Hazal Gönen 131044028
HW05_VERİ YAPILARI RAPOR

İçindekiler

Bu program 3 farklı class icerir bunlar:

1. LinkedListRec
2. ListsOperation
3. TowersOfHanoi

LinkedListRec Class için SISTEM GEREKSINIMLERI:

Bu class remove fonksiyonunu barindirir ve remove(data) şeklinde çağırılır. Listedeki butun elemanlara bakılıp data ya eşit olanlar silinir.Ancak bunu recursive yapmamız istendi.

Ilk olarak LinkedListRec classin objesi oluşturulmali ve daha sonra add metodu ile bu objeye elemanlar eklenmeli

Daha sonra objenin remove fonksiyonu çağırılır ve artık yeni listede remove fonksiyonuna parametre olarak gönderilen eleman hiç kalmayacak şekilde silinir

toString metodu ile liste ekrana basilabilir ve bu metodda dahil olmak üzere butun metodlar recursive yazılmıştır.

LinkedListRec Class için PROBLEM COZUM YONTEMLERI:

Bu classi aslında özelleşmiş bir linkedlist olarak yazdım. İçerisinde Node inner classi bariniyor ve böylece recursive fonksiyonlari next node'u ve head node'u kullanabildiğim için kolaylıkla yazabildim.

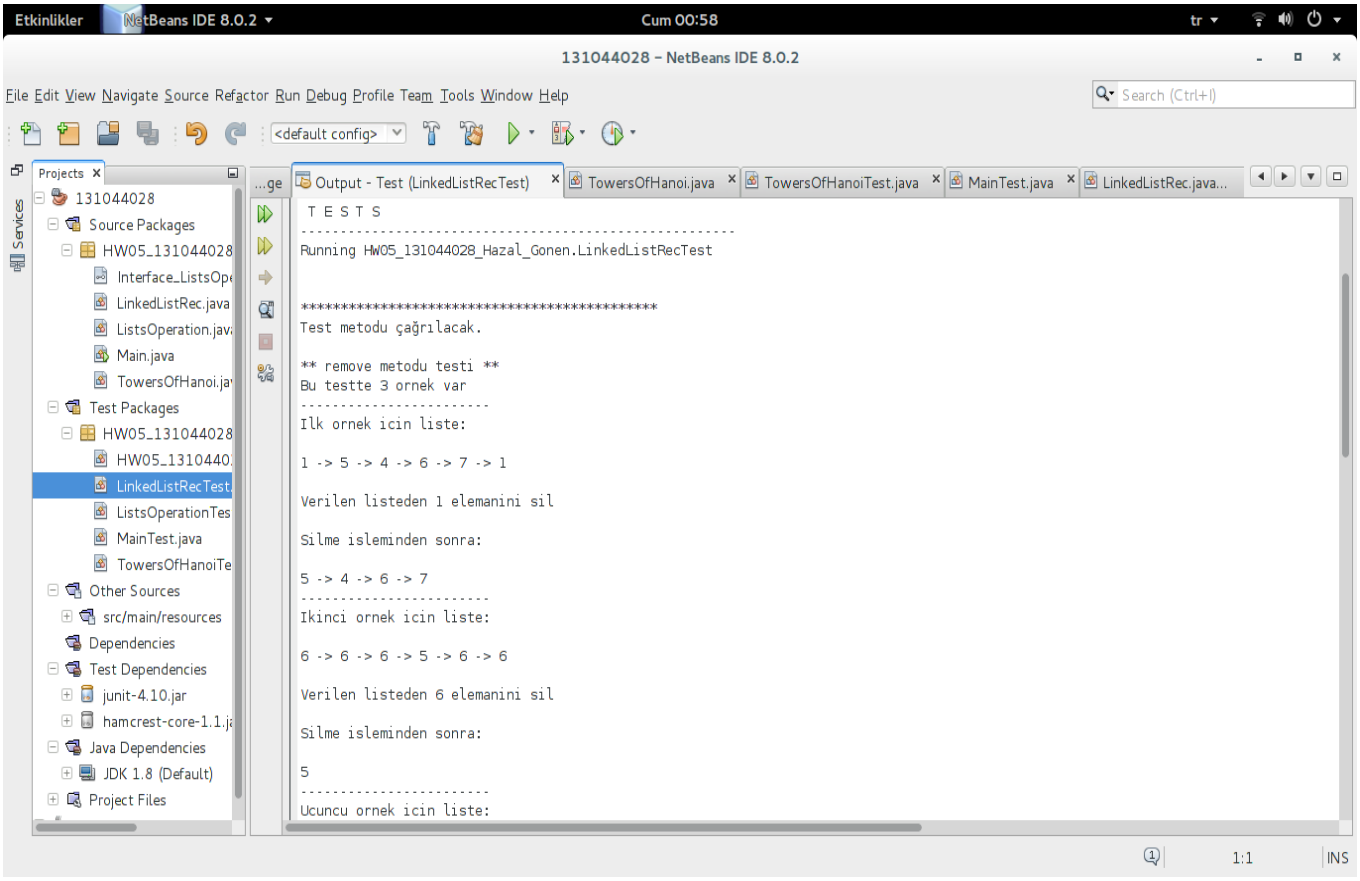
Özelleşmiş linkedlist olduğu için add , remove, size, toString metodlarına sahip.
Bu fonksiyonların wrapperlari hariç kendileri recursive olarak çalışıyor.

