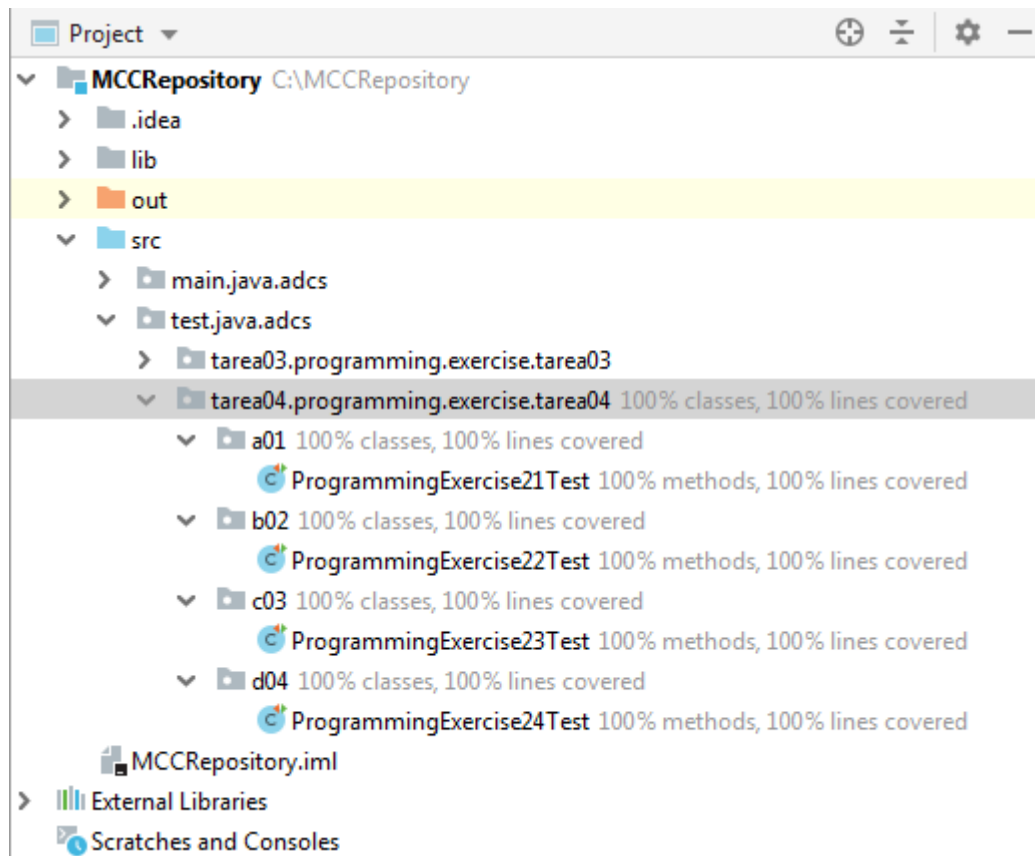
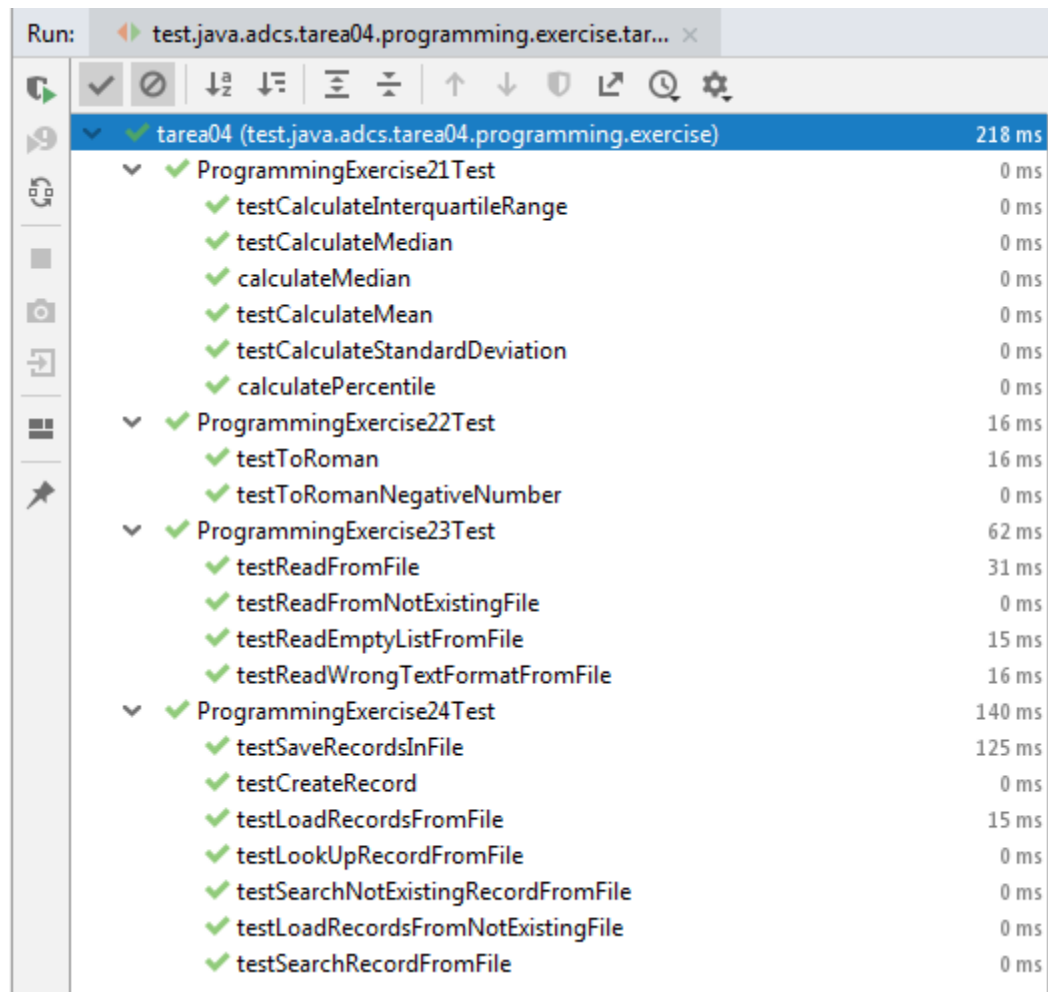


