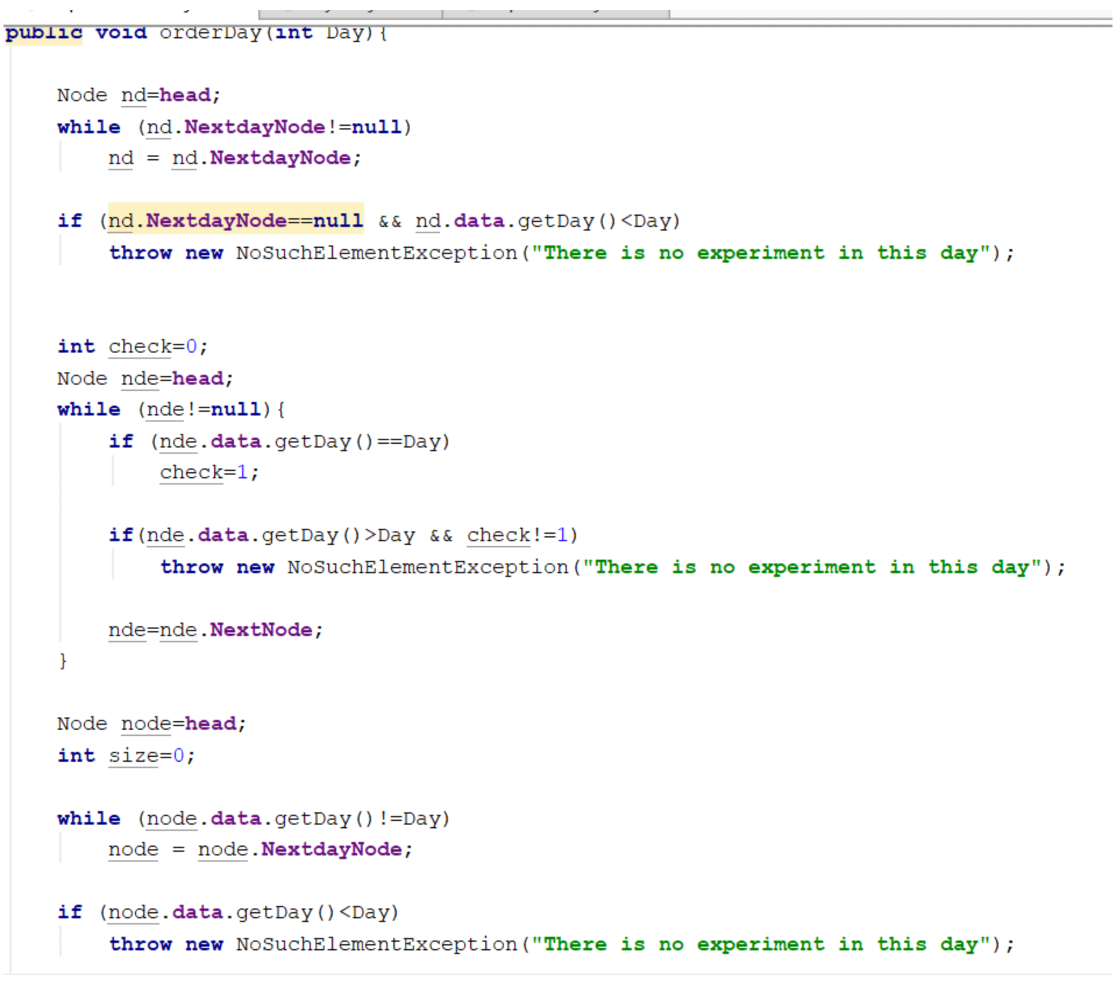


```java
public Node orderExperiments() {

    int k = 0;
    int size=0;

    Node node = head;
    while (node != null) {
        ++size;
        node = node.NextNode;
    }

    Experiment[] exp_array = new Experiment[size+1];

    Node add;
    myIter it = this.iterator();
    int x=0;
    while(it.hasNext()){
        add=it.next;
        exp_array[x] = new Experiment(add.data.getSetup(),add.data.getDay(),add.data.getCompleted(),add.data.getAccuracy());
        it.next();
        ++x;
    }

    Experiment temp;
    int i, j;
    for (i = 0; i < size-1; i++){
        for (j = 0; j < size-i-1; j++){
            if (exp_array[j].getAccuracy() > exp_array[j+1].getAccuracy()){
                temp=exp_array[j];
```

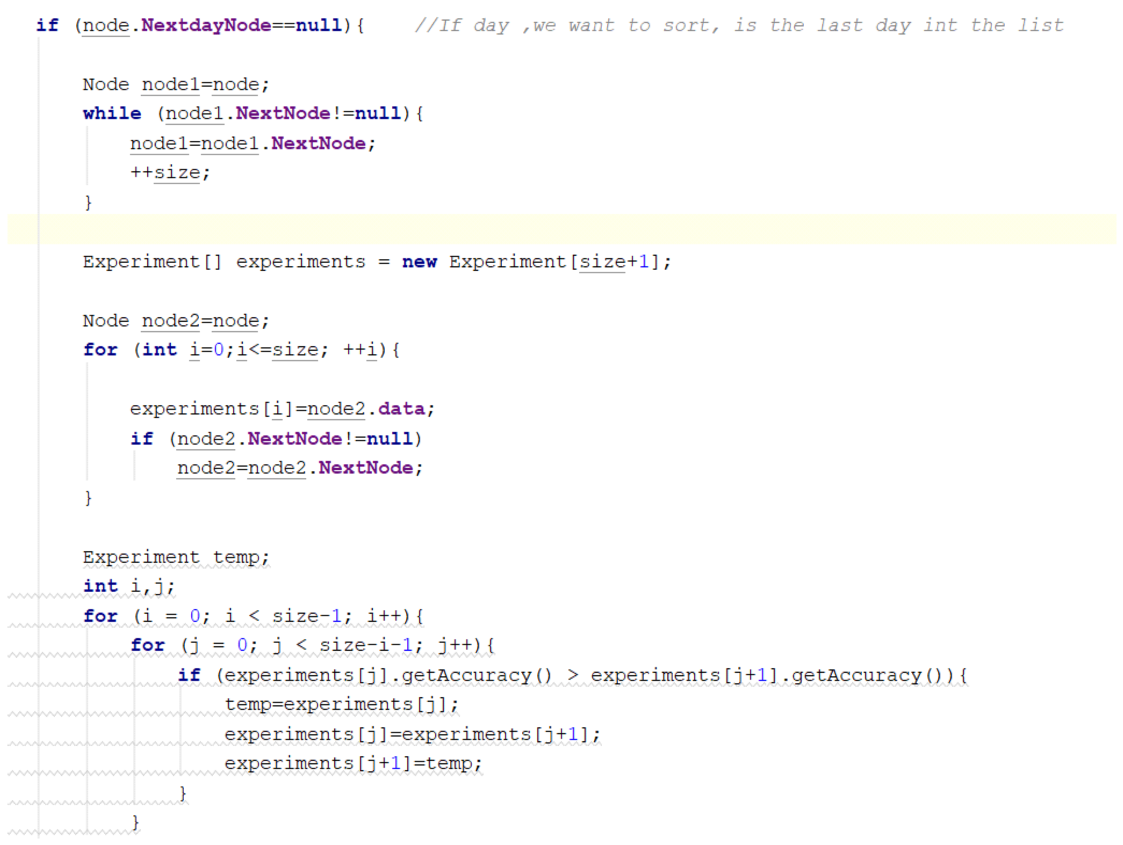


```java
    for (i = 0; i < size-1; i++){
        for (j = 0; j < size-i-1; j++){
            if (exp_array[j].getAccuracy() > exp_array[j+1].getAccuracy()){
                temp=exp_array[j];
                exp_array[j]=exp_array[j+1];
                exp_array[j+1]=temp;
            }
        }
    }


    ExperimentList sortedList = new ExperimentList();

    int t=0;
    sortedList.head = new Node(exp_array[t]);
    Node temp_head = sortedList.head;

    t = 1;
    while(t<size){
        sortedList.head.NextNode=new Node(exp_array[t]);
