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 gerçeklememiz istenilmiştir. Linked List classı extend edilmeyecektir şartı getirilerek. Bu yapıyı değişken(veriable) 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.

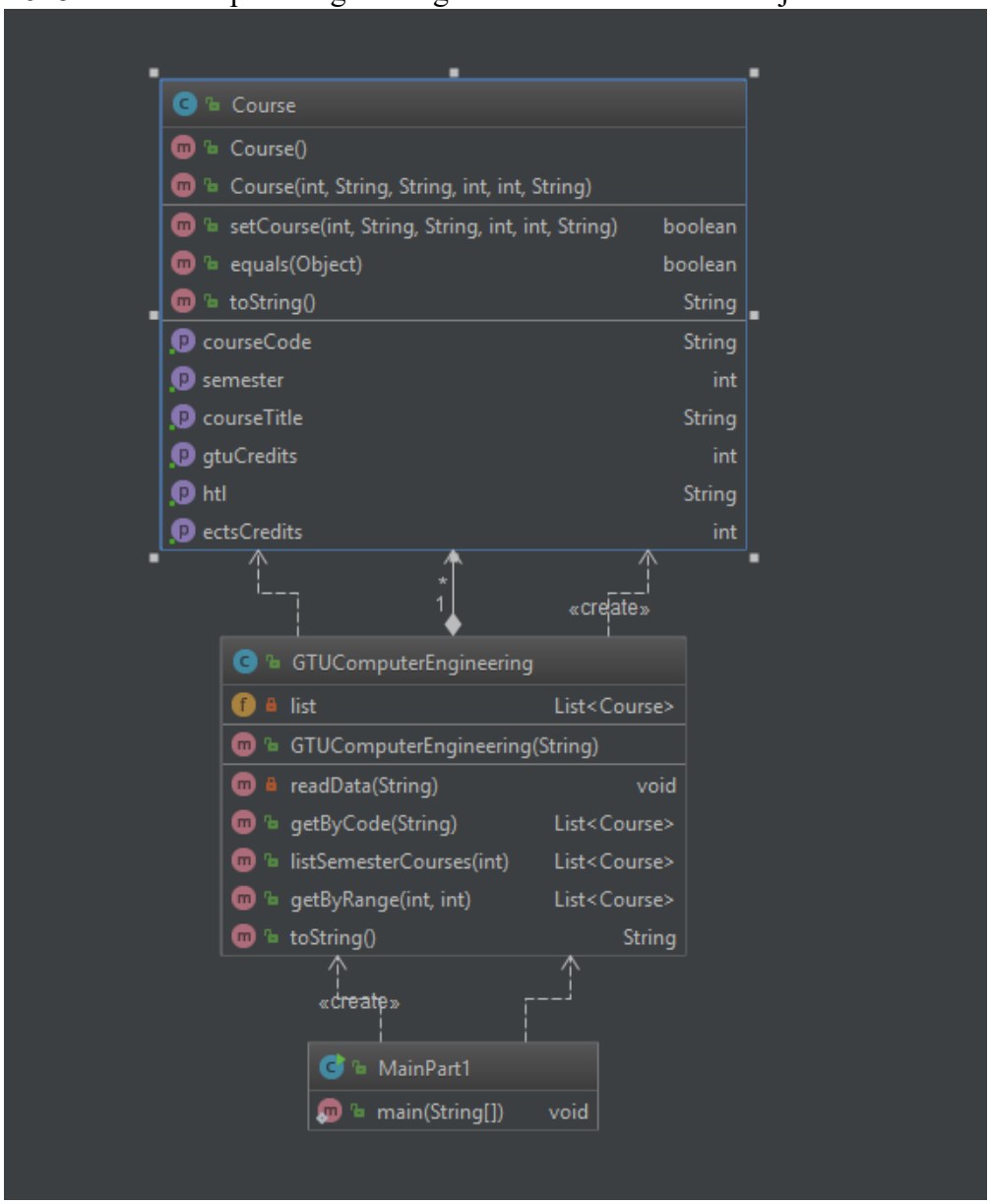
Part2: Bu bölümde kullanıcı sisteme Course.csv bütün derslere ait bilgiler bulunan dosyayı vermelidir bütün bilgiler bu dosyadan okunarak Course objeleri şeklinde MyLinkedList yapısında tutulur. Bu listin en temel özelliği içerisindeki elemanların belli metodlarını kullanıma kilitleyip açabilmektir. 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 semesterler) 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