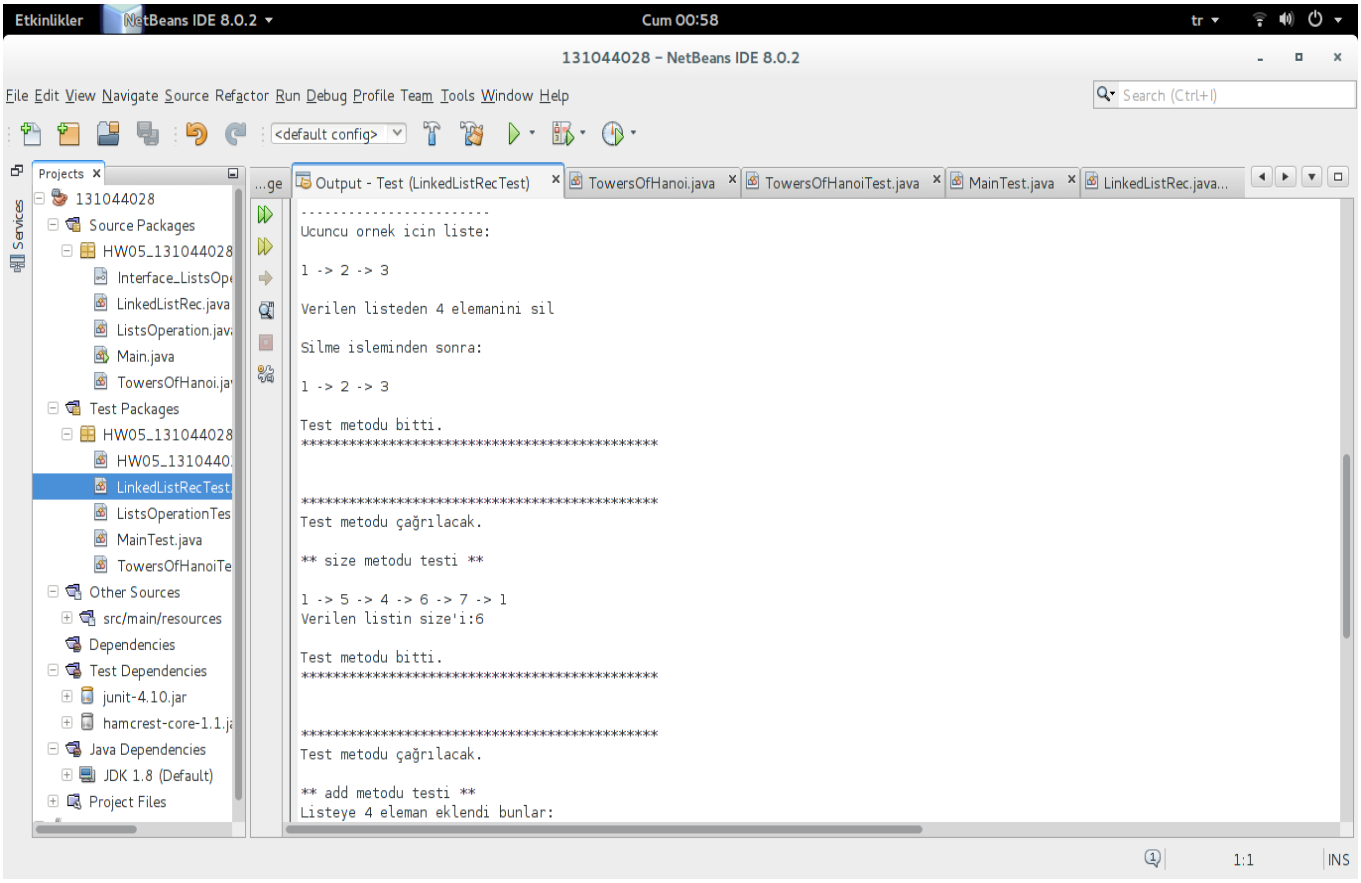
remove yaparken parametre olarak Node head ve Node head.next ini gönderiyorum çünkü kontrol yaparken sonraki elemanlara ulaşabilmem gerekiyor.

ilk yazdıktan sonra şöyle bir problemle karşılaştım eğer ilk indexte silinecek eleman varsa recursive fonksiyonum onu bulamıyor bu sebeple wrapper olarak yazdığım fonksiyonu da hemen recursive çevirdim ve head.data kontrolunu o şekilde çözdüm.