        sortedList.head=sortedList.head.NextNode;
        ++t;
    }
    sortedList.head = temp_head;

    for (Experiment elem : sortedList) {
        System.out.println(elem);
    }

    return sortedList.head;
```

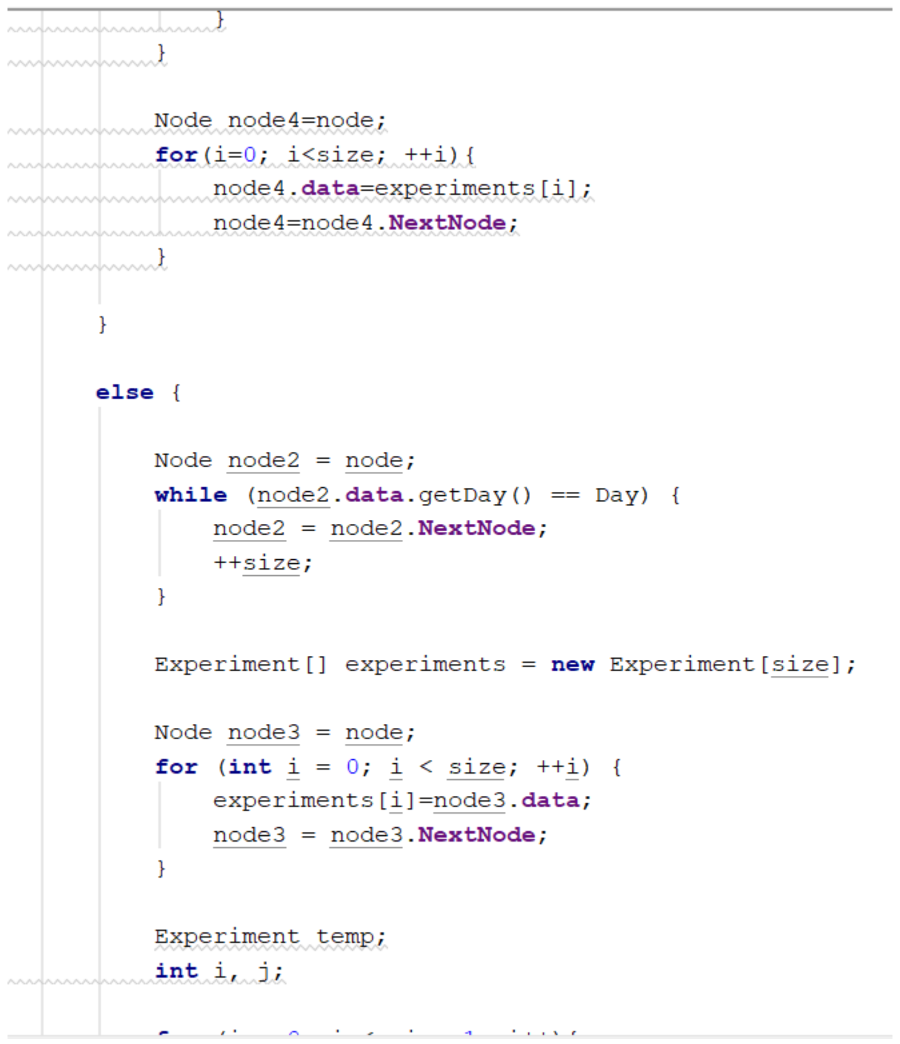
Time Complexity for orderExperiments Method: It has two while loops at first and they work n times. Then it will enter for loop and inner for loop. Two of them works n times, so (n x n) it works in O(n^2) . There are one more while and for loop but they are work n times again and lower terms are unsignificant. As a result time complexity of this method is O(n^2).



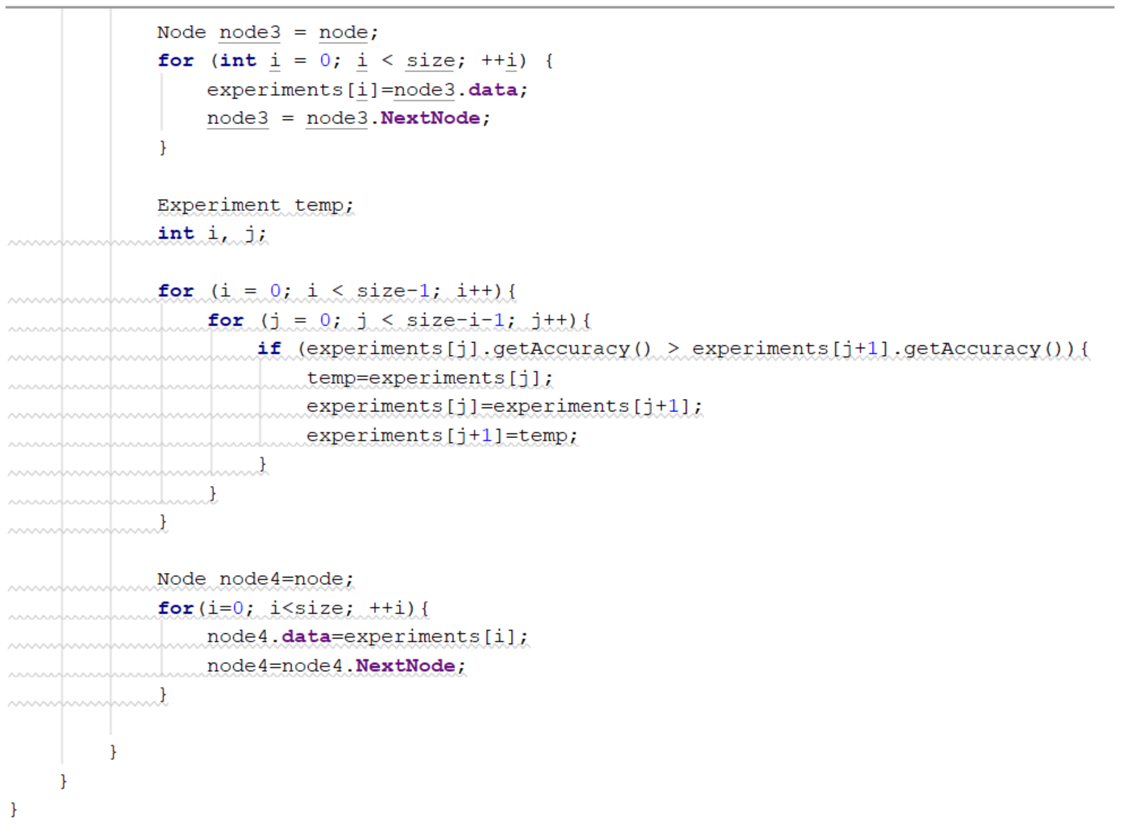
```java
public void orderDay(int Day){

    Node nd=head;
    while (nd.NextdayNode!=null)
        nd = nd.NextdayNode;

    if (nd.NextdayNode==null && nd.data.getDay()<Day)
        throw new NoSuchElementException("There is no experiment in this day");


    int check=0;
    Node nde=head;
    while (nde!=null){
        if (nde.data.getDay()==Day)
            check=1;

        if(nde.data.getDay()>Day && check!=1)
            throw new NoSuchElementException("There is no experiment in this day");

        nde=nde.NextNode;
    }

    Node node=head;
    int size=0;

    while (node.data.getDay()!=Day)
        node = node.NextdayNode;

    if (node.data.getDay()<Day)
        throw new NoSuchElementException("There is no experiment in this day");