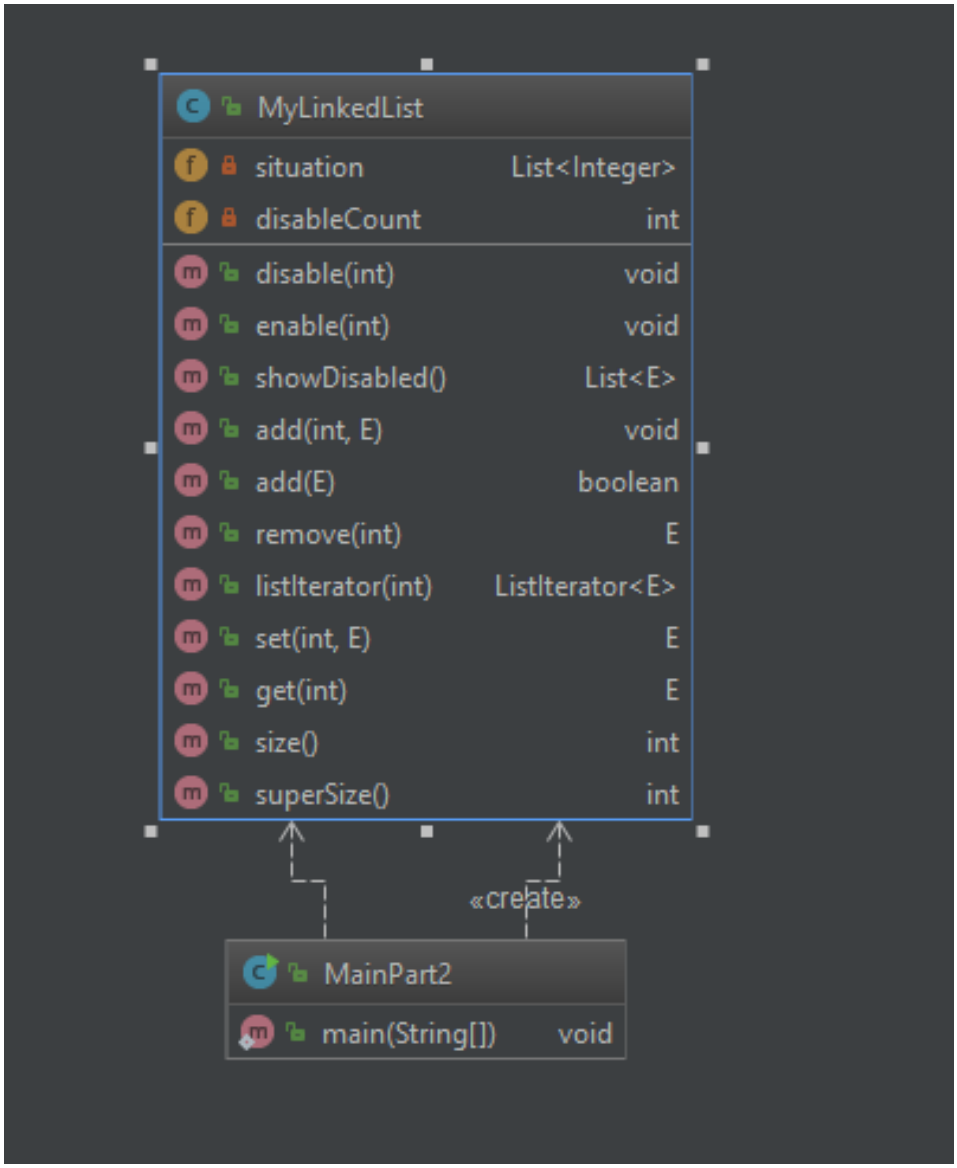
2.1 Class Diagrams

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.

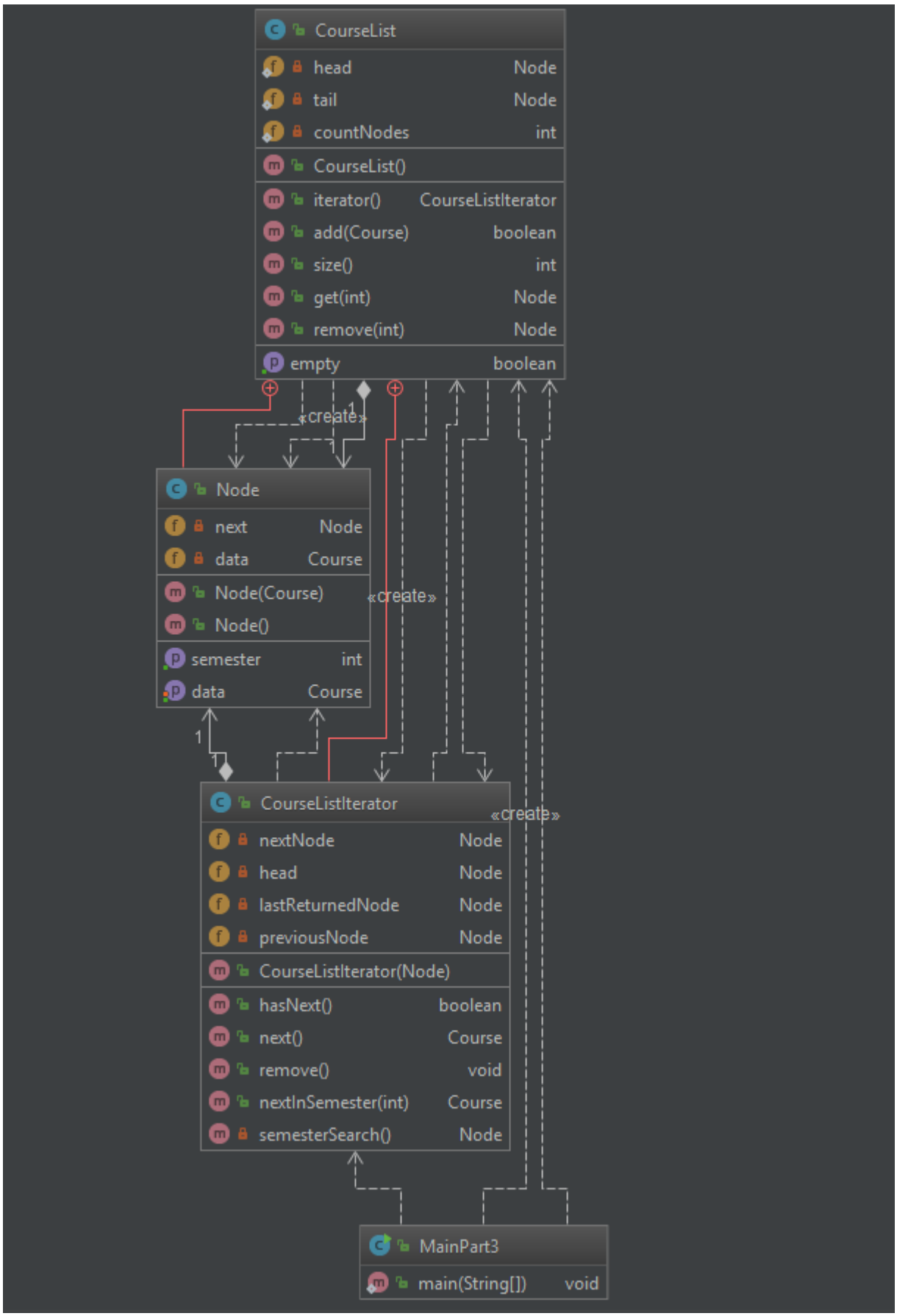
- add(): Liste yeni eleman ekler
- remove(int index): Listten verilen indexteki eleman siler
- size(): listenin büyüklüğünü döndürür

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


CourseList
head Node
tail Node
countNodes int
CourseList()
iterator() CourseListIterator
add(Course) boolean
size() int
get(int) Node
remove(int) Node
empty boolean
«create»
1
Node
next Node
data Course
Node(Course)
Node()
semester int
data Course
«create»
1
1
CourseListIterator
«create»
nextNode Node
head Node
lastReturnedNode Node
previousNode Node
CourseListIterator(Node)
hasNext() boolean
next() Course
remove() void
nextInSemester(int) Course
semesterSearch() Node
MainPart3
main(String[]) void

2.2 Problem Solution Approac

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

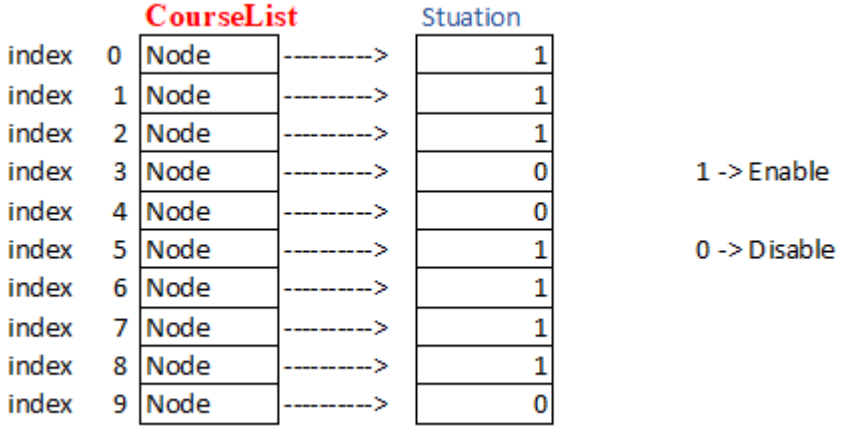
- getByCode(String code) : Verilen koddaki dersleri döndürecek
- listSemesterCourses (int semester): Verilen dönemdeki dersleri döndürecek
- getByRange(int start_index, int last_index): Verilen aralıktaki dersleri döndürecek

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 medodlar 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 elemanının 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.

