

**Gebze Technical University
Computer Engineering**

CSE 222 - 2018 Spring

HOMEWORK 3 REPORT

**BURAK DEMİRCİ
141044091**

1 INTRODUCTION

1.1 Problem Definition

Part1: Bu bölümde GTU Bilgisayar Mühendisliği derslerinden oluşan node'ları Java Linked List class yapısı kullanarak aşağıdaki metodları gerçekleştirmemiz istenilmiştir. Linked List classı extend edilmeyecektir şartı getirilerek. Bu yapıyı değişken(variable) yapısında kullanılmamız hedeflenmiştir.

- `getByCode(String code)` : Verilen koddaki dersleri döndürecek
- `listSemesterCourses (int semester)`: Verilen dönemdeki dersleri döndürecek
- `getByRange(int start_index, int last_index)`: Verilen aralıktaki dersleri döndürecek

Part2: Bu bölümde Java Linked List classını extends ederek kendi özel Linked List classımızı oluşturmamız isteniyor bu Linked List Generic bütün objeler için çalışabilmelidir. Bu Linked Listin özelliği ise bu listin elemanlarını `enable(int index)` ve `disable(int index)` özelliklerinin olmasıdır bu `disable(int index)` özelliği bu listin indexteki elemanın bazı metodları kullanıma kapatıyor `get`, `set`, `size`, `remove` ve `listIterator` methodları

Part3: Bu bölümde ise Java Linked List yapısında fakat kendi içinde semester lar circular olacak şekilde bir yapı tanımlamamız isteniyor. Bu yapı için Java Linked List classının objesi kullanılmayacak extends edilmeyecektir.

- `add()`: Liste yeni eleman ekler
- `remove()`: Listten eleman siler
- `next()`: bir sonraki elemana gide
- `nextInSemester()`: Aynı semester(dönem) de bulunan bir sonraki derse gider
- `size()`: listenin büyüklüğünü döndürür

1.2 System Requirements

Course.csv aşağıdaki formatta olmalıdır

Semester; Course Code; Course Title; ECTS Credits; GTU Credits; H+T+L

Örnek: 4; CSE 222; Data Structures And Algorithms; 9; 5; 4+2+0

Part1: Bu bölümde kullanıcı sisteme Course.csv bütün derslere ait bilgiler bulunan dosyayı vermelidir bütün bilgiler bu dosyadan okunarak Course objeleri şeklinde Linked List yapısında tutulur. Kullanıcı `getByCode(String code)` metoduna istediği kursun ismini girerek arama yapıp bu aramaların listesini alır yanlış eksik bir bilgi girdiğinde ise hata (Exception) fırlatır. `listSemesterCourses (int semester)` metodunu istediği dönemin bütün derslerini listeleyerek ulaşabilir burada da yanlış bilgi girdiğinde hata (Exception) fırlatır. `getByRange(int start_index, int last_index)` bu yöntemde ise kullanıcı listede istediği aralıktaki derslerin listesini alabilir.

Part2: Bu bölümde kullanıcı sisteme Course.csv bütün derslere ait bilgiler bulunan dosyayı vermelidir bütün bilgiler bu dosyadan okunarak Course objeleri şeklinde MyLinkedList yapısında tutulur. Bu listin en temel özelliği içerisindeki elemanların belli metodlarını kullanıma kilitleyip açabilmektir. Kullanıcı `disable(int index)` metodu ile verilen listteki konumundaki (indexteki) elemanın bu metodlarını kullanıma kapatabilir. `enable(int index)` metodu ile verilen konumdaki elemanın bu özellikleri eğer kapalıysa kullanıma açar.

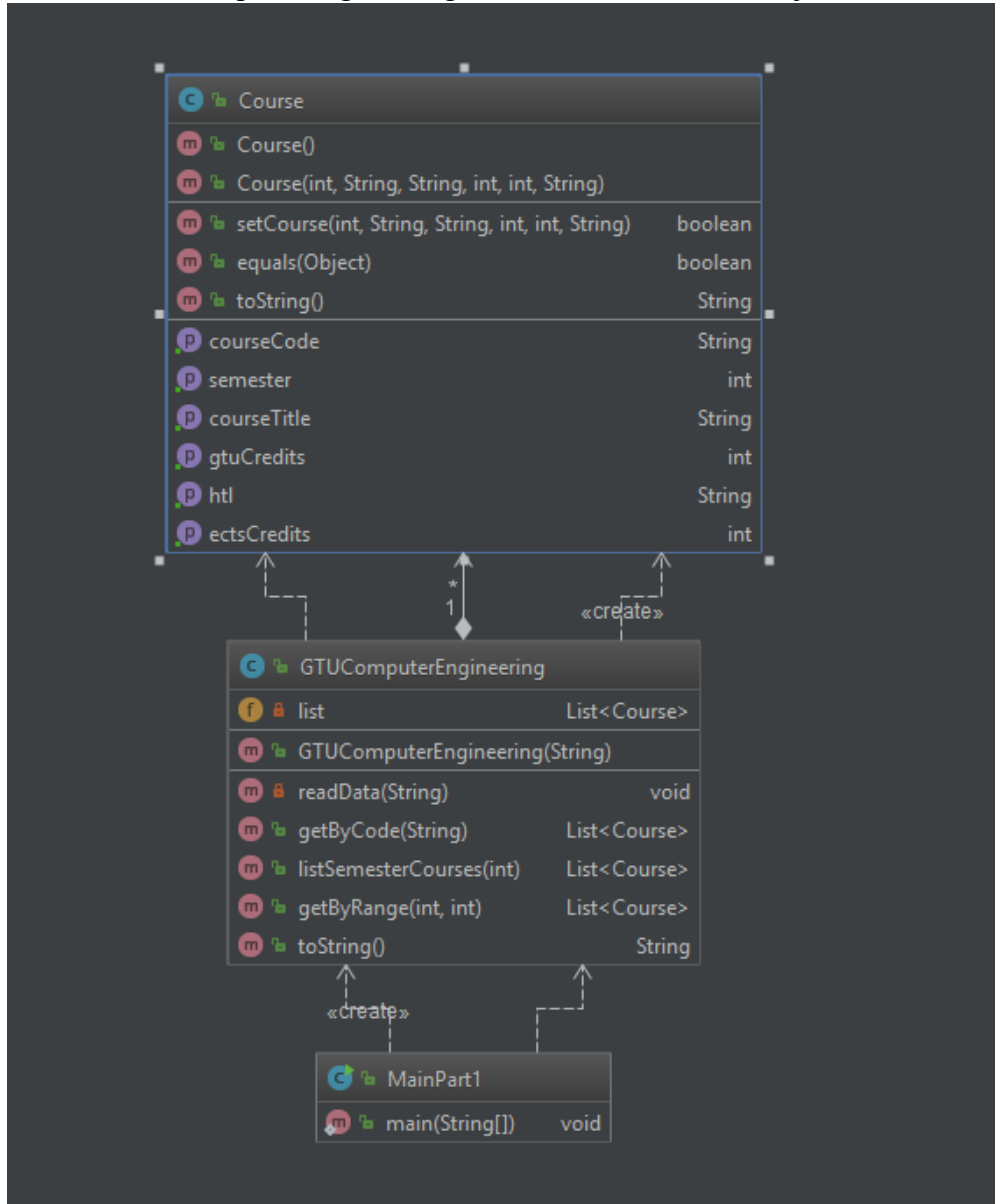
Part3: Bu bölümde kullanıcı sisteme Course.csv bütün derslere ait bilgiler bulunan dosyayı vermelidir bütün bilgiler bu dosyadan okunarak Course objeleri şeklinde CourseList yapısında tutulur bu sistem LinkedList yapısına benzer bir yapı doğrultusunda oluşturulmuştur. LinkedList yapısına benzeyen bu sistemin LinkedListten ayrılan yanı ise içerisinde tuttuğu Course objesinin semester bilgisinden yararlanılarak aynı semester olan yapılar CircularLinked list gibi davranacaktır. Diğerleri arasında (farklı semestерler) ise Linked List gibi bir sistemle bağlı olacaktır.