Programming Exercises 21-24



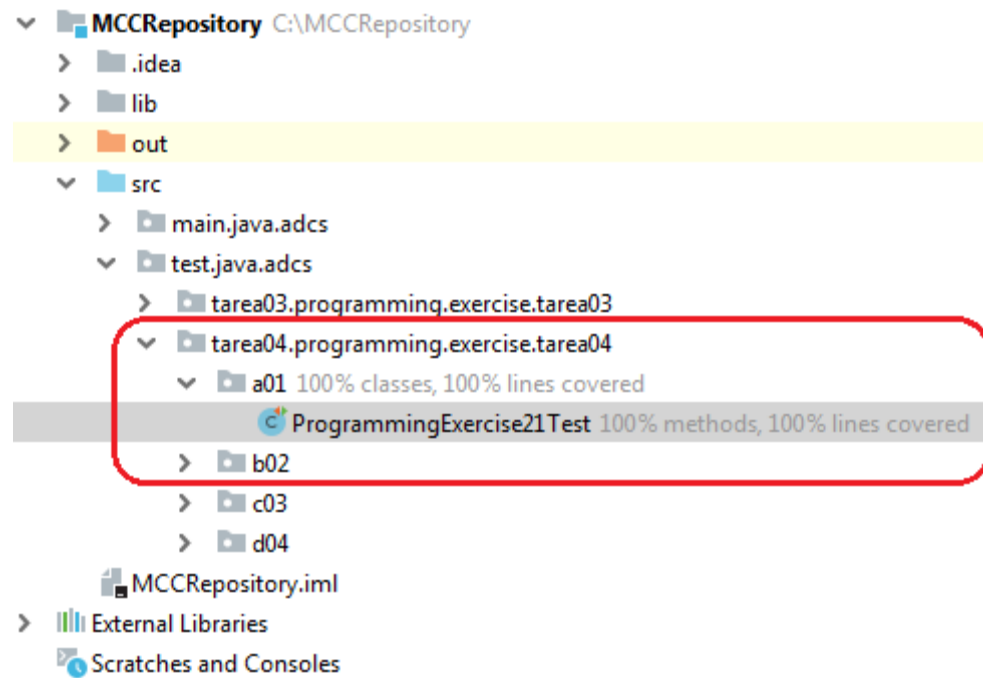


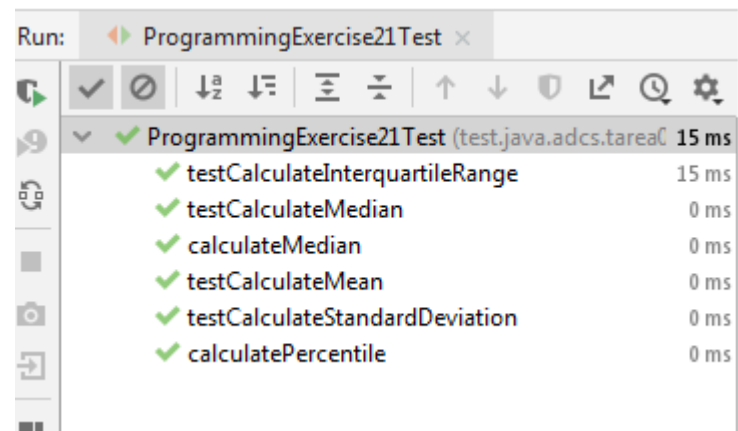
Run:	test.java.adcs.tarea04.programming.exercise.tar...	
▼	✓ tarea04 (test.java.adcs.tarea04.programming.exercise)	218 ms
▼	✓ ProgrammingExercise21Test	0 ms
	✓ testCalculateInterquartileRange	0 ms
	✓ testCalculateMedian	0 ms
	✓ calculateMedian	0 ms
	✓ testCalculateMean	0 ms
	✓ testCalculateStandardDeviation	0 ms
	✓ calculatePercentile	0 ms
▼	✓ ProgrammingExercise22Test	16 ms
	✓ testToRoman	16 ms
	✓ testToRomanNegativeNumber	0 ms
▼	✓ ProgrammingExercise23Test	62 ms
	✓ testReadFromFile	31 ms
	✓ testReadFromNotExistingFile	0 ms
	✓ testReadEmptyListFromFile	15 ms
	✓ testReadWrongTextFormatFromFile	16 ms
▼	✓ ProgrammingExercise24Test	140 ms
	✓ testSaveRecordsInFile	125 ms
	✓ testCreateRecord	0 ms
	✓ testLoadRecordsFromFile	15 ms
	✓ testLookUpRecordFromFile	0 ms
	✓ testSearchNotExistingRecordFromFile	0 ms
	✓ testLoadRecordsFromNotExistingFile	0 ms
	✓ testSearchRecordFromFile	0 ms

Programming Exercises21

The screenshot shows an IDE interface with the following components:

- Project Explorer (Left):** Displays the project structure for 'MCCRepository'. The 'src' folder is expanded, showing 'main.java.adcs' and 'test.java.adcs'. Under 'test.java.adcs', the 'tarea03.programming.exercise.tarea03' folder is selected, and its sub-folder 'tarea04.prog' is also expanded, showing sub-folders 'a01', 'b02', 'c03', and 'd04'.