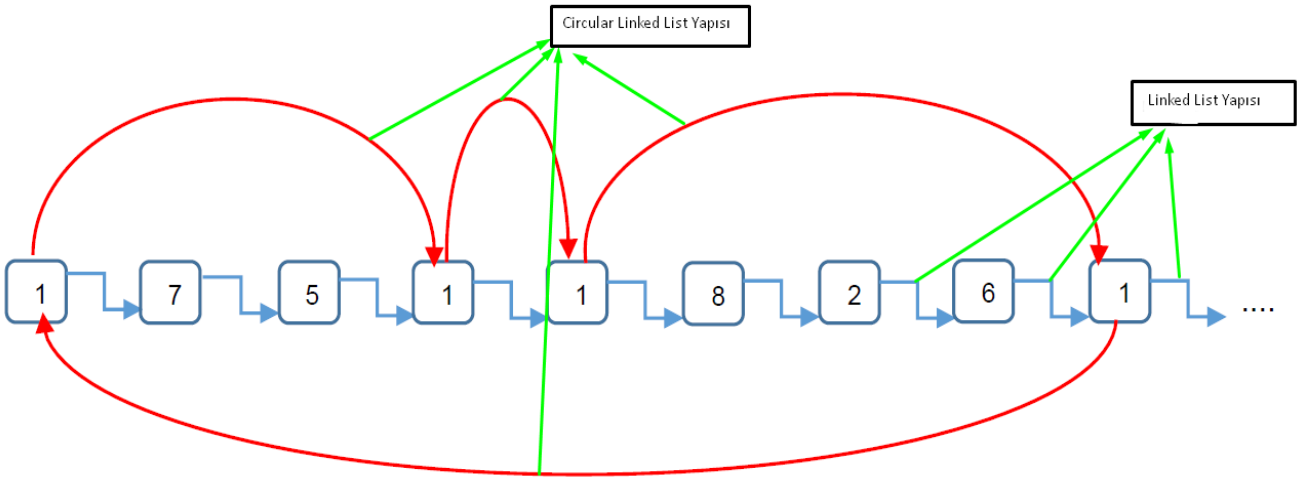
- add(E obj)
- add(int index, E obj)
- remove(int index)
- listIterator(int index)
- set(int index)
- get(int index)

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.



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.

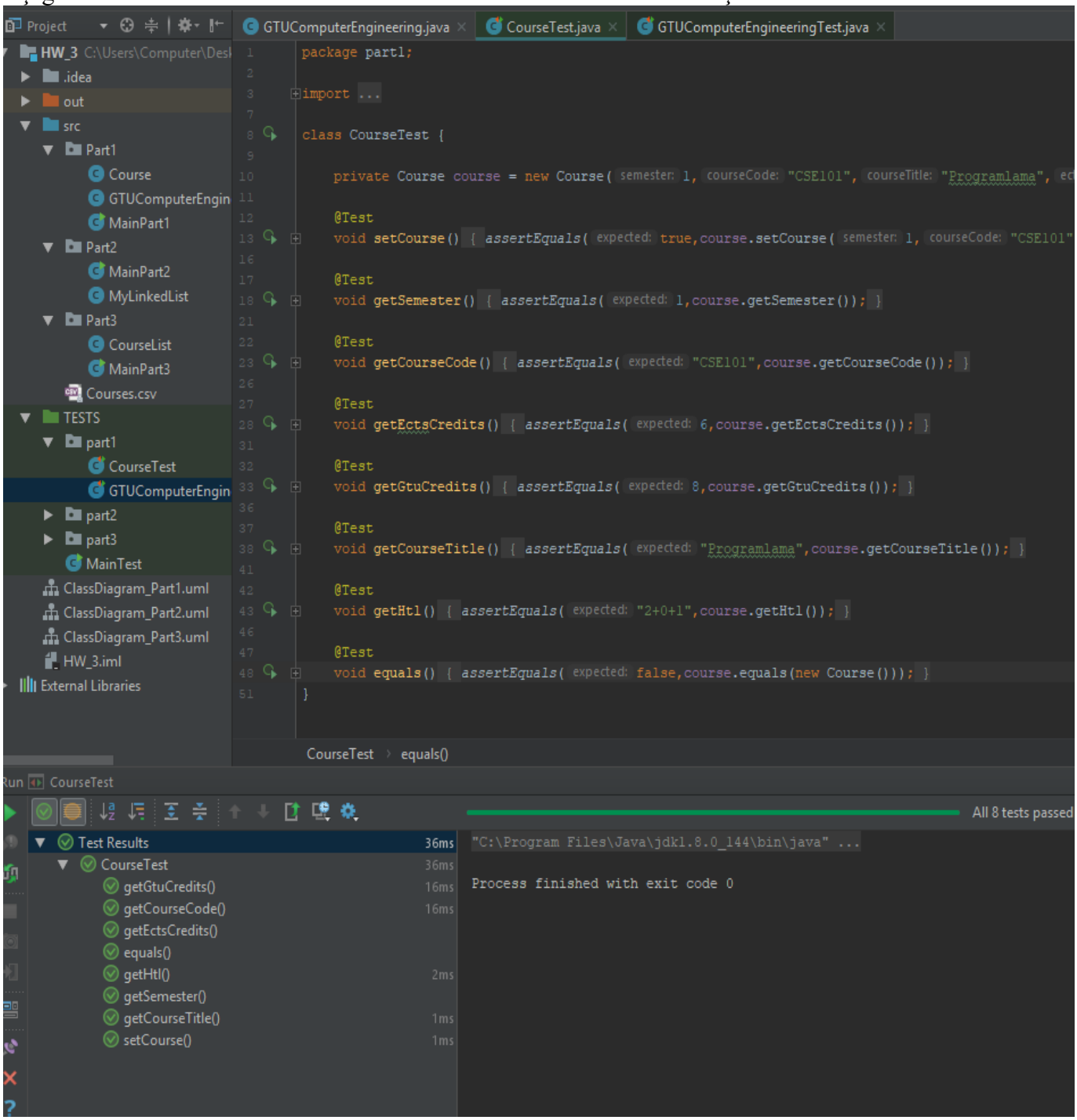
- add(Course cs): Liste yeni eleman(Course) ekler
- remove(int index) , remove() : Listten eleman siler
- next(): bir sonraki elemana gide
- nextInSemester(int semester):Aynı semester(dönem) de bulunan bir sonraki derse gider ve liste bitince tekrar başa dönerek devam eder.
- size(): listenin enable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