Bunu sağlamak için CourseIterotor inner classı Iterator interfacesinden implement edilerek kullanılmıştır normal iteratore ek olarak nextInSemester(int semester) alan bir metodu bulunmaktadır.

2 METHOD

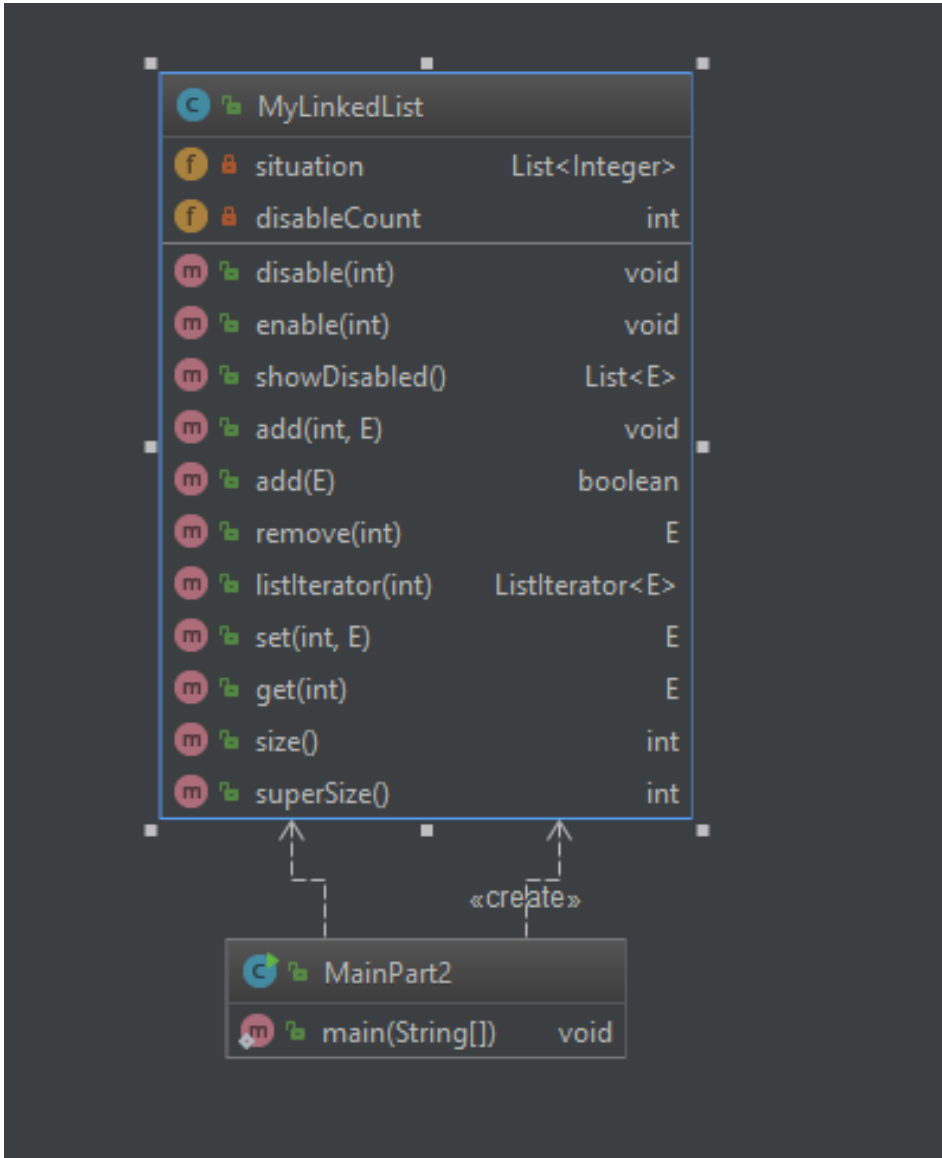
2.1 Class Diagrams

Part1: GTUComputerEngineering classındaki list Course objesini node olarak kullanıyor.