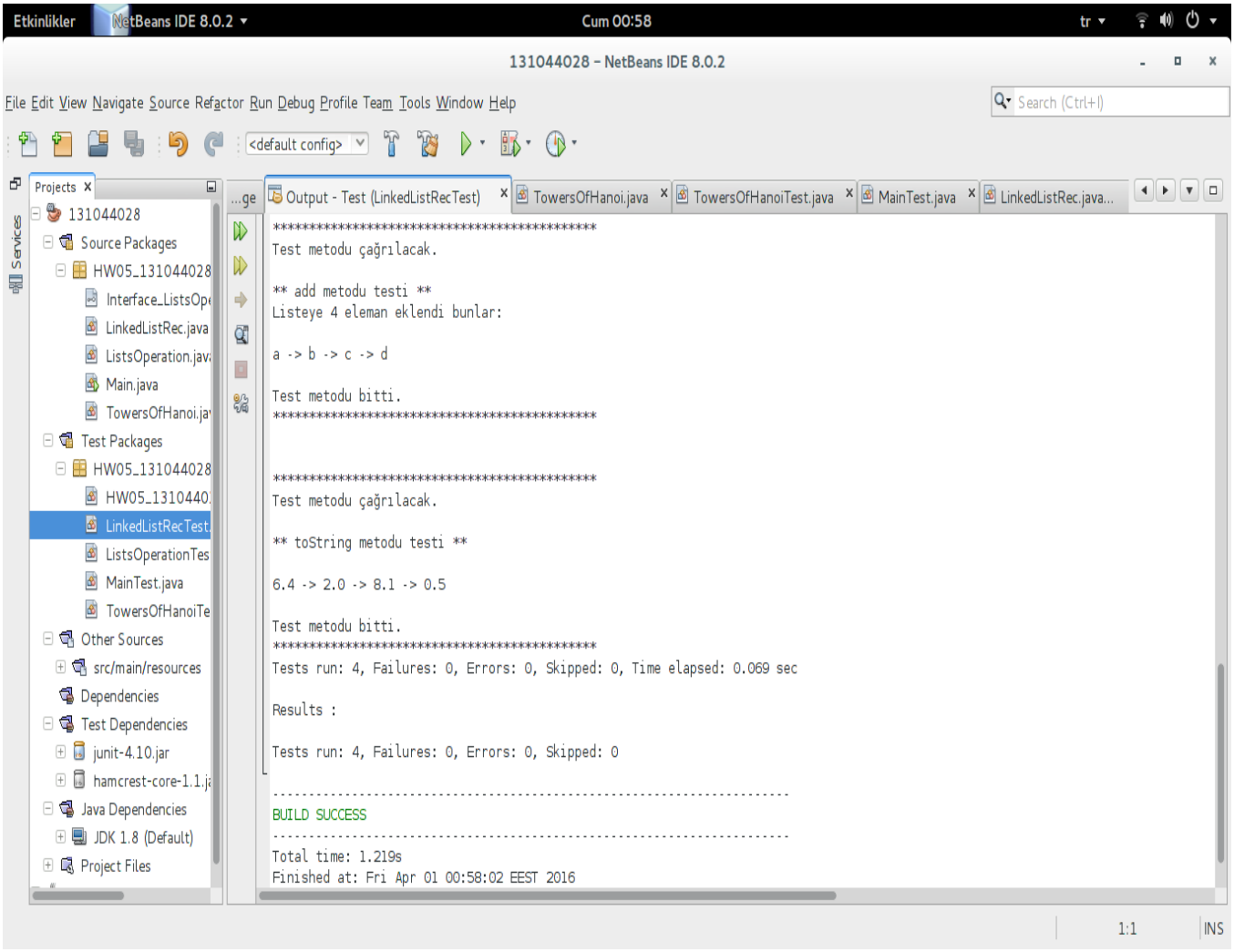
Zamanımı alan diğer bir konu ise ardarda gelen sayıları sileceği zaman biraz sorunlar çıkıyordu ancak onu da recursive yaptığım satırın yerini değiştirerek çözdüm.

LinkedListRec Classinin J-UNIT ekran goruntuleri:




Etkinlikler
NetBeans IDE 8.0.2
Cum 00:58
tr
131044028 – NetBeans IDE 8.0.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
<default config>
Projects
Services
131044028
Source Packages
HW05_131044028
Interface_ListsOp
LinkedListRec.java
ListsOperation.jav
Main.java
TowersOfHanoi.ja
Test Packages
HW05_131044028
HW05_1310440
LinkedListRecTest
ListsOperationTes
MainTest.java
TowersOfHanoiTe
Other Sources
src/main/resources
Dependencies
Test Dependencies
junit-4.10.jar
hamcrest-core-1.1.j
Java Dependencies
JDK 1.8 (Default)
Project Files
...ge
Output - Test (LinkedListRecTest)
TowersOfHanoi.java
TowersOfHanoiTest.java
MainTest.java
LinkedListRec.java...

Ucuncu ornek icin liste:
1 -> 2 -> 3
Verilen listeden 4 elemanini sil
Silme isleminden sonra:
1 -> 2 -> 3
Test metodu bitti.
****
****
Test metodu çağrılacak.
** size metodu testi **
1 -> 5 -> 4 -> 6 -> 7 -> 1
Verilen listin size'i:6
Test metodu bitti.
****
****
Test metodu çağrılacak.
** add metodu testi **
Listeye 4 eleman eklendi bunlar:
1:1
INS


Etkinlikler
NetBeans IDE 8.0.2
Cum 00:58
tr
131044028 – NetBeans IDE 8.0.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
<default config>
Projects
131044028
Source Packages
HW05_131044028
Interface_ListsOp
LinkedListRec.java
ListsOperation.jav
Main.java
TowersOfHanoi.ja
Test Packages
HW05_131044028
HW05_1310440
LinkedListRecTest
ListsOperationTes
MainTest.java
TowersOfHanoiTe
Other Sources
src/main/resources
Dependencies
Test Dependencies
junit-4.10.jar
hamcrest-core-1.1.j
Java Dependencies
JDK 1.8 (Default)
Project Files
Output - Test (LinkedListRecTest)
TowersOfHanoi.java
TowersOfHanoiTest.java
MainTest.java
LinkedListRec.java...
****
Test metodu çağrılacak.
** add metodu testi **
Listeye 4 eleman eklendi bunlar:
a -> b -> c -> d
Test metodu bitti.
****
****
Test metodu çağrılacak.
** toString metodu testi **
6.4 -> 2.0 -> 8.1 -> 0.5
Test metodu bitti.
****
Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.069 sec
Results :
Tests run: 4, Failures: 0, Errors: 0, Skipped: 0
BUILD SUCCESS
Total time: 1.219s
Finished at: Fri Apr 01 00:58:02 EEST 2016
1:1
INS

ListsOperation Class için SISTEM GEREKSINIMLERI:

Bu class 3 onemli metoda sahiptir bunlar intersection union ve isSubset.

