Gebze Technical University
Computer Engineering

CSE 222 - 2018 Spring

HOMEWORK 3 REPORT

MUHAMMED OKUMUŞ
151044017

Course Assistant: Mehmet Burak KOCA

1. INTRODUCTION

1.1 Problem Definition

We need to make a manipulatable catalog for university courses. User needs to be able to retrieve parts of the catalog by given parameters. Also in some cases admin should be able to disable courses to prevent them from further manipulation. Admin should be able to enable these courses if it’s necessary. Operations such as adding, removing and accessing these courses are depended on item being enabled or not.

1.2 System Requirements

For starters, courses should be retrieved from a comma separated file. File is expected to be formatted as the following:

- Six values, each line, specifying course information.
- Value order goes as semester, course code, title, credit type 1, credit type 2, weights.
- No empty lines.

After confirming that we have proper file, we can go ahead and start using operations on it. Operations depends on the type of management system we are using.

1.2.1 Course Management System(CMS)

All operations under this system requires a “CourseManagmentSystem” object, after user is created an object of this type and initialized with a proper file they are able:

- Access a course by entering its course code.
- Access a course by entering its index.
- Change a courses information in the catalog that they accessed with the above methods.
- Retrieve all courses in the catalog.
- Retrieve courses in given range of semester. (Starting semester to ending semesters both which included)

Some courses are only accessible by their indexes, these are optional courses and their course codes are “XXX XXX”. This is to prevent accidently accessing an optional course.

1.2.2 Advanced Course Management System

This system utilizes the standard Course Management System(CMS) and adds new traits to enforce preventive measures on the user such as enable and disable options as we mentioned before. Same as CMS, this system also requires a “advancedCSM” object to function. After creating the object, user can access all methods of classic CSM in addition to:

- Removing a course from the catalog.
- Creating an iterator from a specified elements index.
- Enabling and disabling courses.
- Displaying disabled courses.

To explain these functionalities better let’s start with disable. Disable option is usable on all courses in the catalog. Except already disabled items, which has no purpose. When a course is disabled, it’s excluded from being displayed in catalog, cannot be set to new values, disabled item’s index cannot be used to create and iterator from it, and lastly item cannot be removed. Enabling an item, restores its access to these functions.

1.2.3 Part 3 Course Management System

This part is incomplete.

2. METHOD

2.1 Class Diagrams

(This is the minimal version of the diagram to fit the document properly. Bigger version is in assignment folder.)