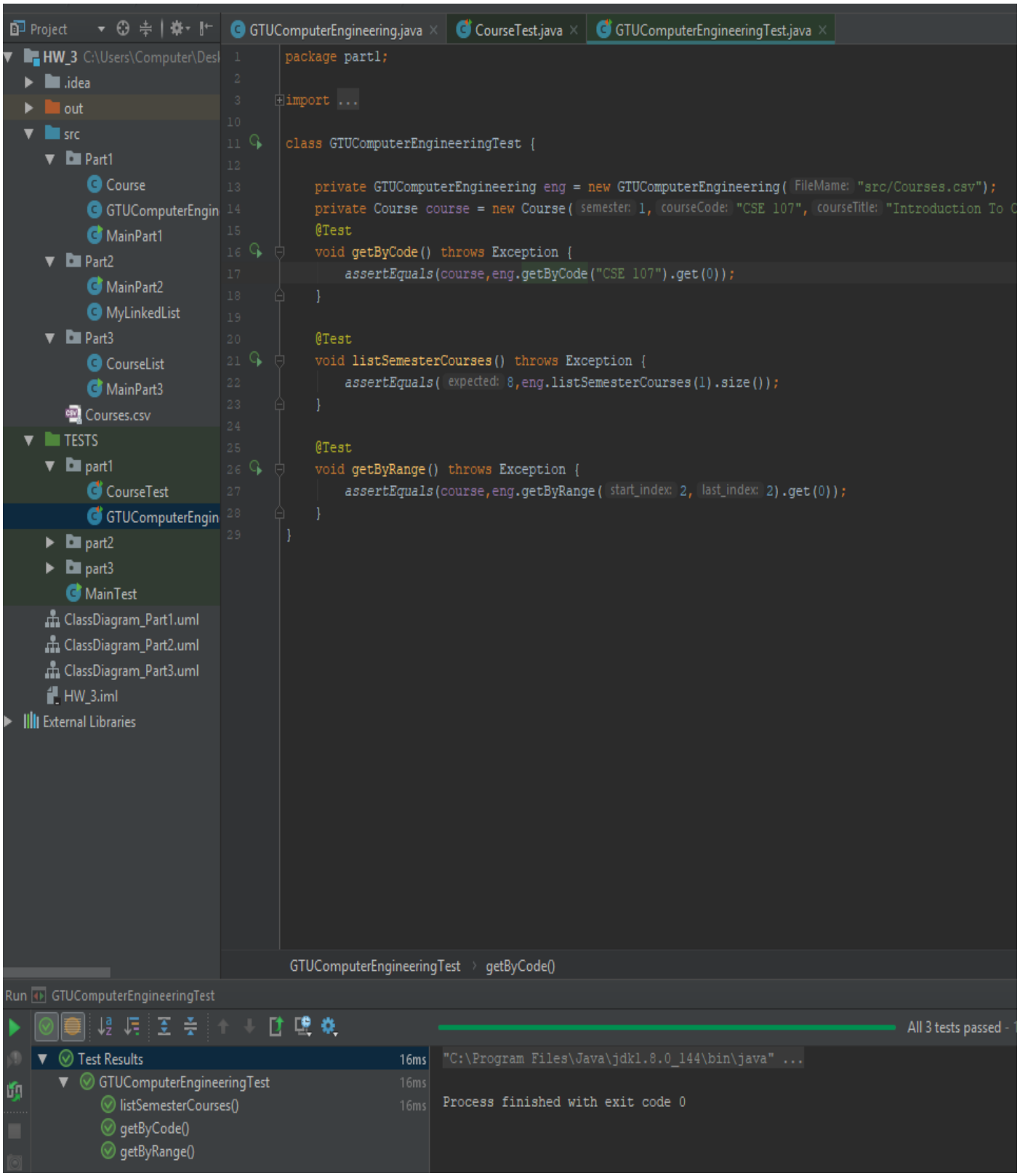
3 RESULT

3.1 Test Cases

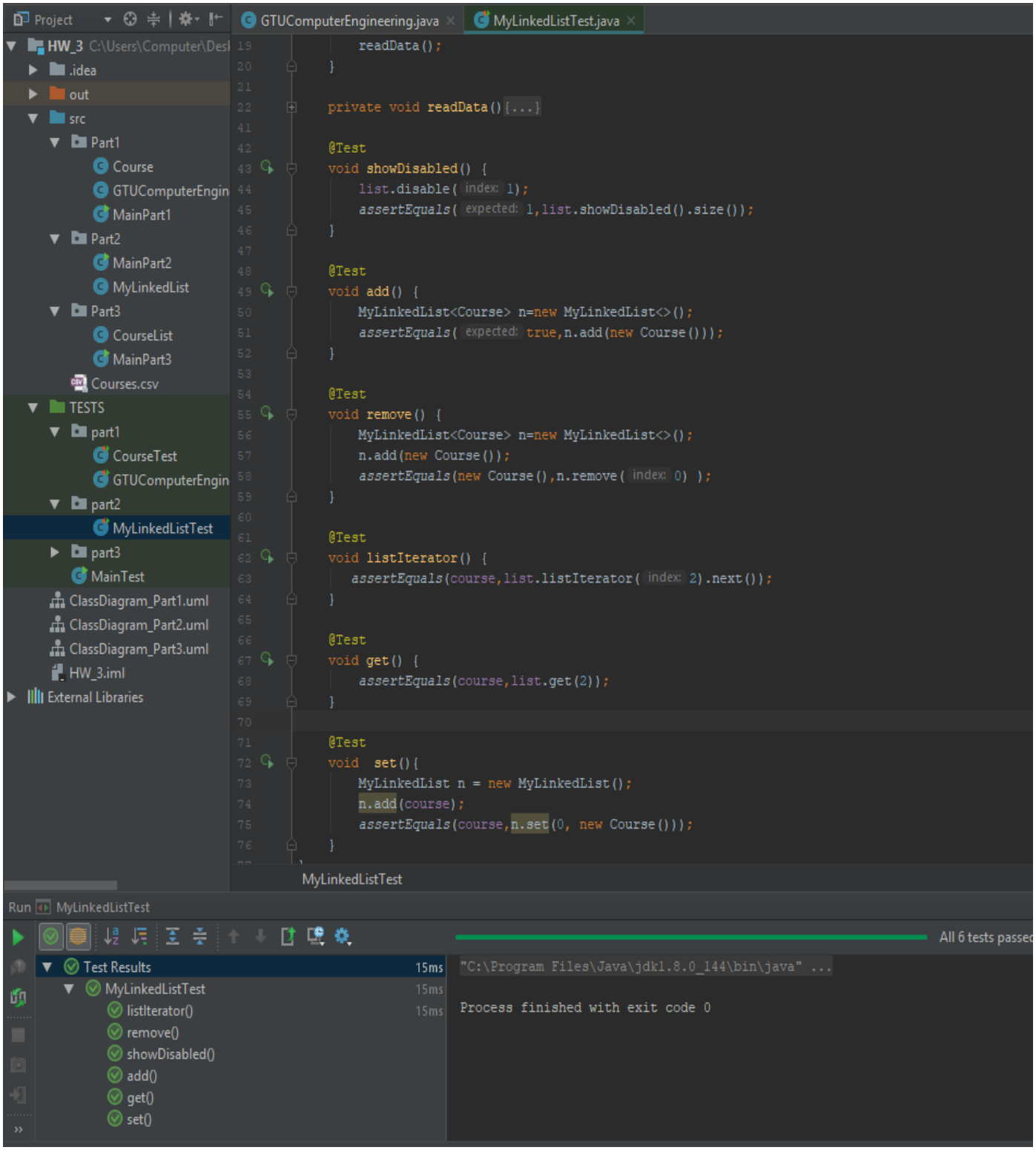
Part1: Aşağıda Part1 JUnit testleri verilmiştir
Aşağıdaki resimde Course Classının bazı metodları JUnit testle sınanmıştır