İlk olarak sıralı listeler oluşturulmalıdır ve eger sırasız liste oluşturulursa onlar constructorda sıralanmalıdır.

İki tane liste oluşturduktan sonra bu classin constructorına parametre olarak bu listeler gonderilir. Artık bu listelerle istediğimiz işlemleri yapabiliriz.

Bu bahsettiğim 3 fonksiyonu tek tek butun tipler için deneyebiliriz.

Obje intersectionOfLists metodunu çağırır ve dönen şeyi direk ekrana bastığı zaman iki listenin kesişim kümesi ekrana basılır.

Obje unionOfLists metodunu çağırır ve dönen şeyi direk ekrana bastığı zaman iki listenin birleşim kümesi ekrana basılır

Obje isSubset metodunu çağırır eger true donerse ikinci liste ilk listenin alt kumesidir demek olur. false doner ise alt kume değildir.

Yapılacak işlemler bu kadar.

ListsOperation Class için PROBLEM COZUM YONTEMLERI:

Bu class kesinlikle bu üç metoda sahip olacağı icin bunları bir interface classina aldim ve artık bu 3 kume islemine sahip bir class yapmak istersem bu interfaceden implement edebilir.

Bu class iki tane listeyle çalıştığı için içerisinde private 2 liste var ve constructor direk olarak bu iki listeyi parametre olarak gelen listeler sıralı mı diye bakıp daha sonra içerisinde ki data memberlara butun elemanları addAll metodu kulanarak ekliyor.

Artık elimde çalışabileceğim 2 tane listem var ve içeride kullandığım 3 wrapper metodun da private olarak asıl işi yapan metodları var.

Bu asıl işi yapan private metodlar recursive olarak çalışıyor. intersectionOfListsi yaparken ikinci listeyi sürekli ilerleterek recursive kolu sağladım ve ilk listede ikinci listenin ilk elemanı var mı diye bakmamı sağlayan yardımcı bir isElementOfList metodu yazdım.

Eger eleman ikisinde de varsa ArrayListime elemanı ekliyorum.

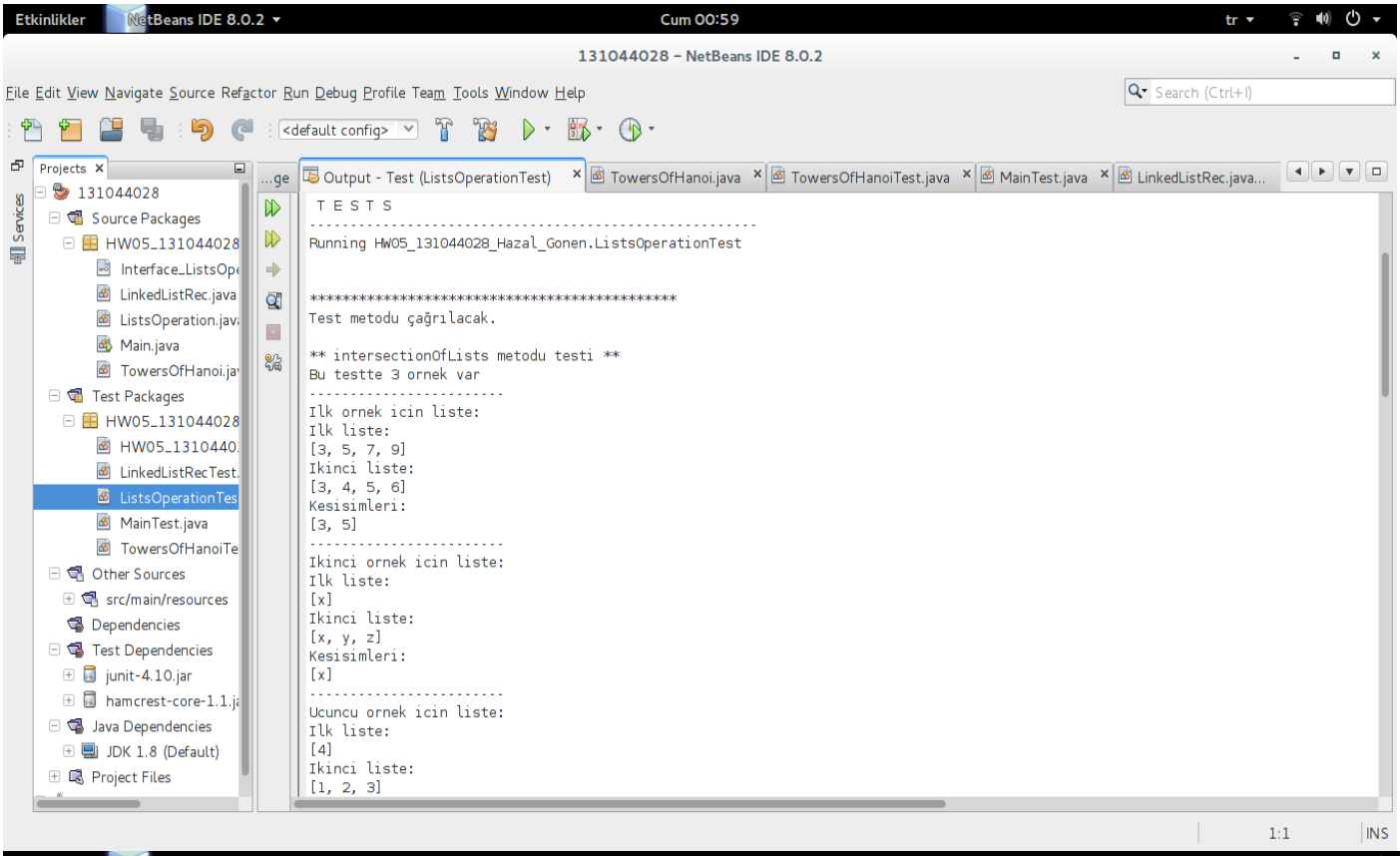
ArrayListi ilerletmek için subList metodunu kullandım ve 1. indexinden itibaren alt kümesini yolluyor her seferinde böylece karşılaştırma işlemini sadece 2.listenin ilk elemanıyla yapmam yetiyor.
recursive'in bitme koşulu ise her seferinde alt kümesini aldığım listenin boş olması.