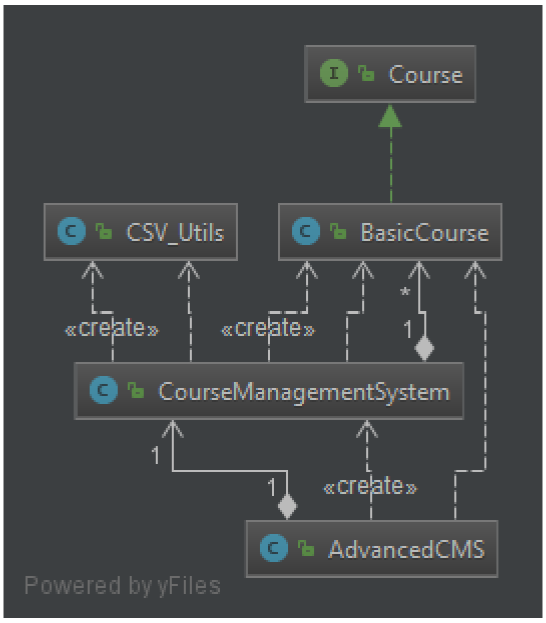


Figure 1

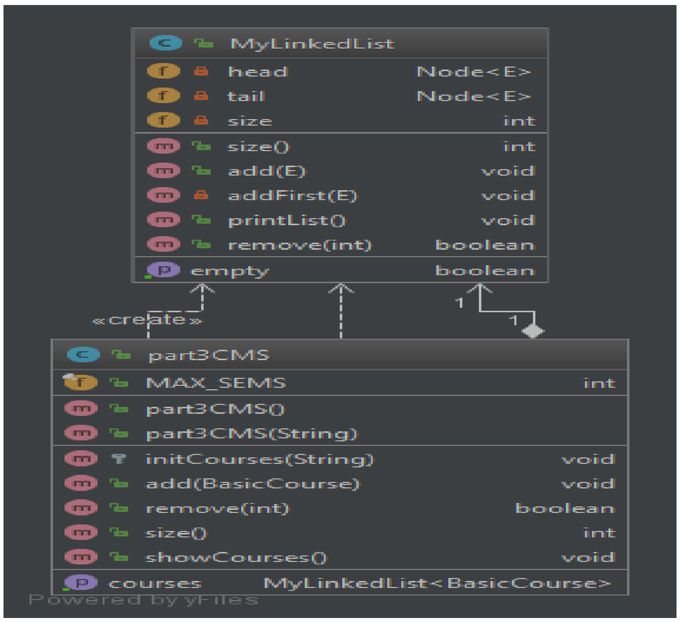


Figure 2

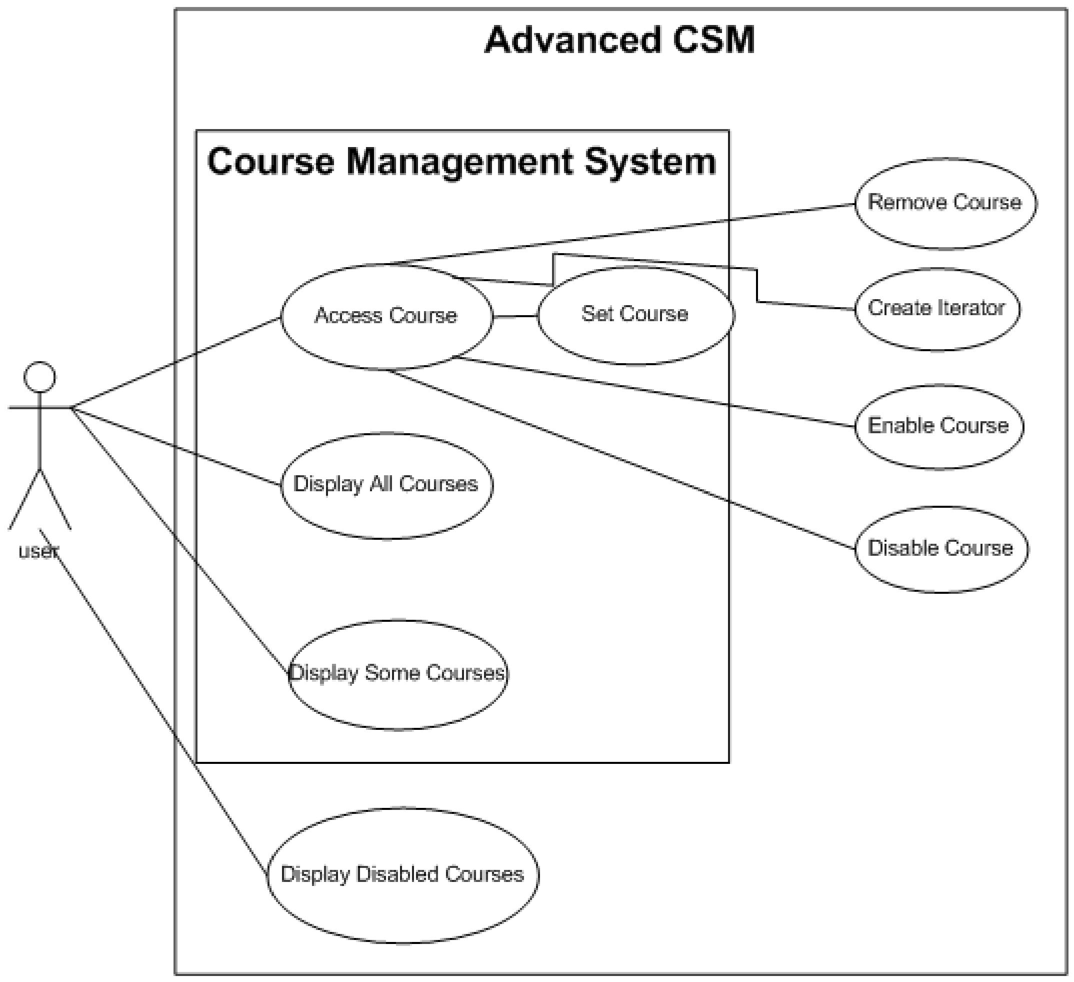
2.1.1 Figure 1

Let's start by small parts of the diagram. We have a Course Interface which specifies the methods a BasicCourse class should have. A Basic Course is used by CMS and AdvancedCMS as it represents a single course in the catalog. We can see that there is many to one relation between BasicCourse and CMS'. One other thing to note is that CMS uses an instance of CSV_Utils. Purpose of this is to read courses from a file to initialize the catalog. Also, as we mentioned, AdvancedCMS uses an instance of CMS to access its useful methods and not to invent wheel again.

2.1.2 Figure 2

This part is incomplete.

2.2 Use Case Diagrams



As we can see from diagram, AdvancedCSM can benefit from all public CSM methods.

2.3 Problem Solution Approach

First time reading the problem, most basic thing we will need is an object to hold a course information. So, I started by creating a Course class to hold course information for a single course. Then to manipulate all the courses in the catalog, I created another class called Course Management System which has a data field to store courses. To store the courses, I used LinkedList container offered by java as expected in the assignment. For initializing courses in this list, I created a utility class to read csv files, called CSV_Utils. To implement required methods seen in use case diagram, I benefited from using a java LinkedList's methods.

2.3.1 Part 1 Methods & Time Complexities

```
public int listSemesterCourses(int semester){
   if (semester < 1 || semester > MAX_SEMS){
       System.out.println("There is no " + semester + ". semester in the 
program.");
       return 0;
   }
   else{

System.out.println("*************************************************************
************************************************");
       System.out.println(semester + ". Semester Courses");
       for(BasicCourse item : courses){
           if(Integer.toString(semester).equals(item.getSemester()))
               System.out.println(item);
       }

System.out.println("*************************************************************
************************************************");
       return 1;
   }
```

listSemesters method: If the inputted semester is a valid semester in the catalog, this code runs at Θ (n) complexity. We acquire this by inspecting the main loop of the code. It iterates through whole catalog which has size of n (number of courses). Prints if the wanted value is found and continuous until the end no matter what. So, we can say that the code always will run at Θ(n) complexity, assuming all other comparison and printing operations are constant time.

```
public int getByRange(int startIndex, int lastIndex){
    if(startIndex > lastIndex){
        System.out.println("Starting semester value cannot be bigger than ending 
value.");
        return 0;
    }

    else if(startIndex <= 0 || lastIndex > MAX_SEMS){
        System.out.println("Semesters out of range.");
        return 0;
    }
    else{
        System.out.println("Courses between " + startIndex + ". and " + 
lastIndex + ". semesters.");
        for(int i = startIndex; i <= lastIndex; i++)
            listSemesterCourses(i);

        return 1;
    }
```

getByRange method: This code is also very similar to listSemester method. Only difference is that it prints given range of semesters. To do this operation it uses listSemester method for range of semester user inputted (Let's say k semesters). We confirmed that listSemester's complexity is Θ(n), and it will be called k times, so we have k*Θ(n) which normalizes to Θ(n).

```java
public int getByCode(String code){
    int found = 0;

    for(BasicCourse item : courses){
        if(code.equals(item.getCourseCode())){
            System.out.println(item);
            found = 1;
        }
    }

    if(found == 0)
        System.out.println("No courses found with that code.");

    return found;
```

getByCode method: It's easy to see that this method runs at Θ(n) also. I will mention this codes in a later section as this type of methods are frequent.

```java
public BasicCourse get(String code){
    if(code.equals("XXX XXX"))
        throw new IllegalArgumentException("Optional courses are only available
to edit by indexes.");

    for(BasicCourse item : courses){
        if(code.equals(item.getCourseCode())){
            return item;
        }
    }
    throw new NoSuchElementException("Item not found. Code : " + code);
}

public BasicCourse get(int courseIndex){
    return courses.get(courseIndex);
}
```

Bonus get methods: There are also two get methods I implemented for ease of usage. These methods retrieve a course either by course code or index. They perform a linear search. So in best case they work at Ω(1), in worst case O(n). The one that uses index to access benefits from java linked list's get method. Assuming it's linear time to.