Part2:

MyLinkedList classı LinkedList'ti extends ederek aşağıdaki metodlar override edilmiştir



Part3:

CourseList Classı Inner Class olarak aşağıdaki classları kullanır v e aşağıdaki metodları bulundurur.

- `add()`: Liste yeni eleman ekler
- `remove(int index)`: Listten verilen indexteki eleman siler
- `size()`: listenin büyüklüğünü döndürür

Node classı:

Bu listenin elemanlarının tutulduğu sistem Course , semester ve next Node yapıları bulunur.

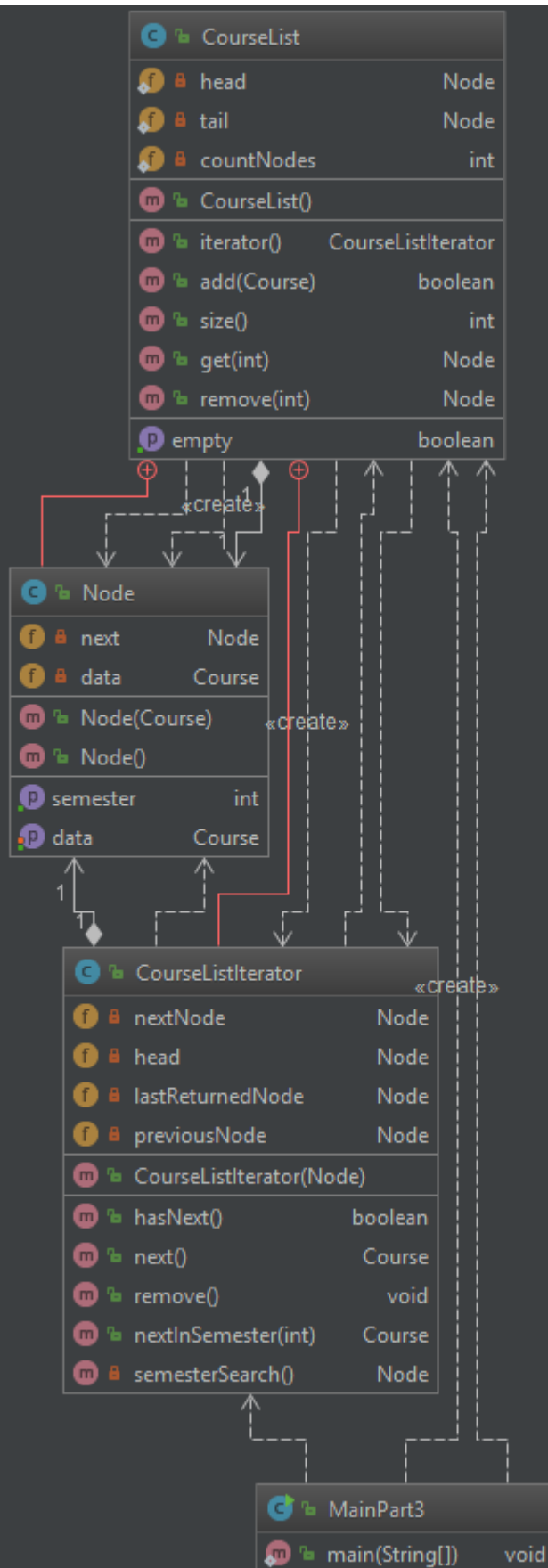
Ve bunlara ait get set metodları ve remove metodu bulunmaktadır.

Iterator classı :

`next()`: bir sonraki elemana gide

`nextInSemester()`: Aynı semester(dönem) de bulunan bir sonraki derse gider

`remove()`: Listten eleman siler



2.2 Problem Solution Approach

Part1: Bu bölümün çözümünde Java Linked List classının extends edilmeden sadece objesi oluşturularak kullanılıp,

- getByCode(String code) : Verilen koddaki dersleri döndürecek
- listSemesterCourses (int semester): Verilen dönemdeki dersleri döndürecek
- getByRange(int start_index, int last_index): Verilen aralıktaki dersleri döndürecek

Yukarıdaki metodların yazılması beklenilmiştir.Çözüm için öncelikle Course derslerin bilgilerini tutabilecek bir class olan Course classı yazıldı bu class derse ait bilgileri ve bunların get() ,set() metodlarını bulunduruyor. GTUComputerEngineering classında ise bu Course objesi her ders için oluşturulup Linked List yapısında bu objeler tutuluyor . Bu listeden yararlanılarak istenilen metodlarda kullanıldı. İstenilen metodlarda parametre bilgisinin yanlış girilmesi durumunda Exception fırlatılmıştır. GTUComputerEngineering classının objesi oluşturulurken okuyacağı Course.csv dosyasının konumu verilerek oluşturulur. Obje oluşurken bu dosyadan bilgileri okuyarak her bir dersi bir Course objesi olarak LinkedList'e ekler. Yukarıdaki metodlar ise Linked Listi kullanarak gerekli işlemleri yerine getirir.

Part2: Bu bölümde oluşturulması istenilen yapı Java LinkedList yapısında bir sistemdir bu sisteme ek olarak bazı özelliklerin eklenilmesi istenilmiştir. Bu özellik Listenin istenilen elemanın get(), set(int index, Object eleman), size(), remove(int index) ve listIterator(int index) bu metodlarını kullanıma kapatıp (disable(int index)) veya kullanıma kapatılan elemanlar tekrardan kullanıma açılabilir (enable(int index)) engelli (disable) olan elemanları görebilmek için showDisable() metodu oluşturulmuştur bu metod disable olan elemanları yeni bir listeye atarak döndürür.Bu sistemde LinkedList yapısına benzer bir yapı olduğu için LinkedList classı extends edilerek kullanılmıştır.Linked Listin aşağıdaki metodları override edilerek kullanılmıştır.