```



```java
if (node.NextdayNode==null){     //If day ,we want to sort, is the last day int the list

    Node node1=node;
    while (node1.NextNode!=null){
        node1=node1.NextNode;
        ++size;
    }

    Experiment[] experiments = new Experiment[size+1];

    Node node2=node;
    for (int i=0;i<=size; ++i){

        experiments[i]=node2.data;
        if (node2.NextNode!=null)
            node2=node2.NextNode;
    }

    Experiment temp;
    int i,j;
    for (i = 0; i < size-1; i++){
        for (j = 0; j < size-i-1; j++){
            if (experiments[j].getAccuracy() > experiments[j+1].getAccuracy()){
                temp=experiments[j];
                experiments[j]=experiments[j+1];
                experiments[j+1]=temp;
            }
        }
```



```java
            }
        }

        Node node4=node;
        for(i=0; i<size; ++i){
            node4.data=experiments[i];
            node4=node4.NextNode;
        }

    }

    else {

        Node node2 = node;
        while (node2.data.getDay() == Day) {
            node2 = node2.NextNode;
            ++size;
        }

        Experiment[] experiments = new Experiment[size];

        Node node3 = node;
        for (int i = 0; i < size; ++i) {
            experiments[i]=node3.data;
            node3 = node3.NextNode;
        }

        Experiment temp;