2.3.2 Part 2 Methods & Time Complexities

```java
public int enable(String courseCode){
    BasicCourse item = advancedCMS.get(courseCode);
    return enableWrap(item);
}
public int enable(int courseIndex){
    BasicCourse item = advancedCMS.get(courseIndex);
    return enableWrap(item);
}

private int enableWrap(BasicCourse item) {
    if(item.isAccessable() == true){
        System.out.println("Item is already enabled.");
        return 0;
    }
    else{
        item.setAccessible(true);
        size++;
        return 1;
    }
}
```

Enable and Disable methods: I will go over only enable method, because enable and disable have same operations but different assignments. There are two types of enable, one works with course index, other with the course code. They both use get operation which we mention to run at O(n). After the get operation all other operations are constant time assignments and comparisons. So, it's easy to say that these methods run at O(n) + c which normalizes to O(n).

```java
public int showDisabled(){
    int numOfDisabledCourses = 0;
    for(BasicCourse item : advancedCMS.getCourses()){
        if(item.isAccessable() == false){
            System.out.println((numOfDisabledCourses+1) + ")  " + item);
            numOfDisabledCourses++;
        }
    }
    if(numOfDisabledCourses == 0)
        System.out.println("There are no disabled courses");

    return numOfDisabledCourses;
}
```

showDisabled method: Like getByRange method, goes over whole list no matter what to find disabled items. If an item is found, it's printed and loop continuous. It runs at O(n).

set, remove and listIterator methods: These methods all perform linear searches to access the given item. Then perform constant time operations. So, they all work at O(n).

2.3.3 Notes on Time Complexities

As we can see most of the functions runs at linear complexity. We could reduce this by using sorted list. But in this case, it's better put our resources to generalize the code to work on all kinds of catalogs. Also, easier to implement. This way of thinking doesn't cost us much thanks to small number of data to manipulate.

2.3.4 Missing parts

I believe that I completed all the required operations for part 1 and part 2, but I didn't complete part 3 due to lack of time and bad design. At first, I tried to implement my own double linked list. The plan was to hold courses in this linked list. Then to link the courses in the same semester, I planned to extend the first linked list to create a circular linked list class. Implement iterators for both linked list types so I can complete the task in part 3 that required next and nextInSemester methods. I planned my time incorrectly, spent too much time on part 1 and part 2. Part 3 required more implementation both combined, in my design at least. So, I couldn't put something good together, it's little bit of frankenstein code in sense that I was trying to implement and design at the same time. So, I scrapped most of it, if there are any stupid parts of code remaining on part3CMS class, please ignore.

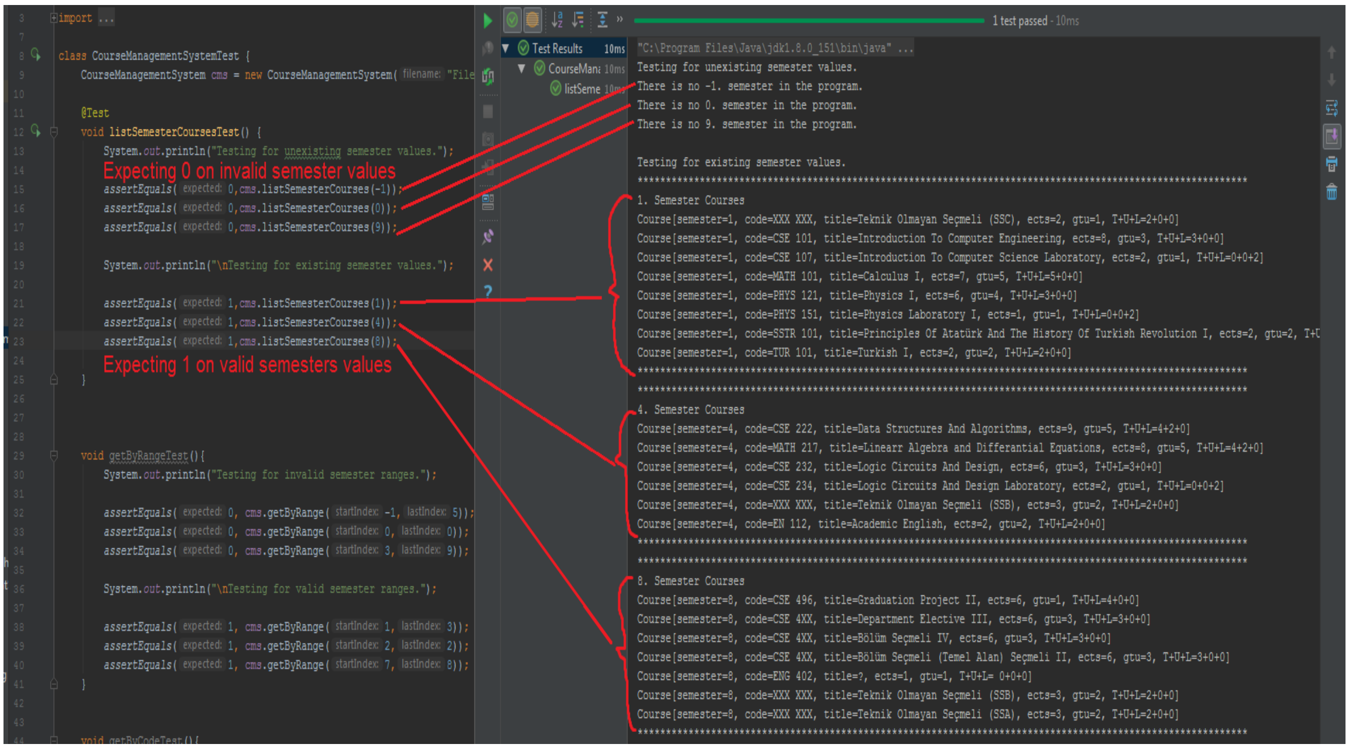
3 RESULTS

3.1 Test Cases

All main tests are printed in a text file, located in the Test Results folder. Some of the better unit test are also stored as html files in this folder. They can be expanded to view the details of the results. Here, I'll mention some of the main tests and unit tests. These test cases are grouped my scenarios under assignment parts. In the test, I tried to test all possible return values for methods to see if the code segments stored under if-else statements execute correctly.