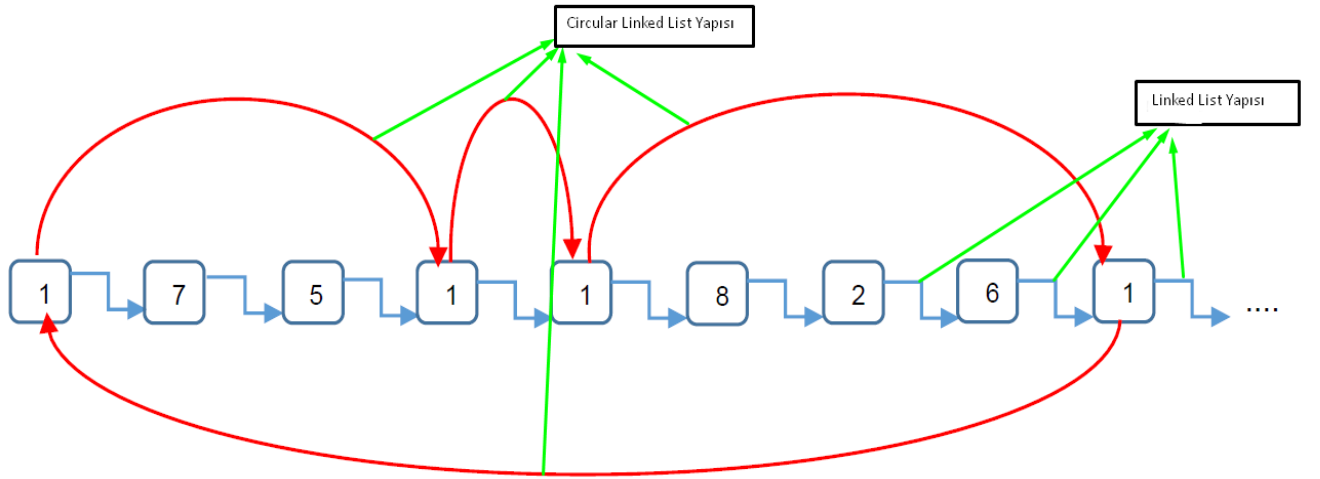
- add(E obj)
- add(int index, E obj)
- remove(int index)
- listIterator(int index)
- set(int index)
- get(int index)

Aşağıdaki resimde tasvir edildiği gibi her eklenen objeye karşılık içeride yeni bir liste şeklinde o objenin durumunu tutan bir bilgi eklenir paralel array sisteminde objenin durumu kolayca takip edilebilir. Bunun avantajı objenin konumunu değiştirmeden durum bilgisini değiştirebilmek obje üzerinde daha az işlem yapmak ve bunun gibi kolaylıklar sağlar.

	CourseList		Situation	
index 0	Node	----->	1	
index 1	Node	----->	1	
index 2	Node	----->	1	
index 3	Node	----->	0	1 -> Enable
index 4	Node	----->	0	
index 5	Node	----->	1	0 -> Disable
index 6	Node	----->	1	
index 7	Node	----->	1	
index 8	Node	----->	1	
index 9	Node	----->	0	

Part3: Bu bölümde Java Linked List yapısında bir sistem tasarlamamız istenilmiştir bu tasarım yapılırken Java Linked List classı extends edilmeyecek ve objesi kullanılmayacaktır bu şartlar nedeni ile kendi CourseList classımızı baştan kodlamamız istenilmiştir. Bu class sistemi elemanları normal LinkedList gibi birbirine bağlı olarak ilerleyecektir fakat nextInSemester(int semester) Iteratoru kullanıldığında aynı dönemde olan dersler CircularLinked list gibi davranıp sadece aynı dönem olan dersler üzerinde dolaşılacaktır. Yapının tasarlanması için LinkedListin temel özelliği olan Node ve Iterator inner classları belirlenmiştir. Node inner classında data olarak Course objesi kullanılmıştır. Iterator classında ise remove(),next() ve hasNext() metodlarına ek olarak nextInSemester(int semester) metodu tanımlanmıştır.

- add(Course cs): Liste yeni eleman(Course) ekler
- remove(int index) , remove() : Listten eleman siler
- next(): bir sonraki elemana gide
- nextInSemester(int semester): Aynı semester(dönem) de bulunan bir sonraki derse gider ve liste bitince tekrar başa dönerek devam eder.
- size(): listenin enable olan eleman sayısını döndürür
- superSize() : listenin bütün elemanlarının sayısını döndürür



3 RESULT

3.1 Test Cases

Part1: Aşağıda Part1 JUnit testleri verilmiştir

Aşağıdaki resimde Course Classının bazı metodları JUnit testle sınanmıştır

The screenshot displays an IDE with three tabs: `GTUComputerEngineering.java`, `CourseTest.java`, and `GTUComputerEngineeringTest.java`. The `CourseTest.java` tab is active, showing the following code:

```
1 package part1;
2
3 import ...
4
5 class CourseTest {
6
7     private Course course = new Course( semester: 1, courseCode: "CSE101", courseTitle: "Programlama", ec
8
9     @Test
10     void setCourse() { assertEquals( expected: true, course.setCourse( semester: 1, courseCode: "CSE101"
11
12     @Test
13     void getSemester() { assertEquals( expected: 1, course.getSemester()); }
14
15     @Test
16     void getCourseCode() { assertEquals( expected: "CSE101", course.getCourseCode()); }
17
18     @Test
19     void getEctsCredits() { assertEquals( expected: 6, course.getEctsCredits()); }
20
21     @Test
22     void getGtuCredits() { assertEquals( expected: 8, course.getGtuCredits()); }
23
24     @Test
25     void getCourseTitle() { assertEquals( expected: "Programlama", course.getCourseTitle()); }
26
27     @Test
28     void getHtl() { assertEquals( expected: "2+0+1", course.getHtl()); }
29
30     @Test
31     void equals() { assertEquals( expected: false, course.equals(new Course())); }
32
33 }
```

The bottom of the image shows the `Run` tab for `CourseTest`, indicating that all tests passed. The `Test Results` table is as follows:

Test Case	Duration
CourseTest	36ms
getGtuCredits()	16ms
getCourseCode()	16ms
getEctsCredits()	16ms
equals()	16ms
getHtl()	2ms
getSemester()	1ms
getCourseTitle()	1ms
setCourse()	1ms

The status bar at the bottom right indicates "All 8 tests passed".

