**GIT Department of Computer Engineering**

**CSE 222/505**

**Spring 2016**

**Homework 03**

**Onur SEZER - 121044074**

**Detailed System Requirements**

Ödev SpecList classı , onun çalistirildiği Main classı ve SpecList classinin JUnit testlerinden oluşur. SpecList class’inda implement edilmiş üç method bulunmaktadır.

* **boolean addAllAtHead(Collection<? extends E> c)**

Parametre olarak aldığı Collection listesini kendi listesinin başına ekler.

c objesini iteratoru yardımıyla tüm elemanlarina ulaşilir ve tek tek index sıfırdan başlayan listIterator(index).add() metodu kullanarak listenin basina c listesi eklenir.

* **List<E> getIntersectList (Collection<? extends E> c)**

Parametre olarak aldığı Collection listesi ile kendi listesinin kesişimini bulur ve liste olarak return eder.

Collection sınıfının contains() metoduna listIterator den gelen E tipli değişkenin olup olmadığına bakar ve eger içinde bulursa oluşturulan listeye eklenir.En son kesişim listesi return edilir.

* **List<E> sortList(decreasing\_or\_increasing)**

Listesini aldiği parametreye göre artarak ya da azalarak sıralar.

Sıralama işlemi coctail sort algoritmasıyla yapıldı.

Coctail sort iteratorle beraber kullanildi.do-while in içindeki birinci while da listin basindan sıralayarak sonuna kadar gelir, diğer while dongusunde ise sonundan basina sıralayarak gelir. İki while dan birisinde swap islemi yapılmazsa distaki do-while in içinden cikar.

**Running Command and Results**

Mainde integer, char, double ve string kullanılarak üç method test edildi.

Main Run edildi:

**------------------------------------------------------------------------**

**Building 121044074 1.0-SNAPSHOT**

**------------------------------------------------------------------------**

**# INTEGER TEST #**

**LinkedList**:

1, 2, 3 ,4 **===**🡺 SpecList in oluşturulan objesine add() methodu ile eklendi.

**Collectin**:

8, 9, 10 **===**🡺 Methodlara parametre olarak gonderilicek

**============================**

**addAllAtHead()**

LinkedListin basina Collection listi eklendi.

8

9 Collection listi, listenin başına eklendi.

10

1

2

3

4

**============================**

**getIntersectList()**

LinkedList ile Collection listinin intersecti bulundu.

8

9 addAllAtHead() metodunda listenin basina eklenen Collection listi,

10 getIntersectList() parametre olarak gönderilince intersect olarak

Collection listinin elemanlarindan oluşan lisiti return etmiş oldu.

**============================**

**sortList()**

LinkedList decreasing olarak siralandi

10

9

8

4

3

2

1

LinkedList increasing olarak siralandi

1

2

3

4

8

9

10

**====================================**

**# CHAR TEST #**

**LinkedList**:

A, B, C, D **===**🡺 SpecList in oluşturulan objesine add() methodu ile eklendi.

**Collectin**:

Z, Y, X **===**🡺 addAllAtHead() ve getIntersectList() methodlarına parametre olarak

gonderilicek

**============================**

**addAllAtHead()**

LinkedListin basina Collection listi eklendi.

Z

Y Collection listi, listenin başına eklendi.

X

A

B

C

D

**============================**

**getIntersectList()**

LinkedList ile Collection listinin intersecti bulundu.

Z

Y

X

**============================**

**sortList()**

LinkedList decreasing olarak siralandi

Z

Y

X

D

C

B

A

LinkedList increasing olarak siralandi

A

B

C

D

X

Y

Z

**====================================**

**# DOUBLE TEST #**

**LinkedList**:

3.2, 5.9, 1.2, 12.003 **===**🡺 SpecList in oluşturulan objesine add() methodu ile eklendi.