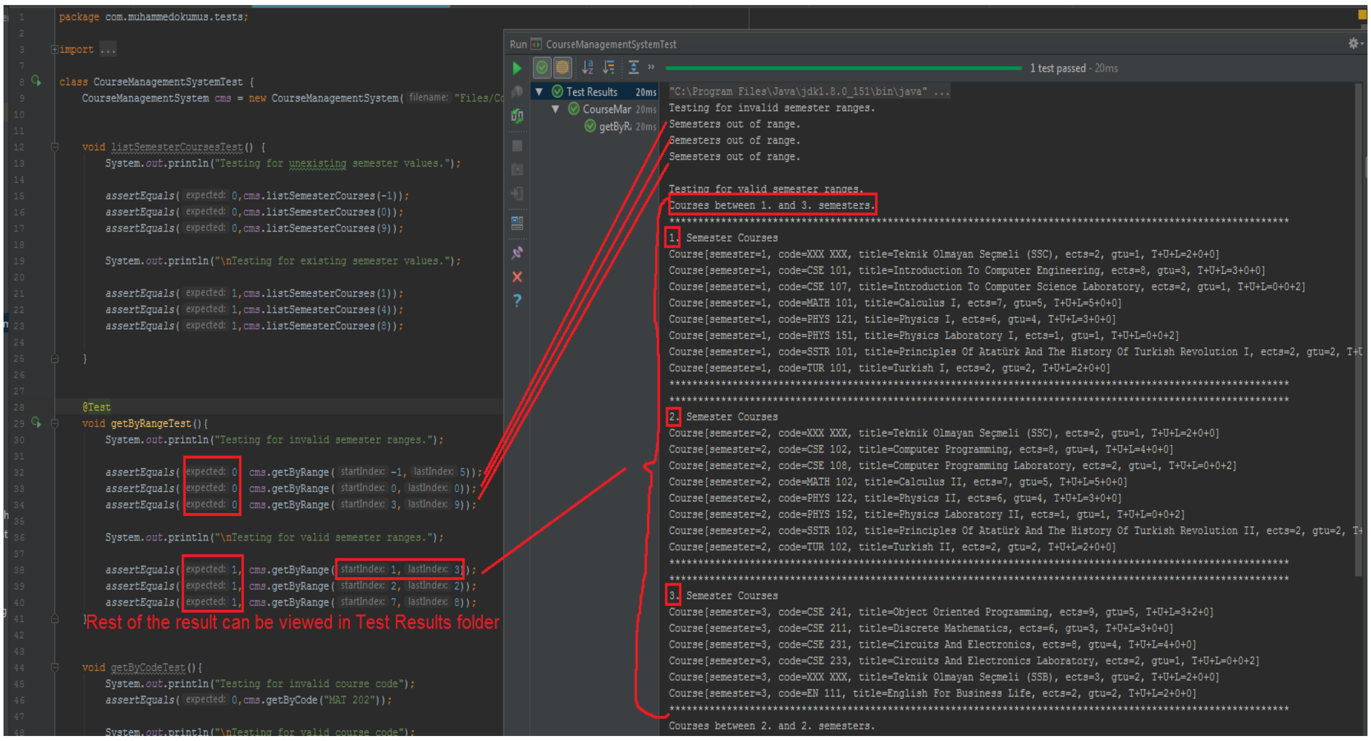
3.2 Running Results

3.2.1 Course Management System Unit Tests


Expecting 0 on invalid semester values
Expecting 1 on valid semesters values
1 test passed - 10ms
Test Results 10ms
CourseMan: 10ms
listSeme 10ms