- Code Editor (Right):** Displays the source code for 'ProgrammingExercise21Te'. The code includes imports for 'org.junit.Test' and 'org.junit.Assert', a Javadoc comment, and the start of a public class definition.
- Context Menu (Center):** A right-click context menu is open over the 'tarea04.prog' folder. The menu items include: 'New', 'Cut', 'Copy', 'Copy Path', 'Copy Reference', 'Paste', 'Find Usages', 'Find in Path...', 'Replace in Path...', 'Analyze', 'Refactor', 'Add to Favorites', 'Show Image Thumbnails', 'Reformat Code', 'Optimize Imports', 'Delete...', 'Build Module 'MCCRepository'', 'Rebuild '...ng.exercise.tarea04'', 'Run 'Tests in 'test.java.adcs.tarea04.programming.exercise.tarea04'' (highlighted with a red box), 'Debug 'Tests in 'test.java.adcs.tarea04.programming.exercise.tarea04'', 'Run 'Tests in 'test.java.adcs.tarea04.programming.exercise.tarea04'' with Coverage' (highlighted with a red box), 'Create 'Tests in 'test.java.adcs.tarea04.programming.exercise.tarea04''...', and 'Show in Explorer'.
- Run Console (Bottom):** Shows the output of the 'Run' command, listing several test methods: 'testSaveReco', 'testCreateReco', 'testLoadReco', and 'testUnkUnRe'.