Aşağıdaki resimde ise GTUComputerEngineering classının metodları JUnit test ile sınanmıştır. Testler başarı ile sonuçlanmıştır.

The image shows an IDE window with three tabs: `GTUComputerEngineering.java`, `CourseTest.java`, and `GTUComputerEngineeringTest.java`. The `GTUComputerEngineeringTest.java` tab is active, displaying the following code:

```
1 package part1;
2
3 import ...
4
5
6
7
8
9
10
11 class GTUComputerEngineeringTest {
12
13     private GTUComputerEngineering eng = new GTUComputerEngineering( "src/Courses.csv");
14     private Course course = new Course( semester: 1, courseCode: "CSE 107", courseTitle: "Introduction To C
15
16     @Test
17     void getByCode() throws Exception {
18         assertEquals(course,eng.getByCode("CSE 107").get(0));
19     }
20
21     @Test
22     void listSemesterCourses() throws Exception {
23         assertEquals( expected: 8,eng.listSemesterCourses(1).size());
24     }
25
26     @Test
27     void getByRange() throws Exception {
28         assertEquals(course,eng.getByRange( start_index: 2, last_index: 2).get(0));
29     }
30 }
```

The left sidebar shows the project structure for `HW_3`, including `src` and `TESTS` directories. The `TESTS` directory contains `part1`, `part2`, and `part3` subdirectories, each with its own test classes. The `part1` directory is expanded, showing `CourseTest` and `GTUComputerEngineeringTest`.

At the bottom, the `Run` tab shows the execution of `GTUComputerEngineeringTest`. The test results are as follows:

Test Results	Time	Message
GTUComputerEngineeringTest	16ms	"C:\Program Files\Java\jdk1.8.0_144\bin\java" ...
listSemesterCourses()	16ms	Process finished with exit code 0
getByCode()		
getByRange()		

The status bar at the bottom right indicates "All 3 tests passed -".

Part2: Aşağıda Part2 JUnit testleri verilmiştir.

The screenshot displays an IDE with two tabs: `GTUComputerEngineering.java` and `MyLinkedListTest.java`. The `MyLinkedListTest.java` tab is active, showing the following code:

```
19      readData();
20    }
21
22    private void readData() {...}
23
24    @Test
25    void showDisabled() {
26        list.disable( index: 1);
27        assertEquals( expected: 1,list.showDisabled().size());
28    }
29
30    @Test
31    void add() {
32        MyLinkedList<Course> n=new MyLinkedList<>();
33        assertEquals( expected: true,n.add(new Course()));
34    }
35
36    @Test
37    void remove() {
38        MyLinkedList<Course> n=new MyLinkedList<>();
39        n.add(new Course());
40        assertEquals(new Course(),n.remove( index: 0 ));
41    }
42
43    @Test
44    void listIterator() {
45        assertEquals(course,list.listIterator( index: 2).next());
46    }
47
48    @Test
49    void get() {
50        assertEquals(course,list.get(2));
51    }
52
53    @Test
54    void set(){
55        MyLinkedList n = new MyLinkedList();
56        n.add(course);
57        assertEquals(course,n.set(0, new Course()));
58    }
59
```

The left sidebar shows the project structure for `HW_3`, including `src` and `TESTS` directories. The `TESTS` directory contains `part1`, `part2`, and `part3` subdirectories. The `part2` subdirectory contains `MyLinkedListTest`.

The bottom panel shows the `Run` tab for `MyLinkedListTest`. It indicates that all 6 tests passed. The test results are as follows:

Test Name	Duration
MyLinkedListTest	15ms
listIterator()	15ms
remove()	
showDisabled()	
add()	
get()	
set()	

The output window shows the command: `"C:\Program Files\Java\jdk1.8.0_144\bin\java" ...` and the message: `Process finished with exit code 0`.

Part3: Aşağıda Part3 JUnit testleri verilmiştir.

The screenshot displays an IDE with two tabs: `GTUComputerEngineering.java` and `CourseListTest.java`. The `CourseListTest.java` tab is active, showing the following code:

```
15 private CourseList list = new CourseList();
16 public CourseListTest(){
17     readData();
18 }
19
20 @Test
21 void add() {
22     CourseList n = new CourseList();
23     assertEquals( expected: true,n.add(new Course()));
24 }
25
26 @Test
27 void isEmpty() {
28     assertEquals( expected: false,list.isEmpty());
29 }
30
31 @Test
32 void size() {
33     assertEquals( expected: 54,list.size());
34 }
35
36 @Test
37 void get() {
38     assertEquals( expected: 1,list.get(0).getData().getSemester());
39 }
40
41 @Test
42 void remove() {
43     Course n = list.get(10).getData();
44     assertEquals(n,list.remove( index: 10).getData());
45 }
46
47 @Test
48 void iterator() {
49
50     CourseList.CourseListIterator itr = list.iterator();
51     assertEquals(list.get(0).getData(),itr.next());
52     assertEquals( expected: 1,itr.nextInSemester(1).getSemester());
53 }
54
```

The left sidebar shows the project structure with folders `Part1`, `Part2`, and `Part3`. Under `Part3`, the `CourseListTest` class is selected. The bottom panel shows the test results for `CourseListTest`, indicating that all 6 tests passed.

Test Results	Time
CourseListTest	15ms
remove()	15ms
add()	
get()	
size()	
iterator()	
isEmpty()	

Process finished with exit code 0

3.2 Running Results

Part1: Aşağıda Part1 Main testi verilmiştir.

```
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
MainPart1 > main()

Run MainPart1
"C:\Program Files\Java\jdk1.8.0_144\bin\java" ...
***** TEST PART 1 *****
*****
getByRange(1,2)
1 CSE 101 Introduction To Computer Engineering 8 3 3+0+0
1 CSE 107 Introduction To Computer Science Laboratory 2 1 0+0+2
getByCode( CSE 101 )
1 CSE 101 Introduction To Computer Engineering 8 3 3+0+0
listSemesterCourses(3)
3 CSE 241 Object Oriented Programming 9 5 3+2+0
3 CSE 211 Discrete Mathematics 6 3 3+0+0
3 CSE 231 Circuits And Electronics 8 4 4+0+0
3 CSE 233 Circuits And Electronics Laboratory 2 1 0+0+2
3 XXX XXX Teknik Olmayan Seemeli (SSB) 3 2 2+0+0
3 EN 111 English For Business Life 2 2 2+0+0

*** getByRange(2,1) -> Test for wrong index throw Exception
java.security.InvalidParameterException
at Part1.GTUComputerEngineering.getByRange(GTUComputerEngineering.java:110)
at Part1.MainPart1.main(MainPart1.java:37)

Process finished with exit code 0
```

Exception testing

Part2: Aşağıda Part2 Main testi verilmiştir.