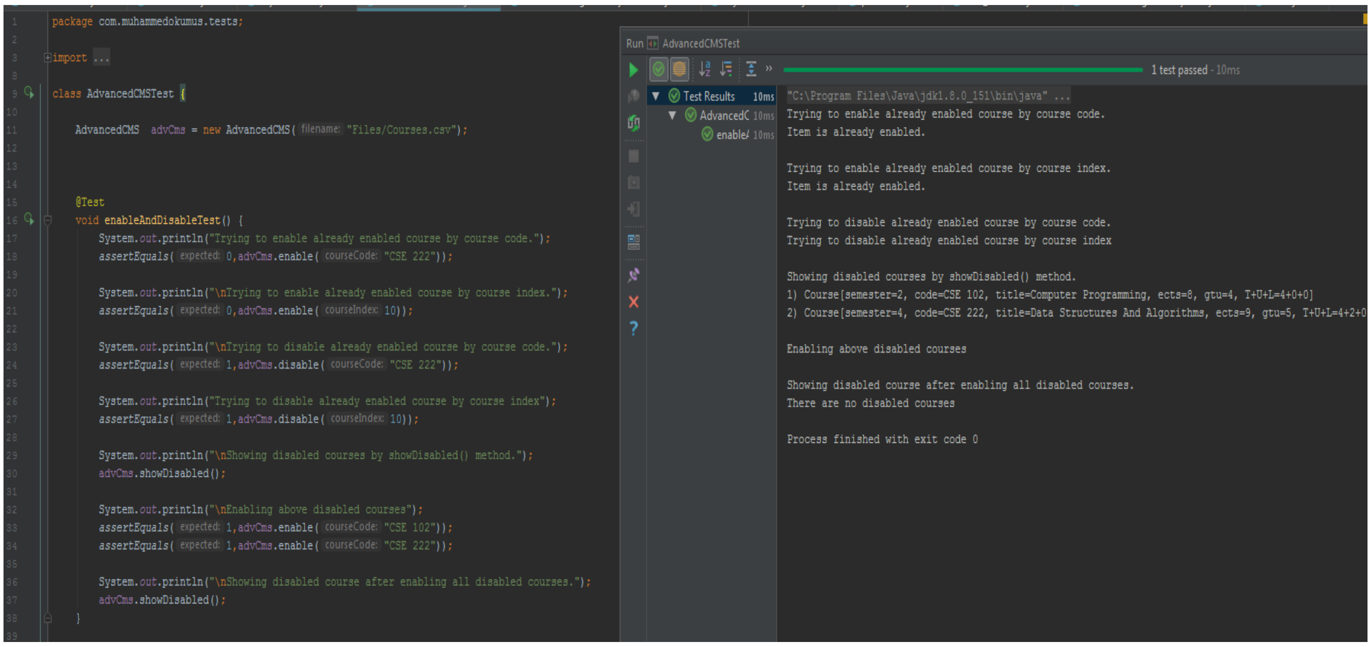
```java
import ...

class CourseManagementSystemTest {
    CourseManagementSystem cms = new CourseManagementSystem( filename: "File

    @Test
    void listSemesterCoursesTest() {
        System.out.println("Testing for unexisting semester values.");

        assertEquals( expected: 0,cms.listSemesterCourses(-1));
        assertEquals( expected: 0,cms.listSemesterCourses(0));
        assertEquals( expected: 0,cms.listSemesterCourses(9));

        System.out.println("\nTesting for existing semester values.");

        assertEquals( expected: 1,cms.listSemesterCourses(1));
        assertEquals( expected: 1,cms.listSemesterCourses(4));
        assertEquals( expected: 1,cms.listSemesterCourses(8));

    }



    void getByRangeTest(){
        System.out.println("Testing for invalid semester ranges.");

        assertEquals( expected: 0, cms.getByRange( startIndex: -1, lastIndex: 5));
        assertEquals( expected: 0, cms.getByRange( startIndex: 0, lastIndex: 0));
        assertEquals( expected: 0, cms.getByRange( startIndex: 3, lastIndex: 9));

        System.out.println("\nTesting for valid semester ranges.");

        assertEquals( expected: 1, cms.getByRange( startIndex: 1, lastIndex: 3));
        assertEquals( expected: 1, cms.getByRange( startIndex: 2, lastIndex: 2));
        assertEquals( expected: 1, cms.getByRange( startIndex: 7, lastIndex: 8));
    }


    void getByCodeTest(){
```

```
"C:\Program Files\Java\jdk1.8.0_151\bin\java" ...
Testing for unexisting semester values.
There is no -1. semester in the program.
There is no 0. semester in the program.
There is no 9. semester in the program.

Testing for existing semester values.
**********************************************************************************************
1. Semester Courses
Course[semester=1, code=XXX XXX, title=Teknik Olmayan Seçmeli (SSC), ects=2, gtu=1, T+U+L=2+0+0]
Course[semester=1, code=CSE 101, title=Introduction To Computer Engineering, ects=8, gtu=3, T+U+L=3+0+0]
Course[semester=1, code=CSE 107, title=Introduction To Computer Science Laboratory, ects=2, gtu=1, T+U+L=0+0+2]
Course[semester=1, code=MATH 101, title=Calculus I, ects=7, gtu=5, T+U+L=5+0+0]
Course[semester=1, code=PHYS 121, title=Physics I, ects=6, gtu=4, T+U+L=3+0+0]
Course[semester=1, code=PHYS 151, title=Physics Laboratory I, ects=1, gtu=1, T+U+L=0+0+2]
Course[semester=1, code=SSTR 101, title=Principles Of Atatürk And The History Of Turkish Revolution I, ects=2, gtu=2, T+U
Course[semester=1, code=TUR 101, title=Turkish I, ects=2, gtu=2, T+U+L=2+0+0]
**********************************************************************************************
**********************************************************************************************
4. Semester Courses
Course[semester=4, code=CSE 222, title=Data Structures And Algorithms, ects=9, gtu=5, T+U+L=4+2+0]
Course[semester=4, code=MATH 217, title=Linearr Algebra and Differantial Equations, ects=8, gtu=5, T+U+L=4+2+0]
Course[semester=4, code=CSE 232, title=Logic Circuits And Design, ects=6, gtu=3, T+U+L=3+0+0]
Course[semester=4, code=CSE 234, title=Logic Circuits And Design Laboratory, ects=2, gtu=1, T+U+L=0+0+2]
Course[semester=4, code=XXX XXX, title=Teknik Olmayan Seçmeli (SSB), ects=3, gtu=2, T+U+L=2+0+0]
Course[semester=4, code=EN 112, title=Academic English, ects=2, gtu=2, T+U+L=2+0+0]
**********************************************************************************************
**********************************************************************************************
8. Semester Courses
Course[semester=8, code=CSE 496, title=Graduation Project II, ects=6, gtu=1, T+U+L=4+0+0]
Course[semester=8, code=CSE 4XX, title=Department Elective III, ects=6, gtu=3, T+U+L=3+0+0]
Course[semester=8, code=CSE 4XX, title=Bölüm Seçmeli IV, ects=6, gtu=3, T+U+L=3+0+0]
Course[semester=8, code=CSE 4XX, title=Bölüm Seçmeli (Temel Alan) Seçmeli II, ects=6, gtu=3, T+U+L=3+0+0]
Course[semester=8, code=ENG 402, title=?, ects=1, gtu=1, T+U+L= 0+0+0]
Course[semester=8, code=XXX XXX, title=Teknik Olmayan Seçmeli (SSB), ects=3, gtu=2, T+U+L=2+0+0]
Course[semester=8, code=XXX XXX, title=Teknik Olmayan Seçmeli (SSA), ects=3, gtu=2, T+U+L=2+0+0]
**********************************************************************************************
```