The screenshot shows the 'Run' window of an IDE. At the top, there is a tab labeled 'Run: ProgrammingExercise21Test'. Below the tab is a toolbar with icons for running, stopping, and other actions. The main area displays a list of test results. The first item is 'ProgrammingExercise21Test' with a green checkmark and a duration of '15 ms'. Below it are six individual test methods, each with a green checkmark and a duration of '0 ms'.

Test Method	Duration
✓ ProgrammingExercise21Test	15 ms
✓ testCalculateInterquartileRange	15 ms
✓ testCalculateMedian	0 ms
✓ calculateMedian	0 ms
✓ testCalculateMean	0 ms
✓ testCalculateStandardDeviation	0 ms
✓ calculatePercentile	0 ms

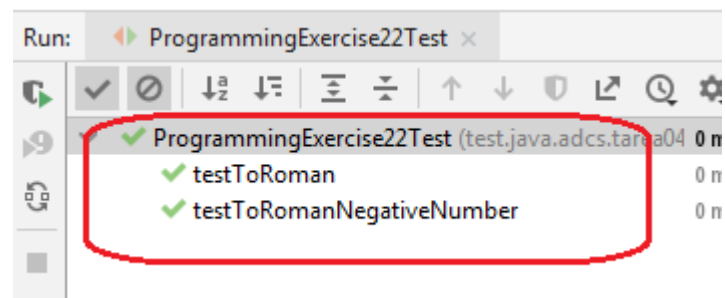
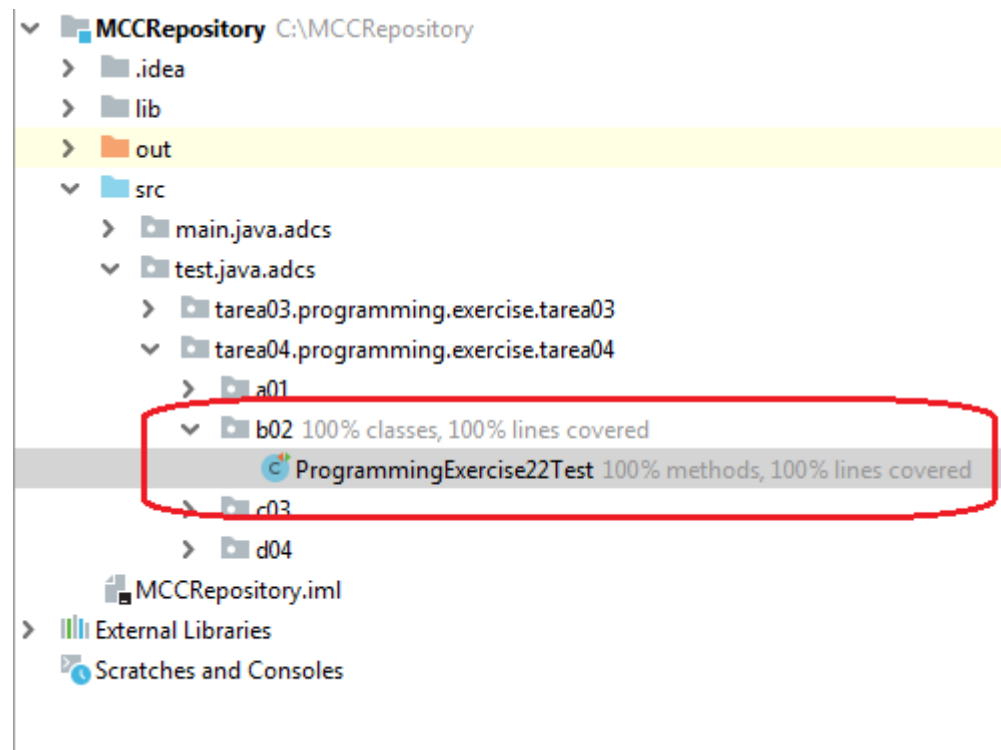
```
ProgrammingExercise21Test.java x
1 package test.java.adcs.tarea04.programming.exercise.tarea04.a01;
2
3 import main.java.adcs.tarea01.programming.exercise.h08.ProgrammingExercise08;
4 import org.junit.Before;
5 import org.junit.Test;
6 import static org.junit.Assert.assertEquals;
7
8 /**
9  * This class is the Programming Exercise 21 used to test the
10  * Programming Exercise 08
11  */
12 public class ProgrammingExercise21Test {
13
14     private ProgrammingExercise08 exercise08;
15     private Double [] numbers;
16
17     @Before
18     public void setup(){
19         exercise08 = new ProgrammingExercise08();
20         numbers = new Double[]{1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0};
21     }
22
23     @Test
24     public void testCalculateMean(){
25         Double mean = exercise08.calculateMean(numbers);
26         Double expected = 5.5d;
27         assertEquals(mean, expected);
28     }
```