        int i, j;

        for (i = 0; i < size - 1; i++){
```



```
            Node node3 = node;
            for (int i = 0; i < size; ++i) {
                experiments[i]=node3.data;
                node3 = node3.NextNode;
            }

            Experiment temp;
            int i, j;

            for (i = 0; i < size-1; i++){
                for (j = 0; j < size-i-1; j++){
                    if (experiments[j].getAccuracy() > experiments[j+1].getAccuracy()){
                        temp=experiments[j];
                        experiments[j]=experiments[j+1];
                        experiments[j+1]=temp;
                    }
                }
            }

            Node node4=node;
            for(i=0; i<size; ++i){
                node4.data=experiments[i];
                node4=node4.NextNode;
            }

        }
    }
}
```

Time Complexity for orderDay Method: There are while and for loops and all work n times, but there is for loop in a other for loop. So two of them works n times. So (n x n)
this loop has a O(n^2) time complexity. As a result lower terms are not important and the time complexity of this method is found as O(n^2).

AS A RESULT , TIME COMPLEXİTY FOR THE WORST CASE :

addExp	getExp	setExp	listExp	removeExp	removeDay	orderEx.	orderDay.
O(n^2)	**O(n)**	**O(n)**	**O(n)**	**O(n)**	**O(n)**	**O(n^2)**	**O(n^2)**