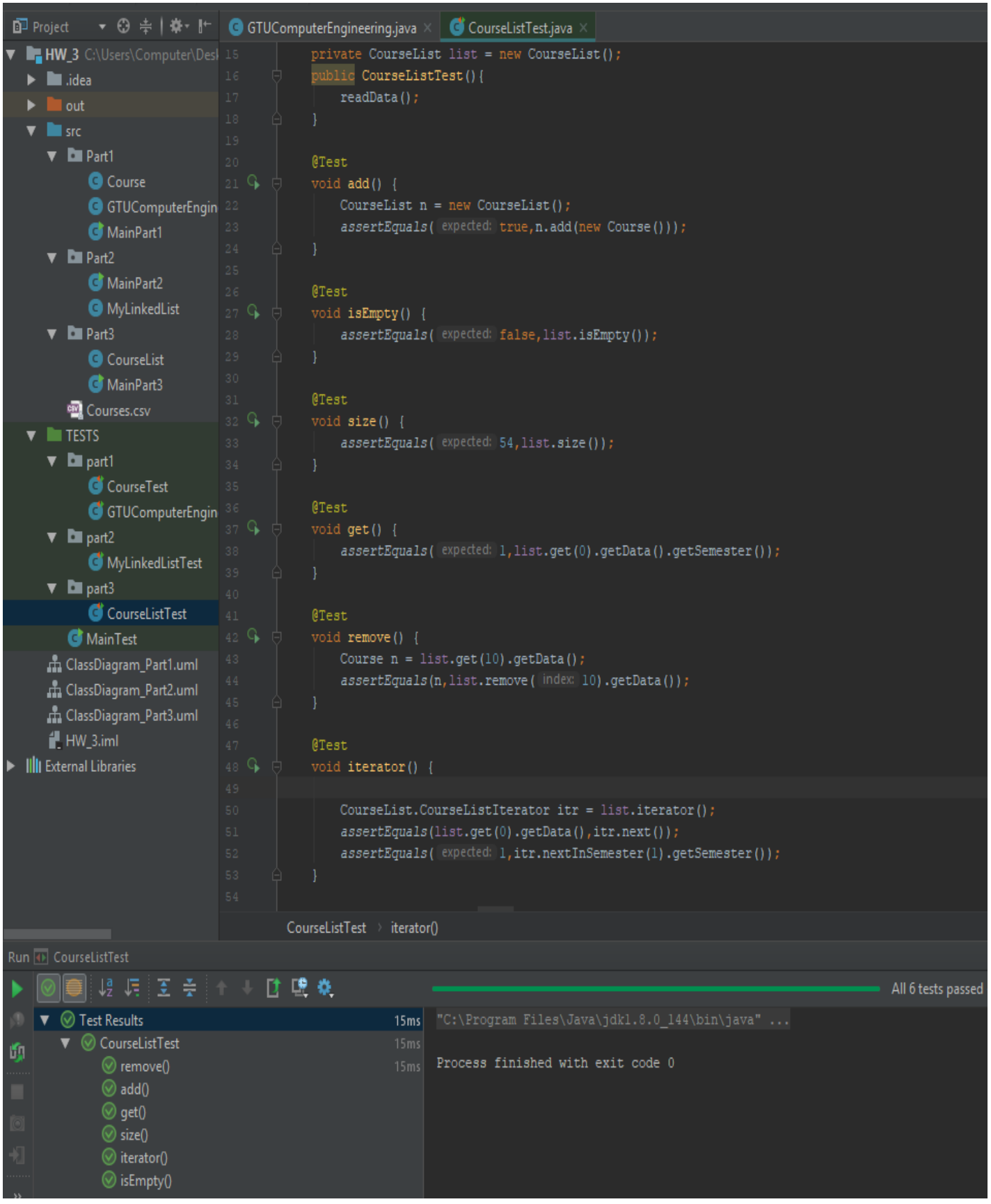
Aşağıdaki resimde ise GTUComputerEngineering classının metodları JUnit test ile sınanmıştır. Testler başarı ile sonuçlanmıştır.



Part2: Aşağıda Part2 JUnit testleri verilmiştir.

