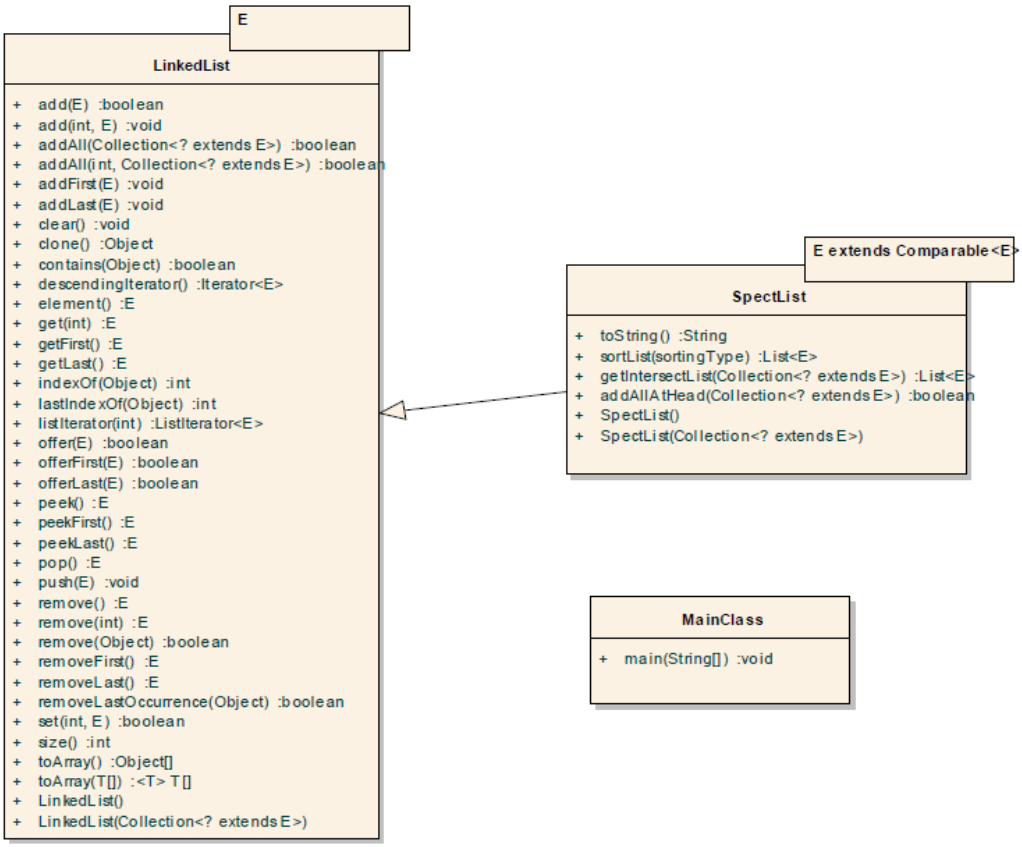
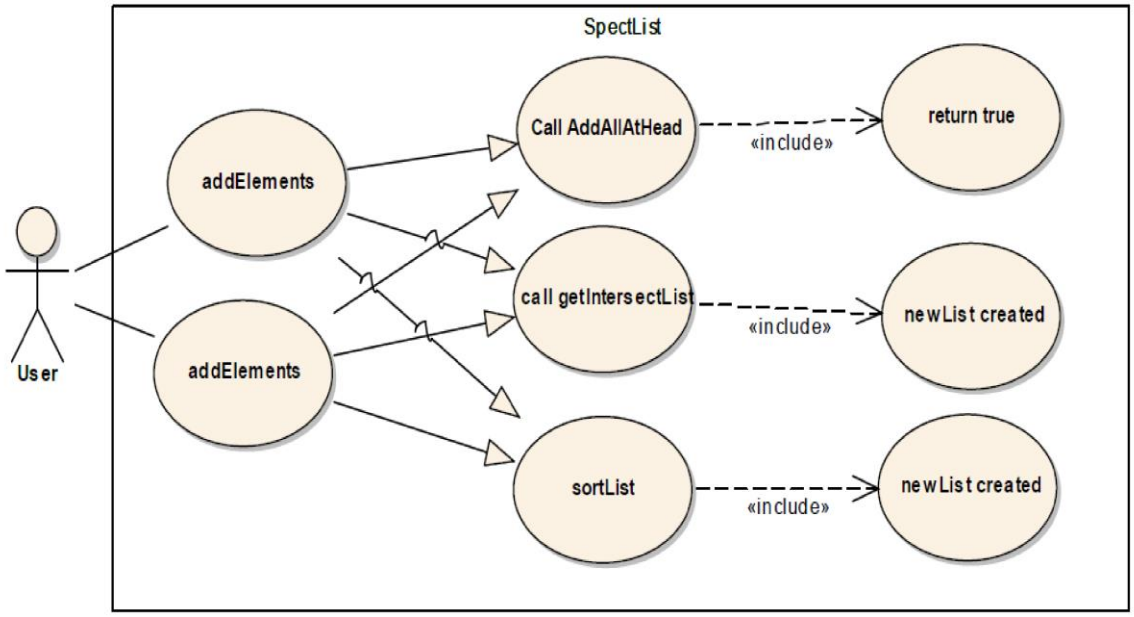
CSE222_HW03
ERCAN UCA 091044011