The screenshot displays an IDE with a project named 'MainPart2'. The left sidebar shows a file tree with folders 'Part2', 'Part3', and 'TESTS'. The 'MainPart2' folder is selected, showing 'MainPart2.java'. The code in the editor is as follows:

```
45
46     System.out.println("\n      GET METHOD FOR DISABLED ELEMENT");
47     list.get(0);
48     list.get(10);
49
50     System.out.println("\n      SET METHOD FOR DISABLED ELEMENT");
51     list.set(1, list.get(2));
52
53     System.out.println("\n      SIZE() and SUPERSIZE()");
54     System.out.println("      "+list.size()+"      "+list.superSize());
55
56     System.out.println("\n enable(10) ");
57     list.enable( index: 10);
58     System.out.println(list.get(10).toString());
59
60     System.out.println("      SHOW DISABLED");
61     show = list.showDisabled();
62     for (int i=0; i<show.size(); i++)
63         System.out.println(show.get(i).toString());
64
```

The Run console shows the output of the program:

```
Run MainPart2
"C:\Program Files\Java\jdk1.8.0_144\bin\java" ...
*****      TEST PART 2      *****
*****
disable(0) disable(1) disable(10)
SHOW DISABLED
1 XXX XXX Teknik Olmayan Seemeli (SSC) 2 1 2+0+0
1 CSE 101 Introduction To Computer Engineering 8 3 3+0+0
2 CSE 108 Computer Programming Laboratory 2 1 0+0+2

GET METHOD FOR DISABLED ELEMENT
The index of : 0 is Disabled
The index of : 10 is Disabled

SET METHOD FOR DISABLED ELEMENT
The index of : 1 is Disabled

SIZE() and SUPERSIZE()
51      54

enable(10)
2 CSE 108 Computer Programming Laboratory 2 1 0+0+2

SHOW DISABLED
1 XXX XXX Teknik Olmayan Seemeli (SSC) 2 1 2+0+0
1 CSE 101 Introduction To Computer Engineering 8 3 3+0+0
```

Part3: Aşağıda Part3 Main testi verilmiştir.

```
Run MainPart3
"C:\Program Files\Java\jdk1.8.0_144\bin\java" ...
*****      TEST PART 3      *****
*****
LIST SIZE : 54
Get(2) :> 1   CSE 107 Introduction To Computer Science Laboratory 2   1   0+0+2

Get(3) :> 1   MATH 101   Calculus I   7   5   5+0+0

REMOVED(2) :> 1 CSE 107 Introduction To Computer Science Laboratory 2   1   0+0+2

LIST SIZE :> 53
Get(2) :> 1   MATH 101   Calculus I   7   5   5+0+0

Get(3) :> 1   PHYS 121   Physics I    6   4   3+0+0

The Iterator Use next() for i < 15
1 XXX XXX Teknik Olmayan Seömeli (SSC) 2 1 2+0+0
1 CSE 101 Introduction To Computer Engineering 8 3 3+0+0
1 MATH 101 Calculus I 7 5 5+0+0
1 PHYS 121 Physics I 6 4 3+0+0
1 PHYS 151 Physics Laboratory I 1 1 0+0+2
1 SSTR 101 Principles Of Atatörk And The History Of Turkish Revolution I 2 2 2+0+0
1 TUR 101 Turkish I 2 2 2+0+0
2 XXX XXX Teknik Olmayan Seömeli (SSC) 2 1 2+0+0
2 CSE 102 Computer Programming 8 4 4+0+0
2 CSE 108 Computer Programming Laboratory 2 1 0+0+2
2 MATH 102 Calculus II 7 5 5+0+0
2 PHYS 122 Physics II 6 4 3+0+0
2 PHYS 152 Physics Laboratory II 1 1 0+0+2
2 SSTR 102 Principles Of Atatörk And The History Of Turkish Revolution II 2 2 2+0+0
2 TUR 102 Turkish II 2 2 2+0+0

The Iterator Use nextInSemester(3) for i < 20
3 CSE 241 Object Oriented Programming 9 5 3+2+0
3 CSE 211 Discrete Mathematics 6 3 3+0+0
3 CSE 231 Circuits And Electronics 8 4 4+0+0
3 CSE 233 Circuits And Electronics Laboratory 2 1 0+0+2
3 XXX XXX Teknik Olmayan Seömeli (SSB) 3 2 2+0+0
3 EN 111 English For Business Life 2 2 2+0+0
3 CSE 241 Object Oriented Programming 9 5 3+2+0
3 CSE 211 Discrete Mathematics 6 3 3+0+0
3 CSE 231 Circuits And Electronics 8 4 4+0+0
3 CSE 233 Circuits And Electronics Laboratory 2 1 0+0+2
3 XXX XXX Teknik Olmayan Seömeli (SSB) 3 2 2+0+0
3 EN 111 English For Business Life 2 2 2+0+0
3 CSE 241 Object Oriented Programming 9 5 3+2+0
3 CSE 211 Discrete Mathematics 6 3 3+0+0
3 CSE 231 Circuits And Electronics 8 4 4+0+0
3 CSE 233 Circuits And Electronics Laboratory 2 1 0+0+2
3 XXX XXX Teknik Olmayan Seömeli (SSB) 3 2 2+0+0
3 EN 111 English For Business Life 2 2 2+0+0
3 CSE 241 Object Oriented Programming 9 5 3+2+0
3 CSE 211 Discrete Mathematics 6 3 3+0+0
```