**Collectin**:

0.1, 41.1, 11.34 **===**🡺 addAllAtHead() ve getIntersectList() methodlarına parametre

olarak gonderilicek

**============================**

**addAllAtHead()**

LinkedListin basina Collection listi eklendi.

0.1

41.1 Collection listi, listenin başına eklendi.

11.34

3.2

5.9

1.2

12.003

**============================**

**getIntersectList()**

LinkedList ile Collection listinin intersecti bulundu.

0.1

41.1

11.34

**============================**

**sortList()**

LinkedList decreasing olarak siralandi

41.1

12.003

11.34

5.9

3.2

1.2

0.1

LinkedList increasing olarak siralandi

0.1

1.2

3.2

5.9

11.34

12.003

41.1

**====================================**

**# STRİNG TEST #**

**LinkedList**:

onur, sezer, veysel, zehra **===**🡺 SpecList in oluşturulan objesine add() methodu ile eklendi

**Collectin**:

abdil, konya, abdil **===**🡺 addAllAtHead() ve getIntersectList() methodlarına parametre

olarak gonderilicek

**============================**

**addAllAtHead()**

LinkedList basina Collection eklendi.

sude

konya Collection listi, listenin başına eklendi

abdil

onur

sezer

veysel

zehra

**============================**

**getIntersectList()**

LinkedList ile Collection listinin intersecti bulundu.

sude

konya

abdil

**============================**

**sortList()**

LinkedList decreasing olarak siralandi

zehra

veysel

sude

sezer

onur

konya

abdil

LinkedList increasing olarak siralandi

abdil

konya

onur

sezer

sude

veysel

zehra

**------------------------------------------------------------------------**

**BUILD SUCCESS**

**------------------------------------------------------------------------**

**Running Time and Results**

**1 - addAllAtHead**() **Methodu** **Running Time Result**

public boolean addAllAtHead(Collection<? extends E> c)

{

int i = 0;

Iterator<E> iter = (Iterator<E>) c.iterator();

while(iter.hasNext()) { **==**🡺 loop dan **n** gelir

listIterator(i).add(iter.next()); **==**🡺 listIterator(i) index aldığından burdan da **n** gelir

i++;

}

return true;

} ==🡺 ϴ(n2) kadar sürer .

**Proof:**

n2  = O(n2)

n2 <= c.n2

c =1 n>0

n = 1 , c = 1 12  <= 1.12

1 <= 1 (true)

n = k, c = 1 k2  <= 1.k2

k2  <= k2

n = k + 1, c =1 k2 +2k+1 <= k2 +2k+1

0 <= 0 ==🡺 n2  = O(n2) induction metoduna gore dogrudur

n2  = Ω(n2)

n2 >= c.n2

c =1 n>0

n = 1 , c = 1 12  >= 1.12

1 >= 1 (true)

n = k, c = 1 k2  <= 1.k2

k2  >= k2

n = k + 1, c =1 k2 +2k+1 >= k2 +2k+1

0 >= 0 ==🡺 n2  = Ω (n2) induction metoduna gore dogrudur

n2  = O(n2) ve n2  = Ω(n2) doğru olduğundan **n2  = ϴ(n2)** de doğrudur.

**2 - getIntersectList** () **Methodu** **Running Time Result**

public List<E> getIntersectList (Collection<? extends E> c)

{

if(c.size() == 0)

throw new InputMismatchException("Empty list !");

List<E> list = new LinkedList<E>();

ListIterator ite = listIterator();

while(ite.hasNext()) **==**🡺 loop dan **n** gelir

{

E temp = (E) ite.next();

if(c.contains(temp)) **==**🡺 contains() den **n** gelir

list.add(temp); ==🡺 constant time

}

return list;