- ✓ Specific List LinkedList Classından extends olmuştur.
- ✓ SpecificListin ek olarak 3 methodu vardır.
  - ➢ **public boolean addAllAtHead(Collection<? extends E> c);**
    - ❖ Bu metod için 2 tane spectList objesi olmalı s1,s2 gibi mesela,
    - ❖ s1'de 1,3,4,5,5 add ile eklensin. S2'ye de 1,3,4,6,7 eklensin.
    - ❖ Çağırılma şekli **s1.addAllAtHead(s2);** return true olursa.
    - ❖ S1 de artık 1,3,4,6,7,1,3,4,5,5 şeklinde olur farklı durumlarda false ise
    - ❖ Zaten excetion handle edilmiştir metodun içinde.
  - ➢ **public List<E> getIntersectList (Collection<? extends E> c);**
    - ❖ Bu metod iki listeden ortak olanları döndürüyor.
    - ❖ Çağırılma şekli; **s3 = s1. getIntersectList(s2);**
    - ❖ S3 de 1,3,4 şeklinde olur.
  - ➢ public List<E> sortList(int sorting);
    - ❖ sorting'in tipine göre (1 increasin, 0 decreasing)
    - ❖ Cocktail sort algoritması uygulanmıştır.
    - ❖ S1 de "ali","seda","can", "bursa","seda" olsun ve sorting:0 olsun.
    - ❖ Çağırılma şekli **s2=s1.sortList(0);**
    - ❖ S2 de "seda","can","bursa","ali" olur.
- ✓ **Class Diagramı**