4 ANALYSIS

```
HW_3 > src > Part1 > GTUComputerEngineering >
GTUComputerEngineering.java x
67
68
69
70 /**
71  * getByCode Returns all courses which have given course code.
72  * @param code The searching Course code
73  * @return List of Course
74  * @throws Exception There is no Matching Exception or NullPointer Exception
75  */
76 public List<Course> getByCode(String code) throws Exception {
77     Iterator iter = list.iterator();
78     List<Course> ret = new LinkedList<>();
79     if(code!=null) {
80         while (iter.hasNext()) {
81             Course c = (Course) iter.next();
82             if (c.getCourseCode().equals(code))
83                 ret.add(c);
84         }
85     }else
86         throw new NullPointerException();
87     if(ret.isEmpty()) {
88         throw new Exception(" getByCode -> There is no matching");
89     }
90
91     return ret;
92 }
93
94 /**
95  * listSemesterCourses Returns all courses on given semester
96  * @param semester The searching semester
97  * @return List of Course
98  * @throws Exception There is no Matching or InvalidParameter Exceptions
99  */
100 public List<Course> listSemesterCourses (int semester) throws Exception {
101     Iterator iter = list.iterator();
102     if(semester<1 || semester>8)
103         throw new InvalidParameterException();
104
105     List<Course> ret = new LinkedList<>();
106     while(iter.hasNext()){
107         Course c = (Course)iter.next();
108         if(c.getSemester()==semester)
109             ret.add(c);
110     }
111     if(ret.isEmpty()) {
112         throw new Exception("listSemesterCourses -> There is no matching");
113     }
114     return ret;
115 }
116
117 /**
118  * getByRange Returns all courses from given start index to last index.
119  * @param start_index The start index for search
120  * @param last_index The last index for search
121  * @return List of Course
122  * @throws Exception InvalidParameterException
123  */
124 public List<Course> getByRange(int start_index, int last_index) throws Exception {
125
126     if(start_index < 0 || start_index > list.size() || last_index < 0
127         || last_index > list.size() || last_index < start_index)
128         throw new InvalidParameterException();
129     List<Course> ret = new LinkedList<>();
130     for(int i=start_index; i<=last_index; i++){
131         ret.add(list.get(i));
132     }
133     return ret;
134 }
135
```

BestCase

$O(1)$ zaman alır code null gelirse best case olur.

WorstCase

$O(n)$ zaman alır bütün listeyi dolaşarak bulduğu elemları ekler ekleme işlemi ve diğer işlemler $O(1)$ olarak kabul edilmiştir

BestCase

$O(1)$ Semester bilgisinin yanlış gelmesi veya listin boş olma durumunda

WorstCase

$O(n)$ zaman alır aranan elemanların listin sonunda olması

BestCase

$O(1)$ zaman alır index bilgilerinin yanlış gelmesi durumunda exception fırlatır

WorstCase

$O(n)$ zaman alır start indexi 0 last indexi ise n verildiği zaman bu süreyi alır liste ekleme süresi $O(1)$ olarak kabul edilmiştir


```

22 public void disable(int index){
23     if(index<0 || index > situation.size())
24         throw new InvalidParameterException();
25     else{
26         situation.set(index,0); O(n)
27         disableCount++;
28     }
29 }
30
31 /**...*/
32 public void enable(int index){
33     if(index<0 || index > situation.size())
34         throw new InvalidParameterException();
35     else{
36         situation.set(index,1); O(n)
37         disableCount--;
38     }
39 }
40
41 /**...*/
42 public List<E> showDisabled(){
43     List<E> ret = new LinkedList<E>();
44     for (int i=0; i<situation.size(); i++){
45         if(situation.get(i)==0){
46             enable(i);
47             ret.add(this.get(i));
48             disable(i);
49         }
50     }
51     return ret;
52 }
53
54 /**...*/
55 @Override
56 public void add(final int index, final E element) {
57     /* 1 : enable
58      * 0 : disable
59      */
60     situation.add(index, element);
61     super.add(index,element);
62 }
63
64 /**...*/
65 @Override
66 public boolean add(E e) {
67     situation.add(1);
68     return super.add(e);
69 }
70
71 /**...*/
72 @Override
73 public E remove(final int index) {
74     if(situation.get(index)==0){
75         System.out.println("The index of : "+index+" is Disabled");
76         return null;
77     }
78     situation.remove(index); O(n)
79     return super.remove(index); O(n)
80 }
81
82 /**...*/
83 @Override
84 public ListIterator<E> listIterator(final int index) {
85     if(index<0 || index > situation.size())
86         throw new InvalidParameterException();
87     if(situation.get(index)==0){
88         System.out.println("The index of : "+index+" is Disabled");
89         return null;
90     }
91     return super.listIterator(index); O(1)
92 }
93
94 /**...*/
95 @Override
96 public E set(final int index, final E element) {
97     if(index<0 || index > situation.size())
98         throw new InvalidParameterException();
99     if(situation.get(index)==0){
100         System.out.println("The index of : "+index+" is Disabled");
101         return null;
102     }
103     return super.set(index,element); O(n)
104 }
105
106 /**...*/
107 @Override
108 public E get(final int index) {
109     if(index<0 || index > situation.size())
110         throw new InvalidParameterException();
111     if(situation.get(index)==0){
112         System.out.println("The index of : "+index+" is Disabled");
113         return null;
114     }
115     return super.get(index); O(n)
116 }

```

BestCase
O(1) zaman alır index bilgisinin yanlış gelme durumu

WorstCase
O(n) zaman alır set fonksiyonundan dolayı

BestCase
O(1) zaman alır index bilgisinin yanlış gelme durumu

WorstCase
O(n) zaman alır set fonksiyonundan dolayı

BestCase
O(1) zaman alır eğer situation.size() = 0 ise

WorstCase
O(n) zaman alır eğer situation.size() = n ise

BestCase
O(1) indexin hatalı gelmesi durumunda

WorstCase
O(n), n. indexe elemanı koymak için elemana gitmesüresi

O(1) zaman alır eğer sistemde tailli gösteren bir bilgi var ise

BestCase
O(1) zaman alır verilen indexin yanlış gelmesi

WorstCase
O(3n) = O(n) zaman alır verilen index n. index ise gerçekleşir

BestCase
O(1) indexin yanlış gelmesi durumunda

WorstCase
O(n) verilen index n. eleman ise enable disable kontrolü için alan süre

BestCase
O(1) yanlış bir indexin gelme durumunda.