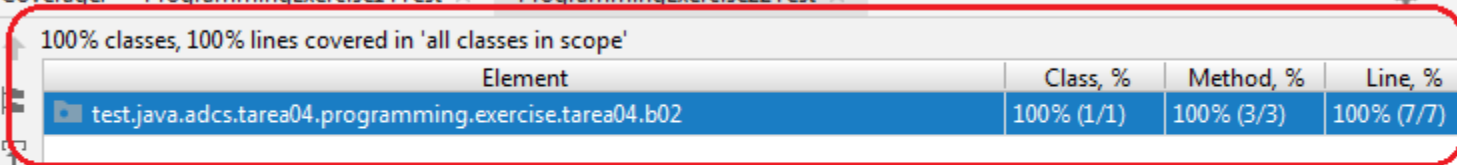
```
ProgrammingExercise21Test.java x
28 }
29
30 @Test
31 public void testCalculateStandardDeviation() {
32     Double standardDeviation = exercise08.calculateStandardDeviation(numbers);
33     Double expected = 2.6076809620810595d;
34     assertEquals(standardDeviation, expected);
35 }
36
37 @Test
38 public void testCalculateMedian() {
39     Double median = exercise08.calculateMedian(numbers);
40     Double expected = 5.5d;
41     assertEquals(median, expected);
42 }
43
44 @Test
45 public void testCalculateInterquartileRange() {
46     Double interquartileRange = exercise08.calculateInterquartileRange(numbers);
47     Double expected = 6.0d;
48     assertEquals(interquartileRange, expected);
49 }
50
51 @Test
52 public void calculateMedian() {
53     Integer median = exercise08.median(1, 2, 3, 6);
54     Integer expected = 4;
55     assertEquals(median, expected);
56 }
57
```

```
57  
58  
59    
60  
61  
62  
63   
64  
65  
66
```

```
@Test  
public void calculatePercentile(){  
    Double percentile = exercise08.calculatePercentile( percentile: 20, numbers);  
    Double expected = 2.0d;  
    assertEquals(percentile, expected);  
}
```


Programming Exercises22



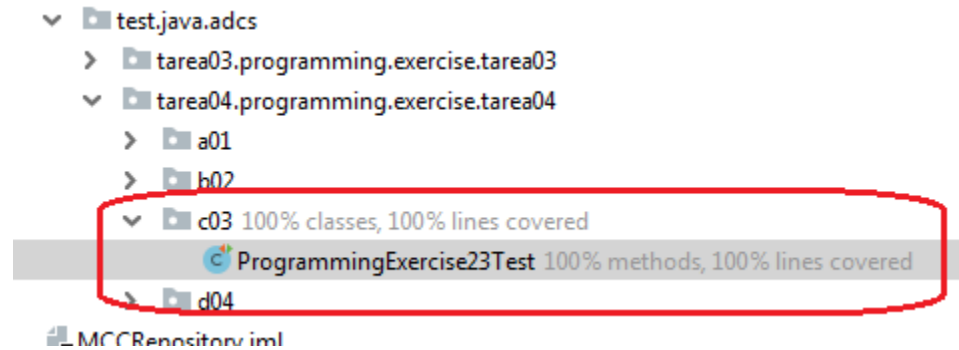


Coverage: ProgrammingExercise14Test x ProgrammingExercise22Test x

100% classes, 100% lines covered in 'all classes in scope'

Element	Class, %	Method, %	Line, %
test.java.adcs.tarea04.programming.exercise.tarea04.b02	100% (1/1)	100% (3/3)	100% (7/7)