✓ **Use Case Diagrams**



✓ **Complexity of Metods**

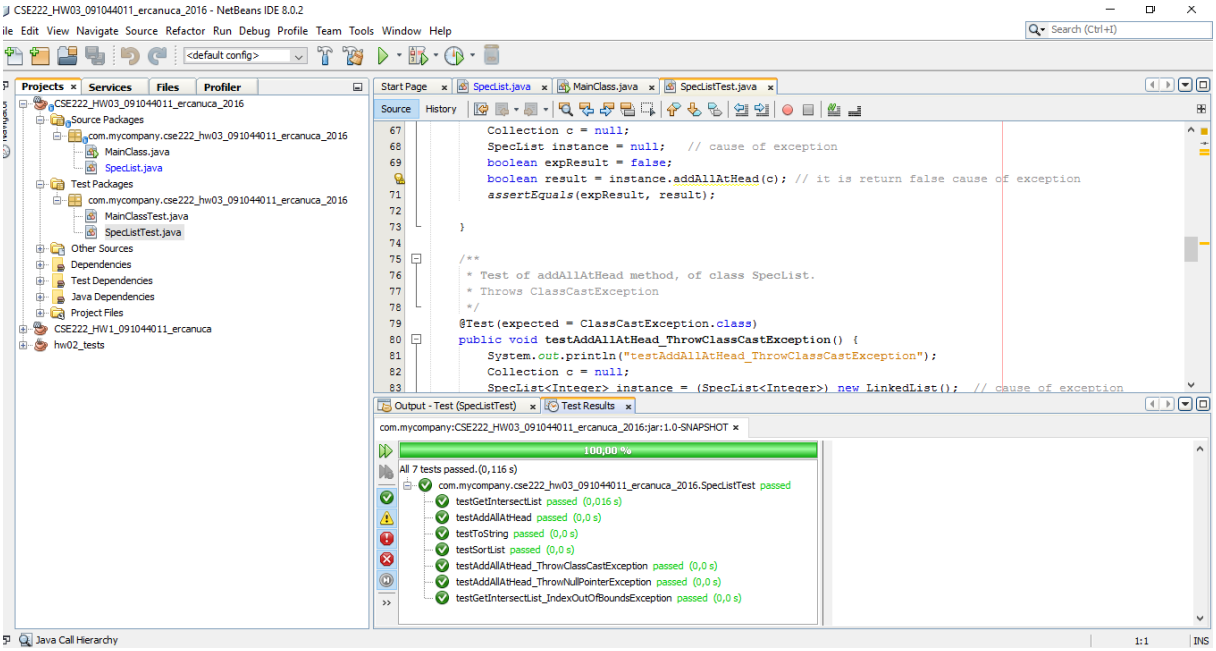
```
69  public List<E> getIntersectList (Collection<? extends E> c){
70      LinkedList<E> intersectionlist = (LinkedList<E>) c;
71      LinkedList<E> returnList = new LinkedList<>();
72
73      int size;
74      // select bigger size
75      if(intersectionlist.size() > size())
76          size = intersectionlist.size();
77      else
78          size = size();
79      // Handling for contains and add throws
80      try{
81          for (int i=0; i< size ; i++)
82          {
83           if(contains(intersectionlist.get(i)) && !returnList.contains(intersectionlist.get(i)))
84              returnList.add(intersectionlist.get(i));
85
86          }
87        }catch(NullPointerException     |
88               ClassCastException       |
89               IllegalStateException    |
90               IllegalArgumentException |
91               UnsupportedOperationException exp){
92          exp.printStackTrace(System.err);
93          return null;
94        }
95      return returnList;
96  }
```

O(1) O(1) T(n)= O(n³)
O(1)
O(1)
O(n³)
O(n²)
O(n) O(n)
O(1)
O(1)

```java
public List<E> sortList(int sorting){
    if (sorting == 1) { // increasing sort
        for (int i = 1; i <= size() / 2; i++) { // phase i of shaker sort
            for (int j = i - 1; j < size() - i; j++) { // first do left to right bubbling pass
                if (this.get(j).compareTo(this.get(j + 1)) > 0) {
                    E temp = get(j);
                    set(j, get(j + 1));
                    set(j + 1, temp);
                }
            }
            for (int j = size() - i; j >= i; j--) { // now do right to left bubbling pass
                if (this.get(j).compareTo(this.get(j - 1)) < 0) {
                    E temp = get(j);
                    set(j, get(j - 1));
                    set(j - 1, temp);
                }
            }
        } // end first for that after if
    } else { // decreasing sort
        for (int i = size() / 2; i >= 1; i--) {  // phase i of shaker sort
            for (int j = size() - i; j >= i; j--) { // first do left to right bubbling pass
                if (this.get(j).compareTo(this.get(j - 1)) > 0) {
                    E temp = get(j);
                    set(j, get(j - 1));
                    set(j - 1, temp);
                }
            }
            for (int j = i - 1; j < size() - i; j++) { // now do right to left bubbling pass
                if (this.get(j).compareTo(this.get(j + 1)) < 0) {
                    E temp = get(j);
                    set(j, get(j + 1));
                    set(j + 1, temp);
                }
            }
        }
    }  // end first for that after else
    return this;
}
```

```java
public boolean addAllAtHead(Collection<? extends E> c){
    LinkedList<E> link = (LinkedList<E>) c;
    try{
        for (int i = link.size()-1; i >=0 ; i--) {
            addFirst(link.get(i));
        }
    }
    catch(NullPointerException |
          ClassCastException |
          IllegalStateException |
          IllegalArgumentException exp){
        exp.printStackTrace(System.err);
        return false;
    }
    return true;
}
```

✓ **Tests**



✓ **Main tests**

----------Tester I----------

----------INTEGERS----------

Created specificList Class object type of Integer and add 7 elements like above.