} ==🡺 **ϴ(n2)** kadar sürer

**Proof:**

n2  = O(n2)

n2 <= c.n2

c =1 n>0

n = 1 , c = 1 12  <= 1.12

1 <= 1 (true)

n = k, c = 1 k2  <= 1.k2

k2  <= k2

n = k + 1, c =1 k2 +2k+1 <= k2 +2k+1

0 <= 0 ==🡺 n2  = O(n2) induction metoduna gore dogrudur

n2  = Ω(n2)

n2 >= c.n2

c =1 n>0

n = 1 , c = 1 12  >= 1.12

1 >= 1 (true)

n = k, c = 1 k2  <= 1.k2

k2  >= k2

n = k + 1, c =1 k2 +2k+1 >= k2 +2k+1

0 >= 0 ==🡺 n2  = Ω (n2) induction metoduna gore dogrudur

n2  = O(n2) ve n2  = Ω(n2) doğru olduğundan **n2  = ϴ(n2)** de doğrudur.

**3 - getIntersectList** () **Methodu** **Running Time Result**

if(status.equals("increasing"){

do { **==**🡺 loop dan **n** gelir (worst)

swap = false;

while (ite1.hasNext()){ **==**🡺 loop dan **n** gelir (worst, best)

E el1 = ite1.next();

ite2 = listIterator(ite1.nextIndex()); **==**🡺 listIterator() dan **n** gelir (worst, best)

if(ite2.hasNext()){

E el2 = ite2.next();

if(el1.compareTo(el2) > 0)

{

E temp = el1;

ite1.set(el2);

ite2.set(temp);

swap = true;

}

}

}

if(!swap)

break;

swap = false;

while(ite1.hasPrevious()) { **==**🡺 loop dan **n** gelir (worst)

E el1 = ite1.previous();

ite2 = listIterator(ite1.nextIndex()); **==**🡺 listIterator() dan **n** gelir (worst)

if(ite2.hasPrevious()){

E el2 = ite2.previous();

if(el1.compareTo(el2) < 0)

{

E temp = el1;

ite1.set(el2);

ite2.set(temp);

swap = true;

}

}

}

}while(swap);

} ==🡺 Best case **O(n2)** kadar sürer, Worst case **O(n3)**

Listin sıralı olması durumunda ilk loopa girer ve condiondaki break’ ten cikar.

**Proof:**

n2  = O(n2)

n2 <= c.n2

c =1 n>0

n = 1 , c = 1 12  <= 1.12

1 <= 1 (true)

n = k, c = 1 k2  <= 1.k2

k2  <= k2

n = k + 1, c =1 k2 +2k+1 <= k2 +2k+1

0 <= 0 ==🡺 n2  = O(n2) induction metoduna gore dogrudur

2n3  = O(n3)

n3 <= c. n3

c =2 n>0

n = 1 , c = 2 13  <= 2.13

1 <= 1 (true)

n = k, c = 2 k3  <= 2.k3

k3  <= 2k3

n = k + 1, c =2 k3 +2k2+ k+1 <= 2k3 +4k2 + 2k+2

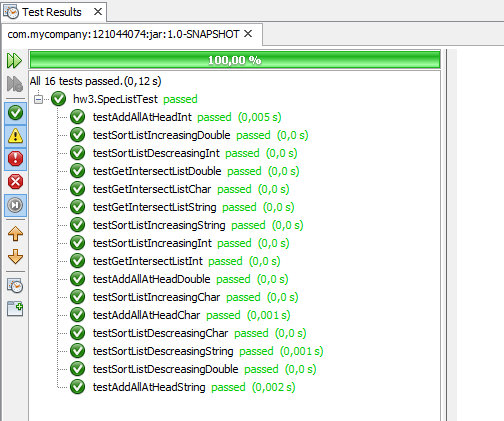
0 <= 2k2+ k+1 ==🡺 2n3  = O(n3) induction metoduna gore

dogrudur

**JUnit Test**

SpecList in methodlarini her biri integer, double, char ve string ile test edildi.

SortList methodunun testini increasing ve decreasing olarak testi yapildi.

****

**Class Diagram**