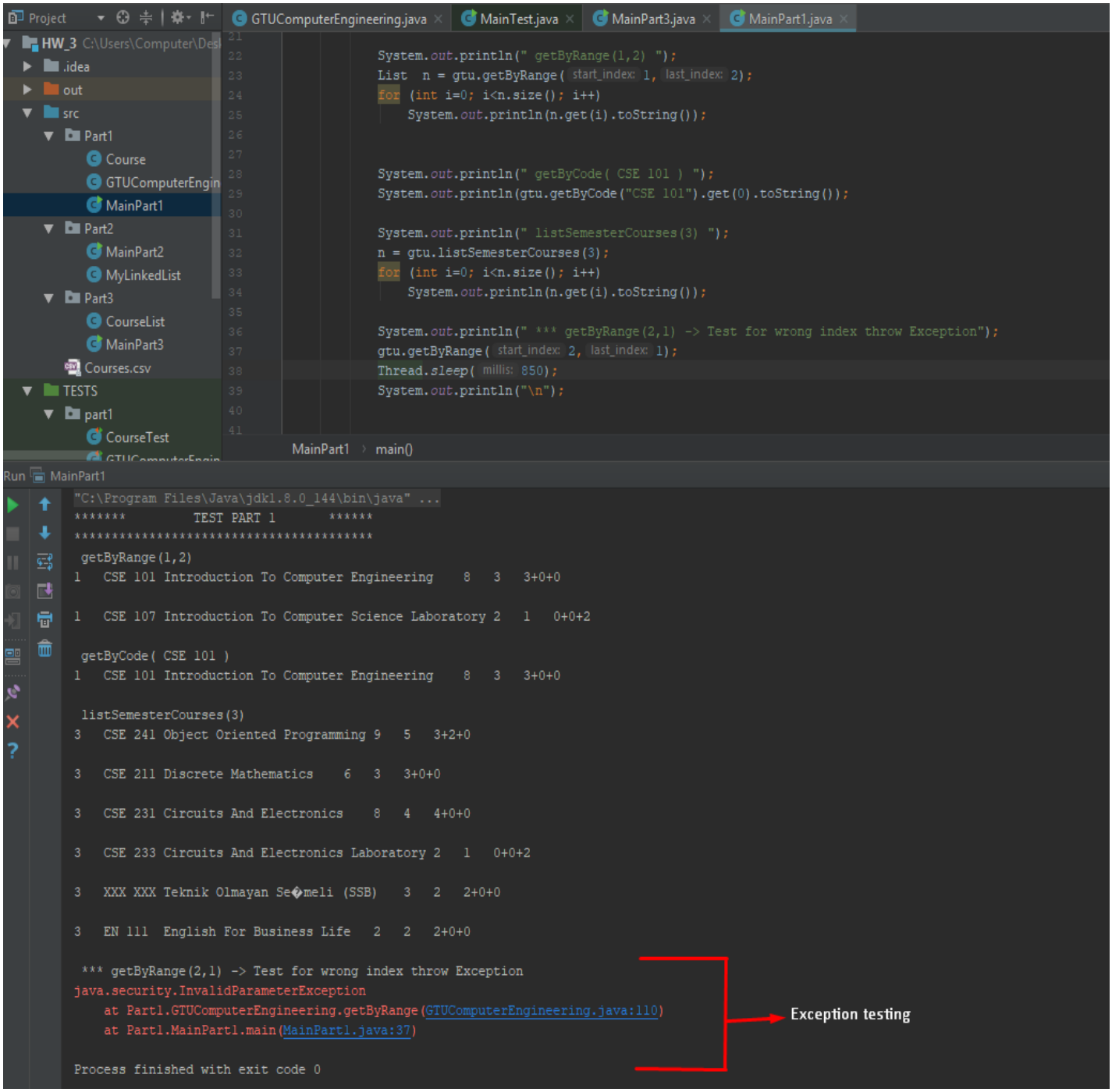


Part3: Aşağıda Part3 JUnit testleri verilmiştir.