SpecList<Integer> specific = new SpecList<>();

specific.add(1);
specific.add(2);
specific.add(5);
specific.add(4);
specific.add(0);
specific.add(2);
specific.add(3);

Created another specificList Class object type of Integer and add 5 elements like above.

SpecList<Integer> specific2 = new SpecList<>();

specific2.add(0);
specific2.add(10);
specific2.add(20);
specific2.add(30);
specific2.add(40);

Called addAllAtHead like specific2.addAllAtHead(specific); and show return value.

addAllAtHead returned value: true

Called specific2 toString method

SpecList{[1, 2, 5, 4, 0, 2, 3, 0, 10, 20, 30, 40]}

----------DOUBLES----------

Created specificList Class object type of Double and add 7 elements like above.

SpecList<Double> specific3 = new SpecList<>();

specific3.add(1.1);
specific3.add(2.5);
specific3.add(5.6);

```
        specific3.add(4.5);
        specific3.add(0.7);
        specific3.add(2.8);
        specific3.add(3.8);

Created another specificList Class object type of Double and add 5 elements like above.
 SpecList<Double> specific4 = new SpecList<>();
        specific4.add(1.0);
        specific4.add(10.78);
        specific4.add(20.77);
        specific4.add(30.44);
        specific4.add(40.99);

Called addAllAtHead like specific4.addAllAtHead(specific3); and show return value.
addAllAtHead returned value: true
Called specific4 toString method
SpecList{[1.1, 2.5, 5.6, 4.5, 0.7, 2.8, 3.8, 1.0, 10.78, 20.77, 30.44, 40.99]}
----------------------STRINGS--------------------
Created specificList Class object type of String and add 4 elements like above.
SpecList<String> specific5 = new SpecList<>();
        specific5.add("ali");
        specific5.add("can");
        specific5.add("sardar");
        specific5.add("kenan");

Created another specificList Class object type of String and add 3 elements like above.
 SpecList<String> specific6 = new SpecList<>();
        specific6.add("elif");
        specific6.add("salih");
        specific6.add("Yusuf");

Called addAllAtHead like specific6.addAllAtHead(specific5); and show return value.
addAllAtHead returned value: true
Called specific6 toString method
SpecList{[ali, can, serdar, kenan, elif, salih, yusuf]}

----------------------Tester I END--------------------


----------------------Tester II--------------------
----------------------INTEGERS--------------------
Created specificList Class object type of Integer and add 7 elements like above.
SpecList<Integer> specific = new SpecList<>();
        specific.add(1);
        specific.add(2);
        specific.add(5);
        specific.add(4);
        specific.add(0);
        specific.add(2);
        specific.add(3);

Created another specificList Class object type of Integer and add 5 elements like above.
 SpecList<Integer> specific2 = new SpecList<>();
```

```
        specific2.add(0);
        specific2.add(1);
        specific2.add(2);
        specific2.add(3);
        specific2.add(4);

Initialzation to getIntersectList the List Class Object.
List<Integer> list1 = (LinkedList<Integer>) specific.getIntersectList(specific2);

Called list1 toString method
[0, 1, 2, 3, 4]
----------------------DOUBLES--------------------
Created specificList Class object type of Double and add 7 elements like above.
SpecList<Double> specific3 = new SpecList<>();
        specific3.add(1.0);
        specific3.add(2.5);
        specific3.add(5.6);
        specific3.add(4.5);
        specific3.add(0.7);
        specific3.add(20.7);
        specific3.add(3.4);

Created another specificList Class object type of Double and add 5 elements like above.
 SpecList<Double> specific4 = new SpecList<>();
        specific4.add(1.0);
        specific4.add(10.78);
        specific4.add(20.7);
        specific4.add(3.4);
        specific4.add(40.99);

Initialzation to getIntersectList the List Class Object.
List<Double> list2 = (LinkedList<Double>) specific4.getIntersectList(specific3);

Called list2 toString method
[1.0, 20.7, 3.4]
----------------------STRINGS--------------------
Created specificList Class object type of String and add 4 elements like above.
SpecList<String> specific5 = new SpecList<>();
        specific5.add("yusuf");
        specific5.add("can");
        specific5.add("elif");
        specific5.add("kenan");

Created another specificList Class object type of String and add 3 elements like above.
 SpecList<String> specific6 = new SpecList<>();
        specific6.add("elif");
        specific6.add("salih");
        specific6.add("Yusuf");

Initialzation to getIntersectList the List Class Object.
List<String> list2 = (LinkedList<String>) specific5.getIntersectList(specific6);

Called list3 toString method
```