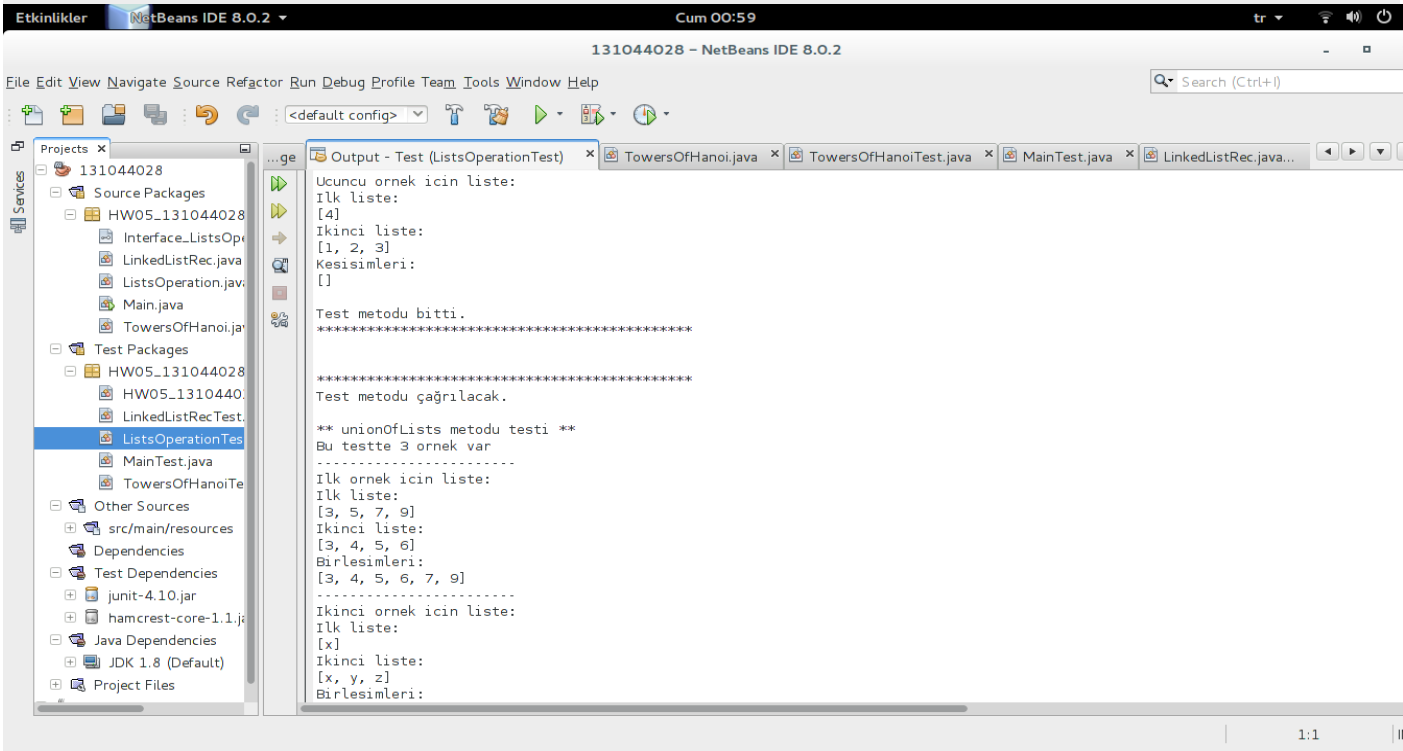
Return edilen listeyi wrapper da sıralayıp öyle return ediyorum.