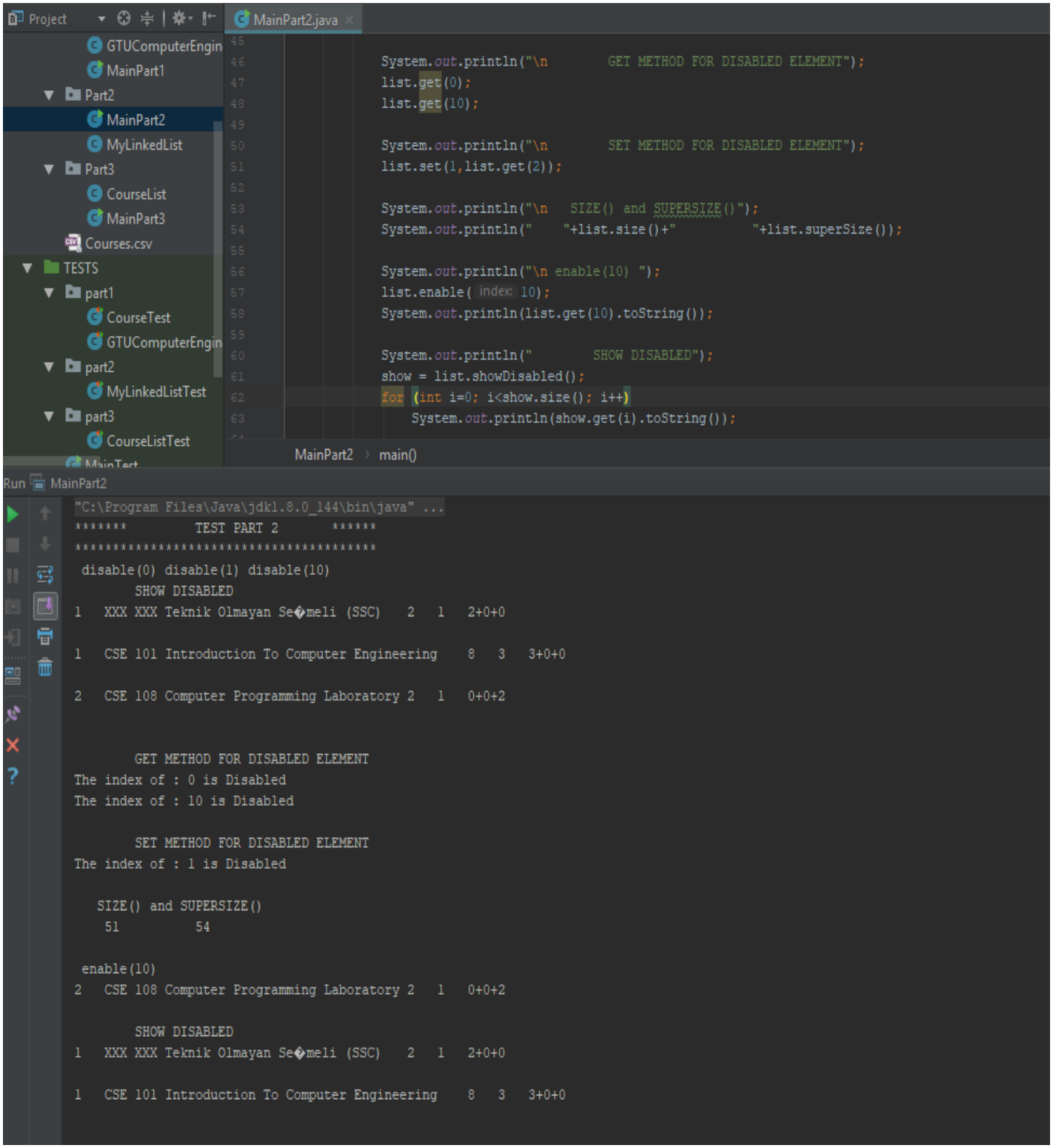


3.2 Running Results

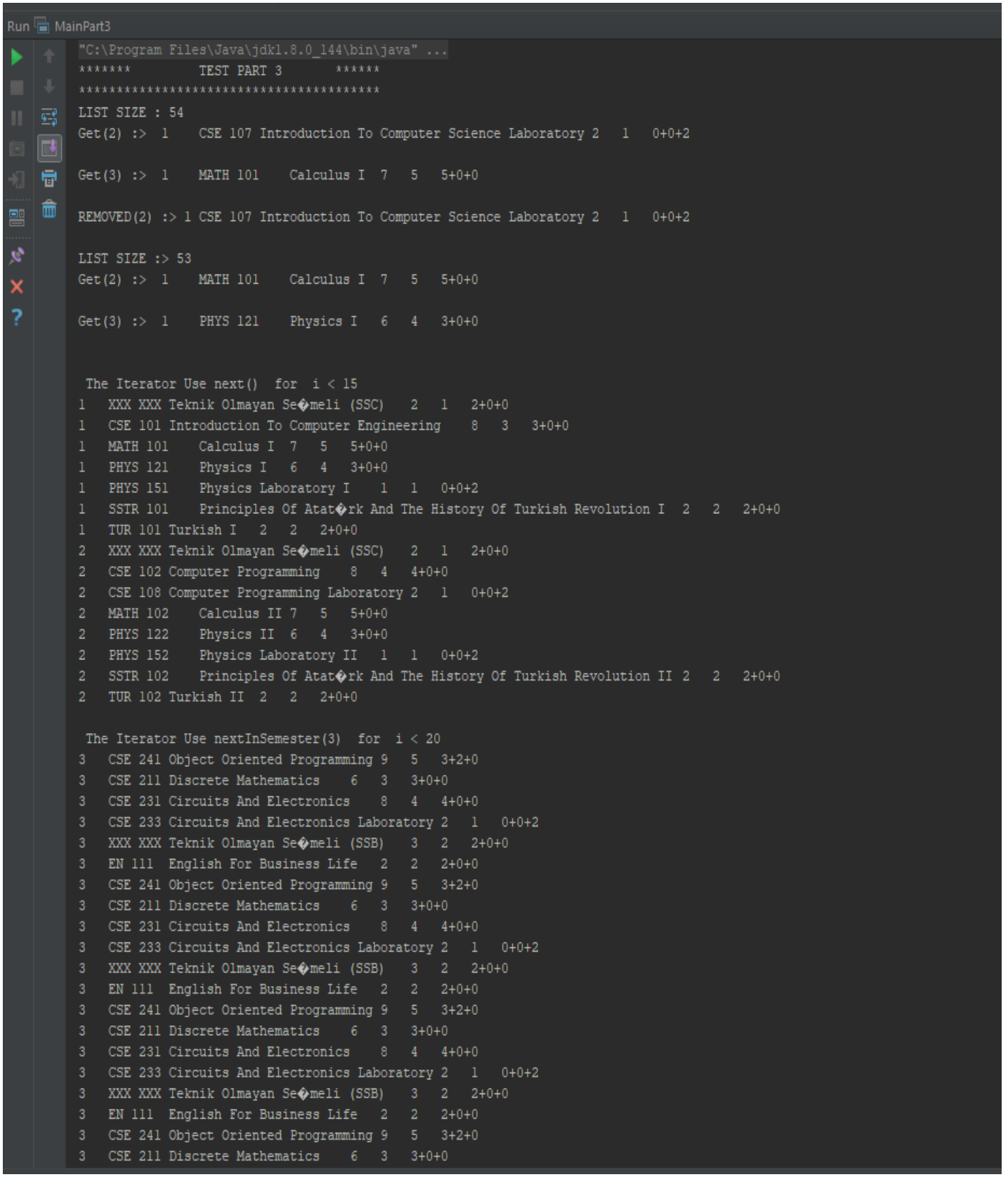
Part1: Aşağıda Part1 Main testi verilmiştir.



Part2: Aşağıda Part2 Main testi verilmiştir.



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 Se?meli (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 Atat?rk 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 Se?meli (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 Atat?rk 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 Se?meli (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 Se?meli (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 Se?meli (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

```
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 serach
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
```

O(1)

O(1)

BestCase

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

WorstCase

O(n) zaman alır bütün listeyi dolaşarak bulduğu elemları ekler ekleme işemi ve diğer işelemler O(1) olarak kabul edilmiştir

BestCase

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

WorstCase

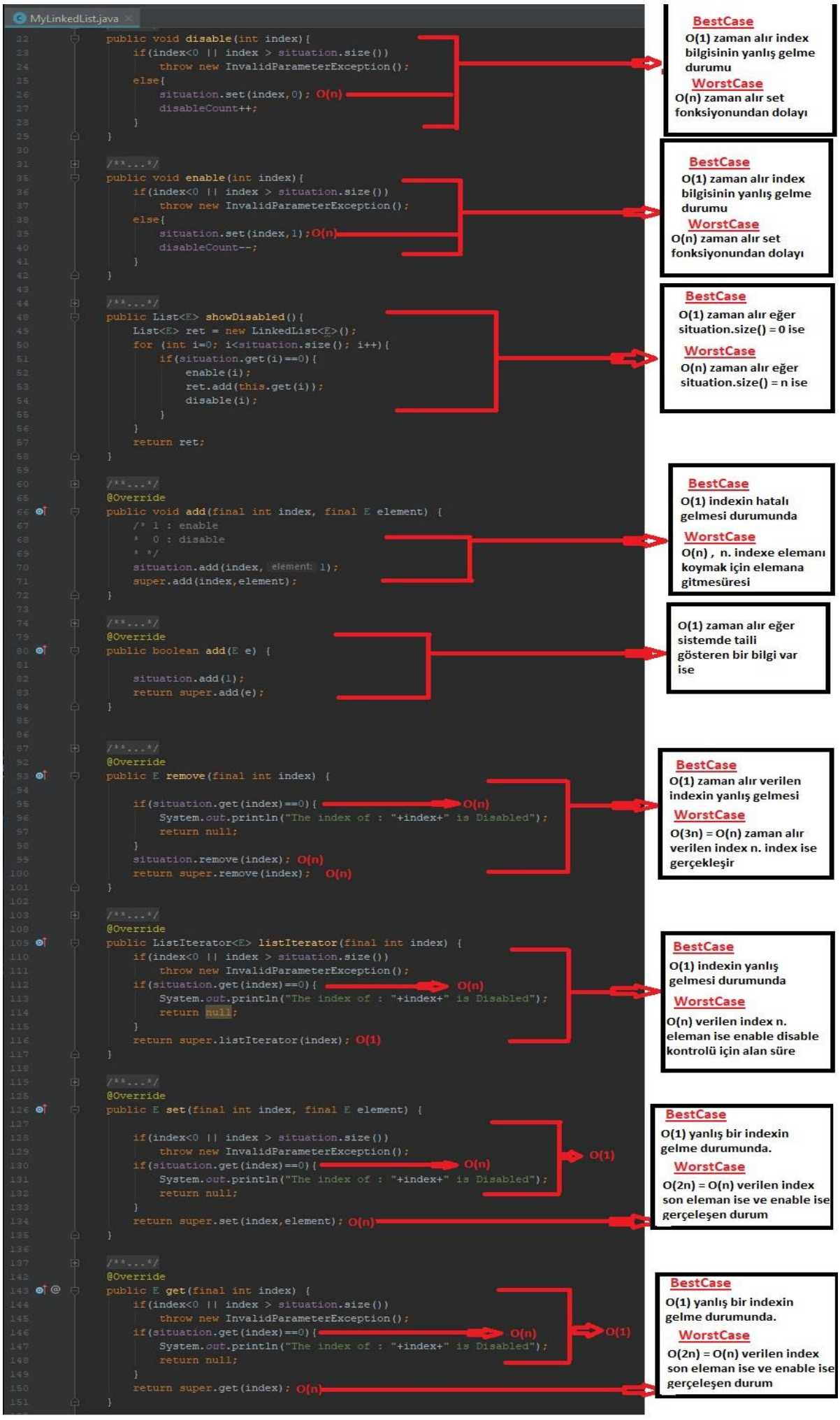
O(n) zaman alır aranılan elemanların listin sonunda olması