Rest of the result can be viewed in Test Results folder
Run CourseManagementSystemTest
1 test passed - 20ms
Test Results 20ms
CourseMar 20ms
getByR 20ms

```java
package com.muhammedokumus.tests;

import ...

class CourseManagementSystemTest {
    CourseManagementSystem cms = new CourseManagementSystem( filename: "Files/C

    void listSemesterCoursesTest() {
        System.out.println("Testing for unexisting semester values.");

        assertEquals( expected: 0,cms.listSemesterCourses(-1));
        assertEquals( expected: 0,cms.listSemesterCourses(0));
        assertEquals( expected: 0,cms.listSemesterCourses(9));

        System.out.println("\nTesting for existing semester values.");

        assertEquals( expected: 1,cms.listSemesterCourses(1));
        assertEquals( expected: 1,cms.listSemesterCourses(4));
        assertEquals( expected: 1,cms.listSemesterCourses(8));

    }


    @Test
    void getByRangeTest(){
        System.out.println("Testing for invalid semester ranges.");

        assertEquals( expected: 0, cms.getByRange( startIndex: -1, lastIndex: 5));
        assertEquals( expected: 0, cms.getByRange( startIndex: 0, lastIndex: 0));
        assertEquals( expected: 0, cms.getByRange( startIndex: 3, lastIndex: 9));

        System.out.println("\nTesting for valid semester ranges.");

        assertEquals( expected: 1, cms.getByRange( startIndex: 1, lastIndex: 3));
        assertEquals( expected: 1, cms.getByRange( startIndex: 2, lastIndex: 2));
        assertEquals( expected: 1, cms.getByRange( startIndex: 7, lastIndex: 8));
    }


    void getByCodeTest(){
        System.out.println("Testing for invalid course code");
        assertEquals( expected: 0,cms.getByCode("MAT 202"));

        System.out.println("\nTesting for valid course code");
```

```
"C:\Program Files\Java\jdk1.8.0_151\bin\java" ...
Testing for invalid semester ranges.
Semesters out of range.
Semesters out of range.
Semesters out of range.

Testing for valid semester ranges.
Courses between 1. and 3. semesters.
**********************************************************************************************
1. Semester Courses
Course[semester=1, code=XXX XXX, title=Teknik Olmayan Seçmeli (SSC), ects=2, gtu=1, T+U+L=2+0+0]
Course[semester=1, code=CSE 101, title=Introduction To Computer Engineering, ects=8, gtu=3, T+U+L=3+0+0]
Course[semester=1, code=CSE 107, title=Introduction To Computer Science Laboratory, ects=2, gtu=1, T+U+L=0+0+2]
Course[semester=1, code=MATH 101, title=Calculus I, ects=7, gtu=5, T+U+L=5+0+0]
Course[semester=1, code=PHYS 121, title=Physics I, ects=6, gtu=4, T+U+L=3+0+0]
Course[semester=1, code=PHYS 151, title=Physics Laboratory I, ects=1, gtu=1, T+U+L=0+0+2]
Course[semester=1, code=SSTR 101, title=Principles Of Atatürk And The History Of Turkish Revolution I, ects=2, gtu=2, T+U
Course[semester=1, code=TUR 101, title=Turkish I, ects=2, gtu=2, T+U+L=2+0+0]
**********************************************************************************************
**********************************************************************************************
2. Semester Courses
Course[semester=2, code=XXX XXX, title=Teknik Olmayan Seçmeli (SSC), ects=2, gtu=1, T+U+L=2+0+0]
Course[semester=2, code=CSE 102, title=Computer Programming, ects=8, gtu=4, T+U+L=4+0+0]
Course[semester=2, code=CSE 108, title=Computer Programming Laboratory, ects=2, gtu=1, T+U+L=0+0+2]
Course[semester=2, code=MATH 102, title=Calculus II, ects=7, gtu=5, T+U+L=5+0+0]
Course[semester=2, code=PHYS 122, title=Physics II, ects=6, gtu=4, T+U+L=3+0+0]
Course[semester=2, code=PHYS 152, title=Physics Laboratory II, ects=1, gtu=1, T+U+L=0+0+2]
Course[semester=2, code=SSTR 102, title=Principles Of Atatürk And The History Of Turkish Revolution II, ects=2, gtu=2, T+
Course[semester=2, code=TUR 102, title=Turkish II, ects=2, gtu=2, T+U+L=2+0+0]
**********************************************************************************************
**********************************************************************************************
3. Semester Courses
Course[semester=3, code=CSE 241, title=Object Oriented Programming, ects=9, gtu=5, T+U+L=3+2+0]
Course[semester=3, code=CSE 211, title=Discrete Mathematics, ects=6, gtu=3, T+U+L=3+0+0]
Course[semester=3, code=CSE 231, title=Circuits And Electronics, ects=8, gtu=4, T+U+L=4+0+0]
Course[semester=3, code=CSE 233, title=Circuits And Electronics Laboratory, ects=2, gtu=1, T+U+L=0+0+2]
Course[semester=3, code=XXX XXX, title=Teknik Olmayan Seçmeli (SSB), ects=3, gtu=2, T+U+L=2+0+0]
Course[semester=3, code=EN 111, title=English For Business Life, ects=2, gtu=2, T+U+L=2+0+0]
**********************************************************************************************
Courses between 2. and 2. semesters.
```



```java
package com.muhammedokumus.tests;

import ...

class AdvancedCMSTest {

    AdvancedCMS  advCms = new AdvancedCMS( filename: "Files/Courses.csv");


    @Test
    void enableAndDisableTest() {
        System.out.println("Trying to enable already enabled course by course code.");
        assertEquals( expected: 0,advCms.enable( courseCode: "CSE 222"));

        System.out.println("\nTrying to enable already enabled course by course index.");
        assertEquals( expected: 0,advCms.enable( courseIndex: 10));

        System.out.println("\nTrying to disable already enabled course by course code.");
        assertEquals( expected: 1,advCms.disable( courseCode: "CSE 222"));

        System.out.println("Trying to disable already enabled course by course index");
        assertEquals( expected: 1,advCms.disable( courseIndex: 10));

        System.out.println("\nShowing disabled courses by showDisabled() method.");
        advCms.showDisabled();

        System.out.println("\nEnabling above disabled courses");
        assertEquals( expected: 1,advCms.enable( courseCode: "CSE 102"));
        assertEquals( expected: 1,advCms.enable( courseCode: "CSE 222"));

        System.out.println("\nShowing disabled course after enabling all disabled courses.");
        advCms.showDisabled();
    }
```