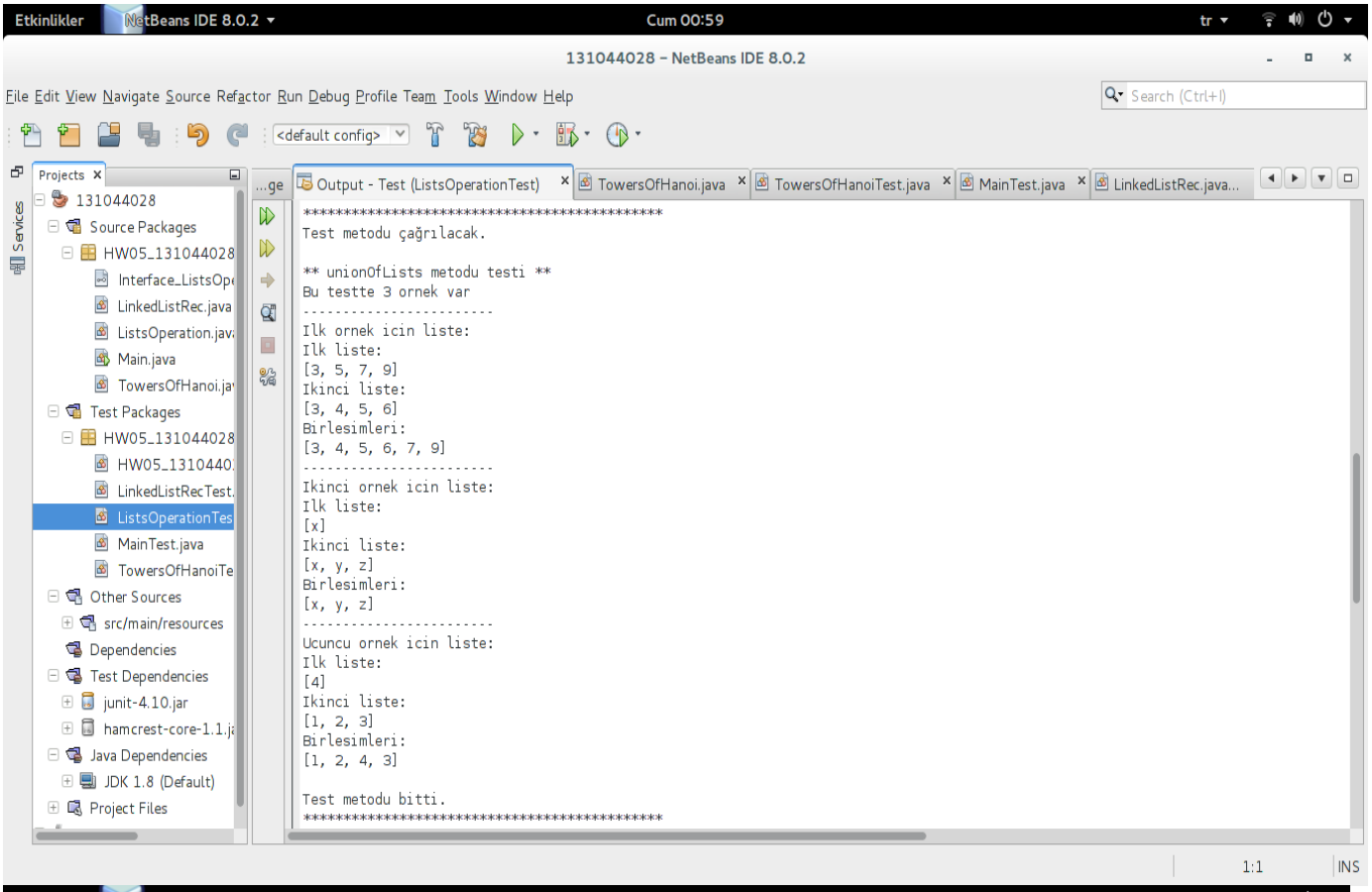
unionOfLists ise birleşimlerini alıyor burda biraz daha zorlandım çünkü tekrar eden elemanlar olucak mı olmıcak mı konusunda kesin birşey söylenmedi aslında küme mantığında tekrar eden eleman olmamalı fakat ben her ihtimale karşı tekrar edenleri de düşünerek biraz uzun bir kod yazdım.

isSubset ise 2.liste 1.listenin alt kumesi mi diye bakıyor.
Bu metodda da yardımcı olarak yazdığım isElementOfList metodunu kullandım.
Eger 2.listenin ilk elamanı 1. listenin içinde varsa yani alt kumesi ise recursive yapıyor alt kume degilse direk false return ederek çıkıyor.

ListOperation Classinin J-UNIT ekran goruntuleri:



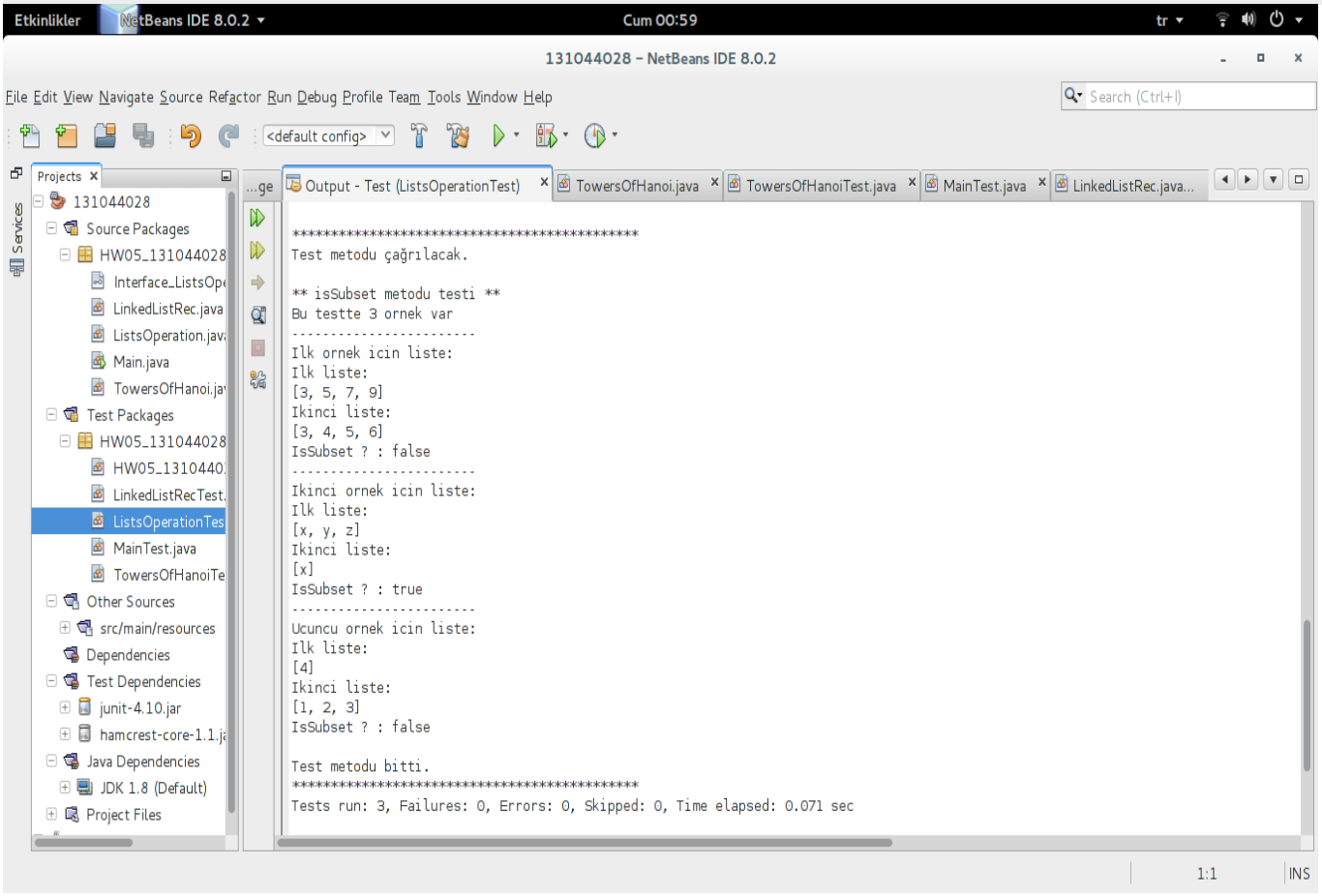



Etkinlikler
NetBeans IDE 8.0.2
Cum 00:59
tr
131044028 – NetBeans IDE 8.0.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
<default config>
Projects
131044028
Source Packages
HW05_131044028
Interface_ListsOp
LinkedListRec.java
ListsOperation.jav
Main.java
TowersOfHanoi.ja
Test Packages
HW05_131044028
HW05_1310440
LinkedListRecTest.
ListsOperationTes
MainTest.java
TowersOfHanoiTe
Other Sources
src/main/resources
Dependencies
Test Dependencies
junit-4.10.jar
hamcrest-core-1.1.j
Java Dependencies
JDK 1.8 (Default)
Project Files
Services
...ge
Output - Test (ListsOperationTest)
TowersOfHanoi.java
TowersOfHanoiTest.java
MainTest.java
LinkedListRec.java...
****
Test metodu çağrılacak.
** unionOfLists metodu testi **
Bu testte 3 ornek var

Ilk ornek icin liste:
Ilk liste:
[3, 5, 7, 9]
Ikinci liste:
[3, 4, 5, 6]
Birlesimleri:
[3, 4, 5, 6, 7, 9]

Ikinci ornek icin liste:
Ilk liste:
[x]
Ikinci liste:
[x, y, z]
Birlesimleri:
[x, y, z]

Ucuncu ornek icin liste:
Ilk liste:
[4]
Ikinci liste:
[1, 2, 3]
Birlesimleri:
[1, 2, 4, 3]
Test metodu bitti.
****
1:1
INS


Etkinlikler
NetBeans IDE 8.0.2
Cum 00:59
tr
131044028 – NetBeans IDE 8.0.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
<default config>
Projects
131044028
Source Packages
HW05_131044028
Interface_ListsOp
LinkedListRec.java
ListsOperation.jav
Main.java
TowersOfHanoi.ja
Test Packages
HW05_131044028
HW05_1310440
LinkedListRecTest.
ListsOperationTes
MainTest.java
TowersOfHanoiTe
Other Sources
src/main/resources
Dependencies
Test Dependencies
junit-4.10.jar
hamcrest-core-1.1.j
Java Dependencies
JDK 1.8 (Default)
Project Files
Services
...ge
Output - Test (ListsOperationTest)
TowersOfHanoi.java
TowersOfHanoiTest.java
MainTest.java
LinkedListRec.java...
****
Test metodu çağrılacak.
** isSubset metodu testi **
Bu testte 3 ornek var

Ilk ornek icin liste:
Ilk liste:
[3, 5, 7, 9]
Ikinci liste:
[3, 4, 5, 6]
IsSubset ? : false

Ikinci ornek icin liste:
Ilk liste:
[x, y, z]
Ikinci liste:
[x]
IsSubset ? : true

Ucuncu ornek icin liste:
Ilk liste:
[4]
Ikinci liste:
[1, 2, 3]
IsSubset ? : false
Test metodu bitti.
****
Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.071 sec
1:1
INS