Programming Exercises23



```
ProgrammingExercise23Test.java x
1 package test.java.adcs.tarea04.programming.exercise.tarea04.c03;
2
3 import main.java.adcs.tarea02.programming.exercise.a01.MyPowerList;
4 import org.junit.After;
5 import org.junit.Before;
6 import org.junit.Test;
7 import java.io.File;
8 import java.io.FileNotFoundException;
9 import java.io.IOException;
10 import java.util.List;
11 import static org.junit.Assert.assertEquals;
12
13 /**
14  * This class is the Programming Exercise 23 used to test the
15  * Programming Exercise 14
16  */
17 public class ProgrammingExercise23Test {
18
19     private MyPowerList list;
20     private String fileName;
21
22     @Before
23     public void setup() {
24         list = new MyPowerList();
25         fileName = "C:\\temp\\output.txt";
26         new File(fileName).delete();
27     }
28
29     @Test(expected = FileNotFoundException.class)
30     public void testReadFromNotExistingFile() throws IOException {
31         list.readPeopleFromFile(fileName);
32     }
33 }
```

```
33 @Test
34 public void testReadFromFile() throws IOException {
35     list.add("100");
36     list.add("101");
37     list.add("102");
38     list.saveToFile();
39     List<String[]> lines = list.readFromFile(fileName);
40     String[] record = null;
41     for(int i=0; i < lines.size(); i++){
42         record = lines.get(i);
43         for(String item : record){
44             item = item.replace( target: "[", replacement: "").replace( target: "]", replacement: "").trim();
45         }
46     }
47     assertEquals(lines.size(), actual: 1);
48     assertEquals(record.length, actual: 3);
49 }
```

```
51 @Test
52 public void testReadWrongTextFormatFromFile() throws IOException {
53     list.add("a");
54     list.add("jfk");
55     list.add("XHGC");
56     list.add("8");
57     list.saveToFile();
58     List<String[]> lines = list.readFromFile(fileName);
59     String[] record = null;
60     for(int i=0; i < lines.size(); i++){
61         record = lines.get(i);
62         for(String item : record){
63             item = item.replace( target: "[", replacement: "").replace( target: "]", replacement: "").trim();
64         }
65     }
66     assertEquals(lines.size(), actual: 1);
67     assertEquals(record.length, actual: 4);
68 }
```

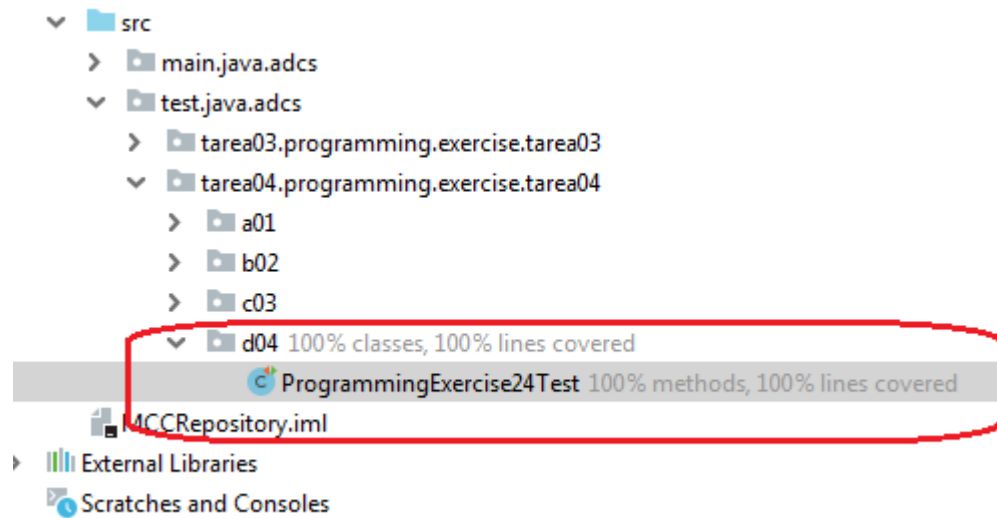
```
69
70     @Test
71     public void testReadEmptyListFromFile() throws IOException {
72         list.clear();
73         list.saveToFile();
74         List<String[]> lines = list.readFromFile(fileName);
75         String[] record = null;
76         for(int i=0; i < lines.size(); i++){
77             record = lines.get(i);
78             for(String item : record){
79                 item = item.replace( target: "[", replacement: "").replace( target: "]", replacement: "").trim();
80             }
81         }
82         assertEquals(lines.size(), actual: 1);
83         assertEquals(record.length, actual: 1);
84     }
85
86     @After
87     public void cleanUp() { new File(fileName).delete(); }
88
89 }
90
91
92
```