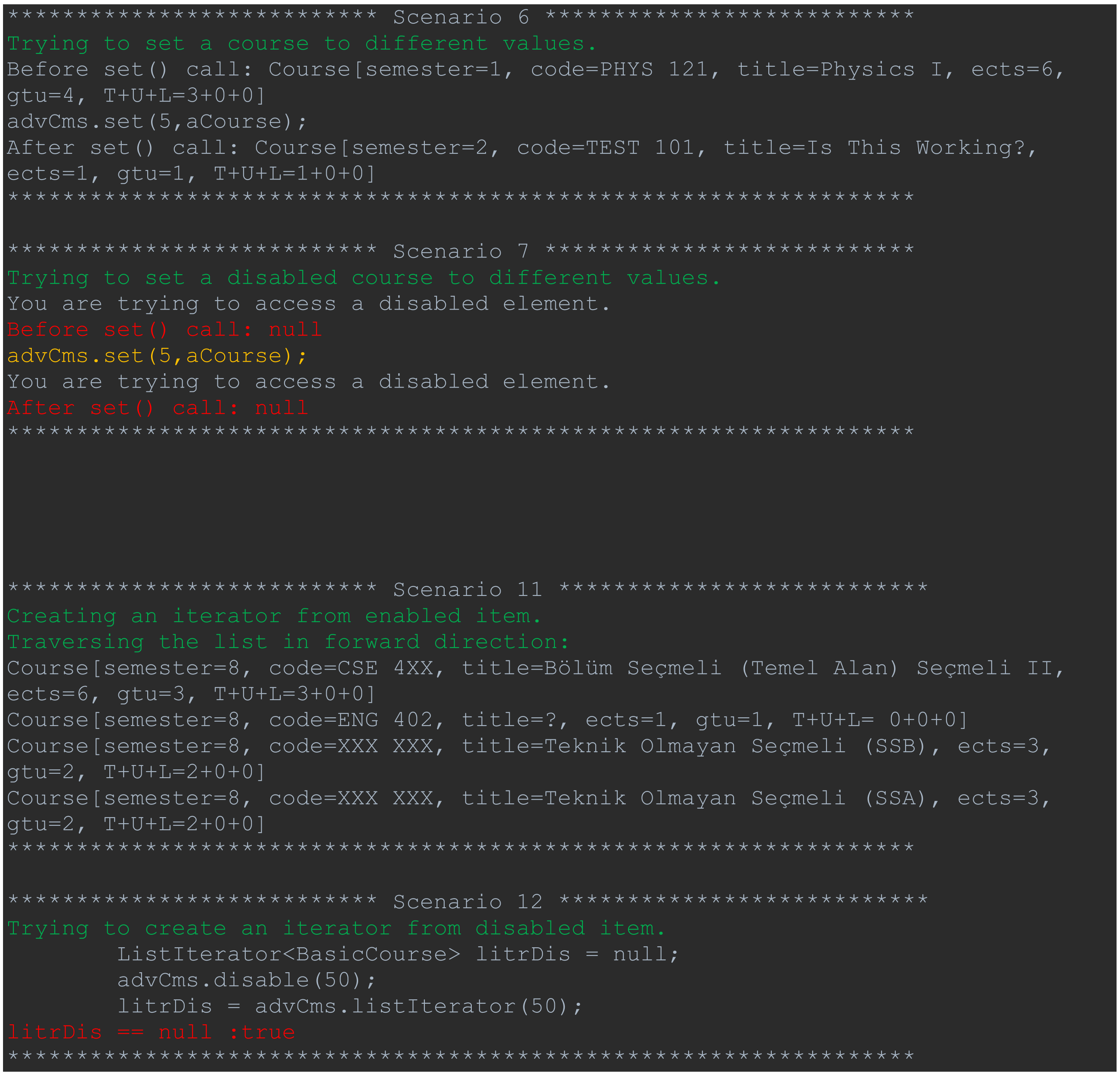
```
"C:\Program Files\Java\jdk1.8.0_151\bin\java" ...
Trying to enable already enabled course by course code.
Item is already enabled.

Trying to enable already enabled course by course index.
Item is already enabled.

Trying to disable already enabled course by course code.
Trying to disable already enabled course by course index

Showing disabled courses by showDisabled() method.
1) Course[semester=2, code=CSE 102, title=Computer Programming, ects=8, gtu=4, T+U+L=4+0+0]
2) Course[semester=4, code=CSE 222, title=Data Structures And Algorithms, ects=9, gtu=5, T+U+L=4+2+0

Enabling above disabled courses

Showing disabled course after enabling all disabled courses.
There are no disabled courses

Process finished with exit code 0
```

3.2.2 Course Management System Main Tests

```
*************************** Scenario 3 ***************************
Trying to access with optional course code XXX XXX.
java.lang.IllegalArgumentException: Optional courses are only available to edit
by indexes.
Didn't forward error to stderr for readability.
******************************************************************
*************************** Scenario 4 ***************************
Trying to access with index and manipulate.
Unedited: Course[semester=1, code=XXX XXX, title=Teknik Olmayan Seçmeli (SSC),
ects=2, gtu=1, T+U+L=2+0+0]
        test4.setCourseCode("255");
        test4.setCourseTitle("İş Hayatı İçin İnglizce");
        test4.setSemester("1");
        test4.setEctsCredit("4");
        test4.setGtuCredit("2");
        test4.setCreditWeight("2+1+1");
Edited: Course[semester=1, code=255, title=İş Hayatı İçin İnglizce, ects=4,
gtu=2, T+U+L=2+1+1]
******************************************************************
```

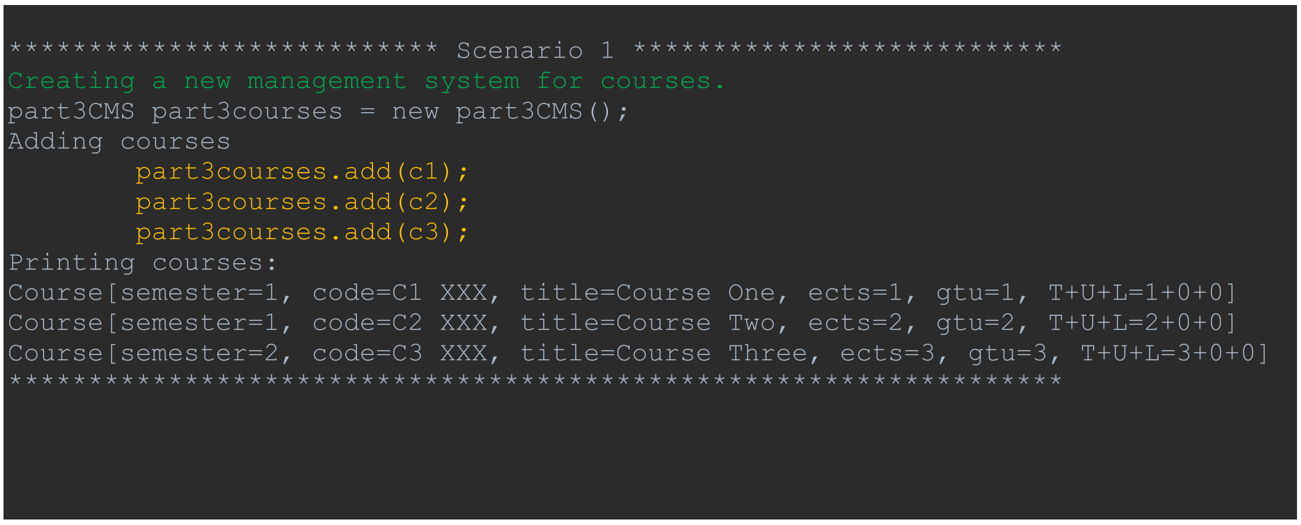
3.3 AdvancedCMS Main Tests

```
*************************** Scenario 5 ***************************
Disabling courses by code, then printing disabled courses with showDisabled()
method
        advCms.disable("CSE 232");
        advCms.disable("EN 112");
        advCms.disable("CSE 211");
        advCms.showDisabled();
1) Course[semester=3, code=CSE 211, title=Discrete Mathematics, ects=6, gtu=3,
T+U+L=3+0+0]
2) Course[semester=4, code=CSE 232, title=Logic Circuits And Design, ects=6,
gtu=3, T+U+L=3+0+0]
3) Course[semester=4, code=EN 112, title=Academic English, ects=2, gtu=2,
T+U+L=2+0+0]
******************************************************************
```