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


MyLinkedList.java
public void disable(int index){
if(index<0 || index > situation.size())
throw new InvalidParameterException();
else{
situation.set(index,0); O(n)
disableCount++;
}
}
BestCase
O(1) zaman alır index bilgisinin yanlış gelme durumu
WorstCase
O(n) zaman alır set fonksiyonundan dolayı
public void enable(int index){
if(index<0 || index > situation.size())
throw new InvalidParameterException();
else{
situation.set(index,1);O(n)
disableCount--;
}
}
BestCase
O(1) zaman alır index bilgisinin yanlış gelme durumu
WorstCase
O(n) zaman alır set fonksiyonundan dolayı
public List<E> showDisabled(){
List<E> ret = new LinkedList<E>();
for (int i=0; i<situation.size(); i++){
if(situation.get(i)==0){
enable(i);
ret.add(this.get(i));
disable(i);
}
}
return ret;
}
BestCase
O(1) zaman alır eğer situation.size() = 0 ise
WorstCase
O(n) zaman alır eğer situation.size() = n ise
@Override
public void add(final int index, final E element) {
/* 1 : enable
* 0 : disable
* */
situation.add(index, element: 1);
super.add(index,element);
}
BestCase
O(1) indexin hatalı gelmesi durumunda
WorstCase
O(n) , n. indexe elemanı koymak için elemana gitmesüresi
@Override
public boolean add(E e) {
situation.add(1);
return super.add(e);
}
O(1) zaman alır eğer sistemde taili gösteren bir bilgi var ise
@Override
public E remove(final int index) {
if(situation.get(index)==0){ O(n)
System.out.println("The index of : "+index+" is Disabled");
return null;
}
situation.remove(index); O(n)
return super.remove(index); O(n)
}
BestCase
O(1) zaman alır verilen indexin yanlış gelmesi
WorstCase
O(3n) = O(n) zaman alır verilen index n. index ise gerçekleşir
@Override
public ListIterator<E> listIterator(final int index) {
if(index<0 || index > situation.size())
throw new InvalidParameterException();
if(situation.get(index)==0){ O(n)
System.out.println("The index of : "+index+" is Disabled");
return null;
}
return super.listIterator(index); O(1)
}
BestCase
O(1) indexin yanlış gelmesi durumunda
WorstCase
O(n) verilen index n. eleman ise enable disable kontrolü için alan süre
@Override
public E set(final int index, final E element) {
if(index<0 || index > situation.size())
throw new InvalidParameterException();
if(situation.get(index)==0){ O(n)
System.out.println("The index of : "+index+" is Disabled");
return null;
}
return super.set(index,element); O(n)
}
O(1)
BestCase
O(1) yanlış bir indexin gelme durumunda.
WorstCase
O(2n) = O(n) verilen index son eleman ise ve enable ise gerçeleşen durum
@Override
public E get(final int index) {
if(index<0 || index > situation.size())
throw new InvalidParameterException();
if(situation.get(index)==0){ O(n)
System.out.println("The index of : "+index+" is Disabled");
return null;
}
return super.get(index); O(n)
}
O(1)
BestCase
O(1) yanlış bir indexin gelme durumunda.
WorstCase
O(2n) = O(n) verilen index son eleman ise ve enable ise gerçeleşen durum

CourseList.java

```java
public class Node
{
    private Node next;
    private Course data;
    private int semester;
    /**...*/
    public Node(Course dat)
    {
        data = dat;
        semester = dat.getSemester();
        this.next = null;
    }

    /**...*/
    public Node() {
        this.data = null;
        this.next = null;
    }

    /**...*/
    public Course getData() { return data; }

    /**...*/
    public void setData(Course data) {
        if(data==null)
            throw  new NullPointerException();
        this.data = data;
    }

    /**...*/
    public int getSemester() { return semester; }
}


/**...*/
public boolean add(Course data) {

    if (data == null) {
        return false;
    }
    Node currentNode = new Node(data);
    if (isEmpty()){
        head = currentNode;
        tail=head;


    } else {
        tail.next = currentNode;
        tail=tail.next;
    }
    countNodes++;
    return true;
}


/**...*/
public boolean isEmpty() { return countNodes<=0; }

/**...*/
public int size() { return countNodes; }

/**...*/
public Node get(int index){

    if(index<0)
        throw new IllegalArgumentException();
    if (index>=countNodes)
        throw new IndexOutOfBoundsException();
    Node temp = head;
    for (int i=1; i<=index; i++){
         temp=temp.next;
    }
    return temp;
}

/**...*/
public Node remove(int index){

    if(index<0)
        throw new IllegalArgumentException();
    if (index>=countNodes)
        throw new IndexOutOfBoundsException();

    Node temp = head;
    Node element = null;
    for (int i=1; i<index; i++){
        temp=temp.next;
    }
    element = temp.next;
    temp.next=temp.next.next;
    countNodes--;
    return element;

}
```

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

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

O(1)

O(n)

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

O(1)

O(n)

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

CourseList.java

```java

    /**...*/
    public CourseListIterator iterator() { return new CourseListIterator(head); }

    /**
     *  The CourseListIterator class
     *  The Node contains Course object
     */
    public class CourseListIterator implements Iterator<Course> {

        private Node nextNode,head,semNext;
        private Node lastReturnedNode;
        private Node previousNode;
        private boolean flag = false;
        /**
         * The Iterator Constructor
         * @param node The Node contains Course object
         */
        public CourseListIterator(Node node){
            this.head=node;
            this.nextNode = node;
            this.semNext=node;
        }

        /**
         * is there a next element on list
         * @return boolean
         */
        public boolean hasNext ()
        {
            if(flag)
                return semNext!=null;
            return nextNode != null;
        }

        /**
         *  next element
         * @return Node
         * @throws NoSuchElementException
         */
        public Course next () throws NoSuchElementException
        {
            flag=false;
            if (!this.hasNext ())
                throw new NoSuchElementException();

            previousNode = lastReturnedNode;
            lastReturnedNode = nextNode;
            nextNode = nextNode.next;
            return lastReturnedNode.data;
        }

        /**
         * Remove the element
         * @throws IllegalStateException
         */
        public void remove() throws IllegalStateException
        {
            if (lastReturnedNode == null){
                throw new IllegalStateException ();
            }
            else{
                lastReturnedNode = lastReturnedNode.next;
                countNodes--;
            }
        }

        /**
         * Move next node in same semester
         * @param semester the semester
         * @return Course
         */
        public Course nextInSemester(int semester){
            flag=true;
            if(semester<1 || semester>8)
                throw new IllegalArgumentException();
            boolean find=false;
            while(!find){
                if(semesterSearch().getSemester()==semester)
                    find=true;
            }
            return  lastReturnedNode.getData();
        }

        /**
         * Helper for nextInSemester(int semester)
         * @return Node
         */
        private Node semesterSearch(){
            if(semNext.next==null)
                semNext.next=head;
            previousNode = lastReturnedNode;
            lastReturnedNode = semNext;
            semNext = semNext.next;
            return  lastReturnedNode;
        }
    }
```

O(1)

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

BestCase

O(1) zaman alır semester bilgisinin yanlış gelmesi durumunda

WorstCase

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

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