WorstCase
O(2n) = O(n) verilen index son eleman ise ve enable ise gerçekleşen durum

BestCase
O(1) yanlış bir indexin gelme durumunda.

WorstCase
O(2n) = O(n) verilen index son eleman ise ve enable ise gerçekleşen durum


```

39 public class Node
40 {
41     private Node next;
42     private Course data;
43     private int semester;
44     /**...*/
45     public Node(Course dat)
46     {
47         data = dat;
48         semester = dat.getSemester();
49         this.next = null;
50     }
51
52     /**...*/
53     public Node() {
54         this.data = null;
55         this.next = null;
56     }
57
58     /**...*/
59     public Course getData() { return data; }
60
61     /**...*/
62     public void setData(Course data) {
63         if(data==null)
64             throw new NullPointerException();
65         this.data = data;
66     }
67
68     /**...*/
69     public int getSemester() { return semester; }
70 }
71
72 /**...*/
73 public boolean add(Course data) {
74     if (data == null) {
75         return false;
76     }
77     Node currentNode = new Node(data);
78     if (isEmpty()){
79         head = currentNode;
80         tail=head;
81     } else {
82         tail.next = currentNode;
83         tail=tail.next;
84     }
85     countNodes++;
86     return true;
87 }
88
89 /**...*/
90 public boolean isEmpty() { return countNodes<=0; }
91
92 /**...*/
93 public int size() { return countNodes; }
94
95 /**...*/
96 public Node get(int index){
97     if(index<0)
98         throw new IllegalArgumentException();
99     if (index>=countNodes)
100         throw new IndexOutOfBoundsException();
101     Node temp = head;
102     for (int i=1; i<=index; i++){
103         temp=temp.next;
104     }
105     return temp;
106 }
107
108 /**...*/
109 public Node remove(int index){
110     if(index<0)
111         throw new IllegalArgumentException();
112     if (index>=countNodes)
113         throw new IndexOutOfBoundsException();
114     Node temp = head;
115     Node element = null;
116     for (int i=1; i<index; i++){
117         temp=temp.next;
118     }
119     element = temp.next;
120     temp.next=temp.next.next;
121     countNodes--;
122     return element;
123 }
124 }

```

O(1) Bütün yöntemler ve metodlar constant(O(1)) zamanda çalışıyor

O(1) Bütün yöntemler ve metodlar constant(O(1)) zamanda çalışıyor

BestCase
O(1) zaman alır indexin hatalı gelmesi durumu veya 0 gelmesi durumunda

WorstCase
O(n) zaman alır index n olarak verildiğinde for döngüsünden dolayı

BestCase
O(1) zaman alır indexin hatalı gelmesi durumu veya 0 gelmesi durumunda

WorstCase
O(n) zaman alır index n olarak verildiğinde for döngüsünden dolayı

```

24
25
26 /**...*/
30 public CourseListIterator iterator() { return new CourseListIterator(head); }
33
34 /**
35  * The CourseListIterator class
36  * The Node contains Course object
37  */
38 public class CourseListIterator implements Iterator<Course> {
39
40     private Node nextNode, head, semNext;
41     private Node lastReturnedNode;
42     private Node previousNode;
43     private boolean flag = false;
44     /**
45      * The Iterator Constructor
46      * @param node The Node contains Course object
47      */
48     public CourseListIterator(Node node) {
49         this.head = node;
50         this.nextNode = node;
51         this.semNext = node;
52     }
53
54     /**
55      * is there a next element on list
56      * @return boolean
57      */
58     public boolean hasNext ()
59     {
60         if(flag)
61             return semNext != null;
62         return nextNode != null;
63     }
64
65     /**
66      * next element
67      * @return Node
68      * @throws NoSuchElementException
69      */
70     public Course next () throws NoSuchElementException
71     {
72         flag = false;
73         if (!this.hasNext ())
74             throw new NoSuchElementException();
75
76         previousNode = lastReturnedNode;
77         lastReturnedNode = nextNode;
78         nextNode = nextNode.next;
79         return lastReturnedNode.data;
80     }
81
82     /**
83      * Remove the element
84      * @throws IllegalStateException
85      */
86     public void remove() throws IllegalStateException
87     {
88         if (lastReturnedNode == null) {
89             throw new IllegalStateException ();
90         }
91         else {
92             lastReturnedNode = lastReturnedNode.next;
93             countNodes--;
94         }
95     }
96
97     /**
98      * Move next node in same semester
99      * @param semester the semester
100     * @return Course
101     */
102     public Course nextInSemester(int semester) {
103         flag = true;
104         if (semester < 1 || semester > 8)
105             throw new IllegalArgumentException();
106         boolean find = false;
107         while (!find) {
108             if (semesterSearch().getSemester() == semester)
109                 find = true;
110         }
111         return lastReturnedNode.getData();
112     }
113
114     /**
115      * Helper for nextInSemester(int semester)
116      * @return Node
117      */
118     private Node semesterSearch() {
119         if (semNext.next == null)
120             semNext.next = head;
121         previousNode = lastReturnedNode;
122         lastReturnedNode = semNext;
123         semNext = semNext.next;
124         return lastReturnedNode;
125     }
126 }
127

```

$O(1)$ Bütün yöntemler ve metodlar constant($O(1)$) zaman alır.

BestCase

$O(1)$ zaman alır semester bilgisinin yanlış gelmesi durumunda

WorstCase

$O(\infty)$ Sonsuz zaman alabilir kullanıcıya bağlı bir durumdur döngü circular linked list üzerinde olduğu için kullanıcının belirlediği şarta göre sonlanır

$O(1)$

$O(1)$ Bütün yöntemler ve metodlar constant($O(1)$) zaman alır.