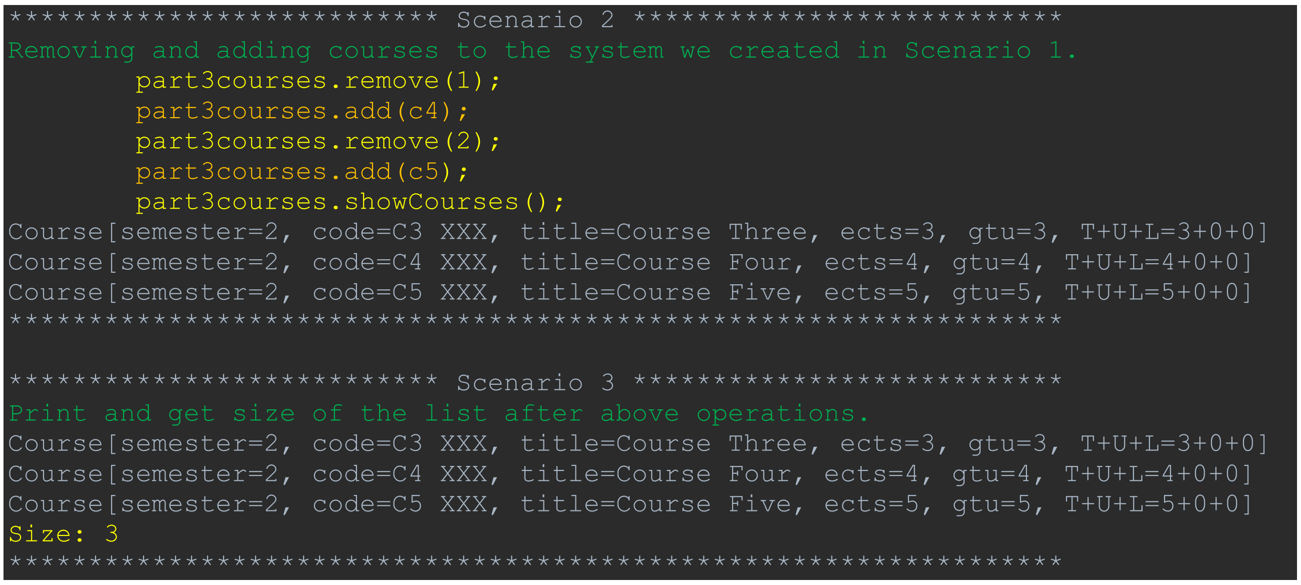


```
*************************** Scenario 6 ***************************
Trying to set a course to different values.
Before set() call: Course[semester=1, code=PHYS 121, title=Physics I, ects=6,
gtu=4, T+U+L=3+0+0]
advCms.set(5,aCourse);
After set() call: Course[semester=2, code=TEST 101, title=Is This Working?,
ects=1, gtu=1, T+U+L=1+0+0]
******************************************************************

*************************** Scenario 7 ***************************
Trying to set a disabled course to different values.
You are trying to access a disabled element.
Before set() call: null
advCms.set(5,aCourse);
You are trying to access a disabled element.
After set() call: null
******************************************************************




*************************** Scenario 11 ***************************
Creating an iterator from enabled item.
Traversing the list in forward direction:
Course[semester=8, code=CSE 4XX, title=Bölüm Seçmeli (Temel Alan) Seçmeli II,
ects=6, gtu=3, T+U+L=3+0+0]
Course[semester=8, code=ENG 402, title=?, ects=1, gtu=1, T+U+L= 0+0+0]
Course[semester=8, code=XXX XXX, title=Teknik Olmayan Seçmeli (SSB), ects=3,
gtu=2, T+U+L=2+0+0]
Course[semester=8, code=XXX XXX, title=Teknik Olmayan Seçmeli (SSA), ects=3,
gtu=2, T+U+L=2+0+0]
******************************************************************

*************************** Scenario 12 ***************************
Trying to create an iterator from disabled item.
        ListIterator<BasicCourse> litrDis = null;
        advCms.disable(50);
        litrDis = advCms.listIterator(50);
litrDis == null :true
******************************************************************
```

3.4 Part 3 Main Tests



```
*************************** Scenario 1 ***************************
Creating a new management system for courses.
part3CMS part3courses = new part3CMS();
Adding courses
        part3courses.add(c1);
        part3courses.add(c2);
        part3courses.add(c3);
Printing courses:
Course[semester=1, code=C1 XXX, title=Course One, ects=1, gtu=1, T+U+L=1+0+0]
Course[semester=1, code=C2 XXX, title=Course Two, ects=2, gtu=2, T+U+L=2+0+0]
Course[semester=2, code=C3 XXX, title=Course Three, ects=3, gtu=3, T+U+L=3+0+0]
******************************************************************
```



```
*************************** Scenario 2 ***************************
Removing and adding courses to the system we created in Scenario 1.
        part3courses.remove(1);
        part3courses.add(c4);
        part3courses.remove(2);
        part3courses.add(c5);
        part3courses.showCourses();
Course[semester=2, code=C3 XXX, title=Course Three, ects=3, gtu=3, T+U+L=3+0+0]
Course[semester=2, code=C4 XXX, title=Course Four, ects=4, gtu=4, T+U+L=4+0+0]
Course[semester=2, code=C5 XXX, title=Course Five, ects=5, gtu=5, T+U+L=5+0+0]
******************************************************************

*************************** Scenario 3 ***************************
Print and get size of the list after above operations.
Course[semester=2, code=C3 XXX, title=Course Three, ects=3, gtu=3, T+U+L=3+0+0]
Course[semester=2, code=C4 XXX, title=Course Four, ects=4, gtu=4, T+U+L=4+0+0]
Course[semester=2, code=C5 XXX, title=Course Five, ects=5, gtu=5, T+U+L=5+0+0]
Size: 3
******************************************************************
```

Full versions of this test can be viewed in test result file.