Coverage: ProgrammingExercise14Test × ProgrammingExercise23Test ×

100% classes, 100% lines covered in 'all classes in scope'

Element	Class, %	Method, %	Line, %
test.java.adcs.tarea04.programming.exercise.tarea04.c03	100% (1/1)	100% (6/6)	100% (46/46)

Programming Exercises24



```

ProgrammingExercise24Test.java x
1  package test.java.adcs.tarea04.programming.exercise.tarea04.d04;
2
3  import main.java.adcs.tarea02.programming.exercise.a01.MyPowerList;
4  import main.java.adcs.tarea02.programming.exercise.f06.Person;
5  import org.junit.After;
6  import org.junit.Before;
7  import org.junit.Test;
8  import java.io.File;
9  import java.io.FileNotFoundException;
10 import java.io.IOException;
11 import java.util.List;
12 import static org.junit.Assert.assertEquals;
13 import static org.junit.Assert.assertTrue;
14
15 /**
16  * This class is the Programming Exercise 24 used to test the
17  * Programming Exercise 15
18  */
19 public class ProgrammingExercise24Test {
20     private static final String AGENDA_ARTISTAS_Y_ACTRICES_CINE_MEXICANO = "C:\\temp\\personajesCineMexicano.txt";
21     MyPowerList<Person> agenda;
22
23     @Before
24     public void setup() { agenda = new MyPowerList(); }
25
26
27
28     @Test
29     public void testCreateRecord(){
30         agenda.add(new Person( name: "Juan Camaney", address: "Calle de la viuda sin numero", phone: "1234567890", email: "juan.camane@cinedeoro.com.mx"));
31         agenda.add(new Person( name: "Mauricio Garces", address: "Calle - modisto de señoras copetonas", phone: "9876543210", email: "modisto.de.rucas@eltodasmias.com"));
32         assertEquals(agenda.size(), actual: 2);
33     }
34

```

```

ProgrammingExercise24Test.java x
33     }
34
35     @Test
36     public void testSaveRecordsInFile(){
37         agenda.add(new Person( name: "Juan Camaney", address: "Calle de la viuda sin numero", phone: "1234567890", email: "juan.camane@cinedeoro.com.mx"));
38         agenda.add(new Person( name: "Mauricio Garces", address: "Calle - modisto de señoras copetonas", phone: "9876543210", email: "modisto.de.rucas@eltodasmias.com"));
39         agenda.saveToFile(AGENDA_ARTISTAS_Y_ACTRICES_CINE_MEXICANO);
40         assertTrue(new File(AGENDA_ARTISTAS_Y_ACTRICES_CINE_MEXICANO).exists());
41     }
42

```



```
42
43
44 @Test
45 public void testLoadRecordsFromFile() throws IOException {
46     agenda.readPeopleFromFile(AGENDA_ARTISTAS_Y_ACTRICES_CINE_MEXICANO);
47 }
48
49 @Test(expected = FileNotFoundException.class)
50 public void testLoadRecordsFromNotExistingFile() throws IOException {
51     agenda.readPeopleFromFile(fileName: "C:\\unknownFile.txt"); }
52
53 @Test
54 public void testSearchRecordFromFile() throws IOException {
55     agenda.add(new Person( name: "Said Olano", address: "Av Aviacion sin Numero", phone: "9876543210", email: "josesaid@gmail.com"));
56     agenda.saveToFile(AGENDA_ARTISTAS_Y_ACTRICES_CINE_MEXICANO);
57     List<Person> people = agenda.findByName( nameToSearch: "Said Olano");
58     assertEquals(people.iterator().next().getName(), actual: "Said Olano");
59 }
60
61 @Test
62 public void testSearchNotExistingRecordFromFile(){
63     List<Person> people = agenda.findByName( nameToSearch: "Juana la cubana");
64     assertEquals(people.size(), actual: 0);
65 }
```

```
65
66
67 @Test
68 public void testLookUpRecordFromFile(){
69     agenda.add(new Person( name: "Mauricio Garces", address: "Calle - modisto de señoras copetonas", phone: "9876543210", email: "modisto.de.rucas@eltodasmias.com"));
70     List<Person> people = agenda.findByName( nameToSearch: "Mauricio Garces");
71     assertEquals(people.get(0).getPhone(), actual: "9876543210");
72 }
73
74 @After
75 public void cleanUp() { agenda.clear(); }
76
77 }
78
```