[elif, yusuf]

----------Tester II END----------

----------Tester III----------
----------INTEGERS----------
Created specificList Class object type of Integer and add 7 elements like above.

```
SpecList<Integer> specific = new SpecList<>();
      specific.add(1);
      specific.add(2);
      specific.add(5);
      specific.add(4);
      specific.add(0);
      specific.add(2);
      specific.add(3);
```

Called specific toString method
SpecList{[1, 2, 5, 4, 0, 2, 3]}
Called specific sorting with decreasing method
SpecList{[5, 4, 3, 2, 2, 1, 0]}
Created another specificList Class object type of Integer and add 5 elements like above.

```
 SpecList<Integer> specific2 = new SpecList<>();
      specific2.add(55);
      specific2.add(20);
      specific2.add(24);
      specific2.add(37);
      specific2.add(40);
```

Called specific2 toString method
SpecList{[55, 20, 24, 37, 40]}
Called specific2 sorting with increasing method
SpecList{[20, 24, 37, 40, 55]}
----------DOUBLES----------
Created specificList Class object type of Double and add 7 elements like above.

```
SpecList<Double> specific3 = new SpecList<>();
      specific3.add(1.1);
      specific3.add(2.5);
      specific3.add(5.6);
      specific3.add(4.5);
      specific3.add(0.7);
      specific3.add(2.8);
      specific3.add(3.8);
```

Called specific3 toString method
SpecList{[1.1, 2.5, 5.6, 4.5, 0.7, 2.8, 3.8]}
Called specific3 sorting with increasing method
SpecList{[0.7, 1.1, 2.5, 2.8, 3.8, 4.5, 5.6]}
Created another specificList Class object type of Double and add 5 elements like above.

```
 SpecList<Double> specific4 = new SpecList<>();
      specific4.add(1.0);
      specific4.add(10.78);
      specific4.add(20.77);
```

```
        specific4.add(30.44);
        specific4.add(40.99);

    Called specific4 toString method
    SpecList{[1.0, 10.78, 20.77, 30.44, 40.99]}
    Called specific4 sorting with decreasing method
    SpecList{[40.99, 30.44, 20.77, 10.78, 1.0]}
    ----------STRINGS----------
    Created specificList Class object type of String and add 4 elements like above.
    SpecList<String> specific5 = new SpecList<>();
        specific5.add("ali");
        specific5.add("can");
        specific5.add("sardar");
        specific5.add("kenan");

    Called specific5 toString method
    SpecList{[ali, can, serdar, kenan]}
    Called specific5 sorting with decreasing method
    SpecList{[serdar, kenan, can, ali]}
    Created another specificList Class object type of String and add 4 elements like above.
     SpecList<String> specific6 = new SpecList<>();
        specific6.add("elif");
        specific6.add("zeynep");
        specific6.add("salih");
        specific6.add("Yusuf");

    Called specific6 toString method
    SpecList{[elif, zeynep, salih, yusuf]}
    Called specific6 sorting with increasing method
    SpecList{[elif, salih, yusuf, zeynep]}

    ----------Tester III END----------
```

- ✓ **Ödev github linki**
  https://github.com/erccanuca/cse222_hw03.git