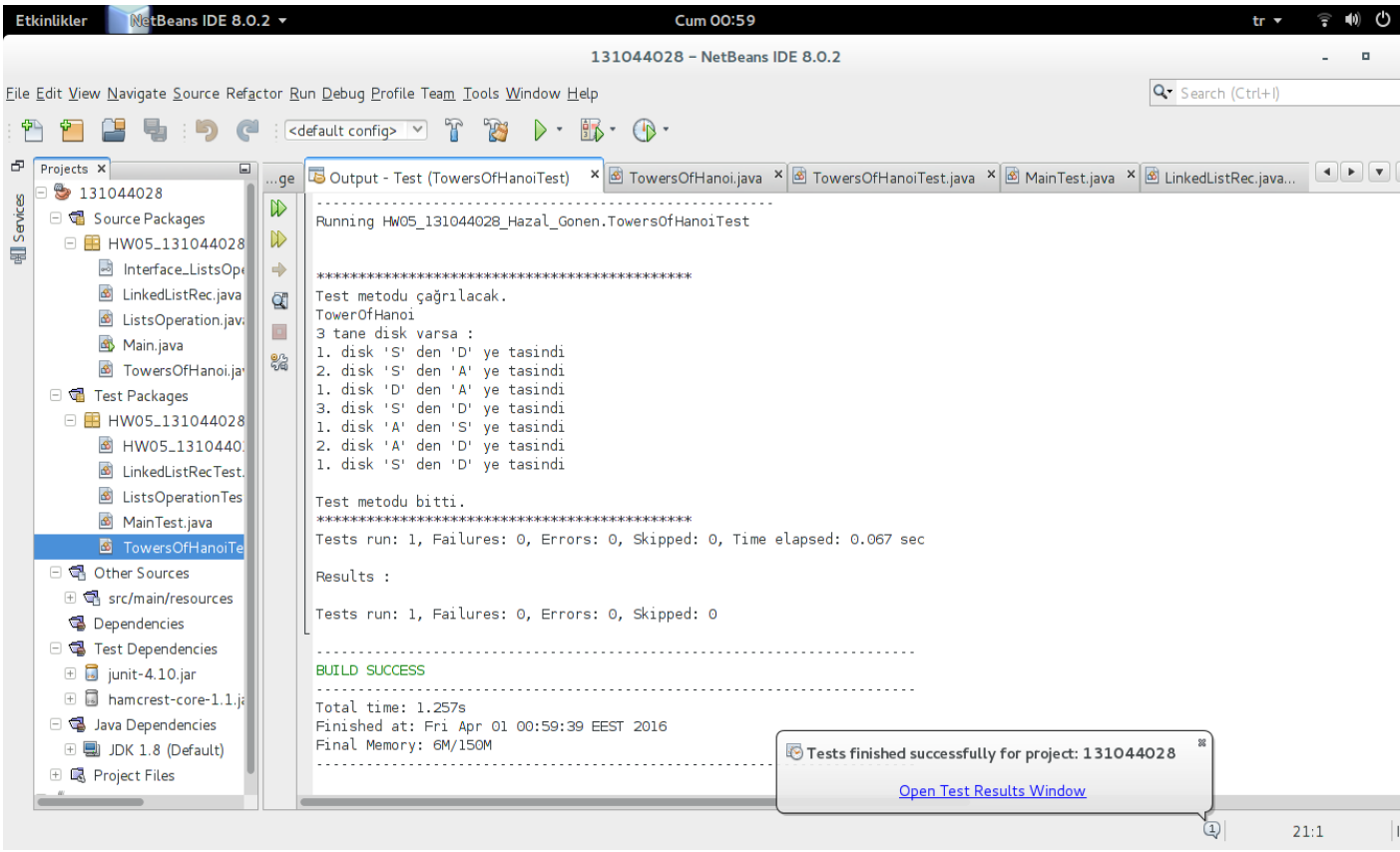
TowersOfHanoi Class için SISTEM GEREKSINIMLERI:

Bu class TowerOfHanoi isimli bir metoda sahiptir ilk olarak 3 tane stack oluşturulmalı ve daha sonra kaç disk istiyorsak o nunla beraber 3 stacki parametre olarak bu fonksiyona göndermeliyiz.
Ve sonuç olarak ekranda kaç numaralı disk hangi kuleden hangi kuleye taşındı bunu gorebiliriz.
Bu metodu recursive yazmak çok kolay ancak iterative yazmak bir hayli zor.

TowersOfHanoi Class için PROBLEM COZUM YONTEMLERI:

Bu classı yazarken stacklerin işime çok yarayacağını dusundum cunku kuleler aynı stack gorevi yapıyor en üste eklenip en ustten cıkartılıyor tek fark alttaki diskin boyutu üstteki diskten daha buyuk olmalı. Butun bunları dusundugumde 3 tane stack olmalı ve disk size'i olmalı fonksiyon iterative olmalı ilk stack'e kaç disk yerleştirilecekse hepsi ters bir şekilde yerleştirildi daha sonra ilk elemanı alıp yardımcı fonksiyona gonderdim ve yardımcı fonksiyon hangisinin disk boyutu kucukse ona gore yer degistirmeler yapılacagını print yapan fonksiyona bildirdi. Boylece sorunu çözmüş oldum. Çok zor bir parttı internetten epey yardım aldım ve recursive gerçekten çok kolaylaştırırdı bunu görmüş oldum.

TowersOfHanoi Classinin J-UNIT ekran goruntuleri:



MAINDE YAZILAN TEST SENARYOLARI:

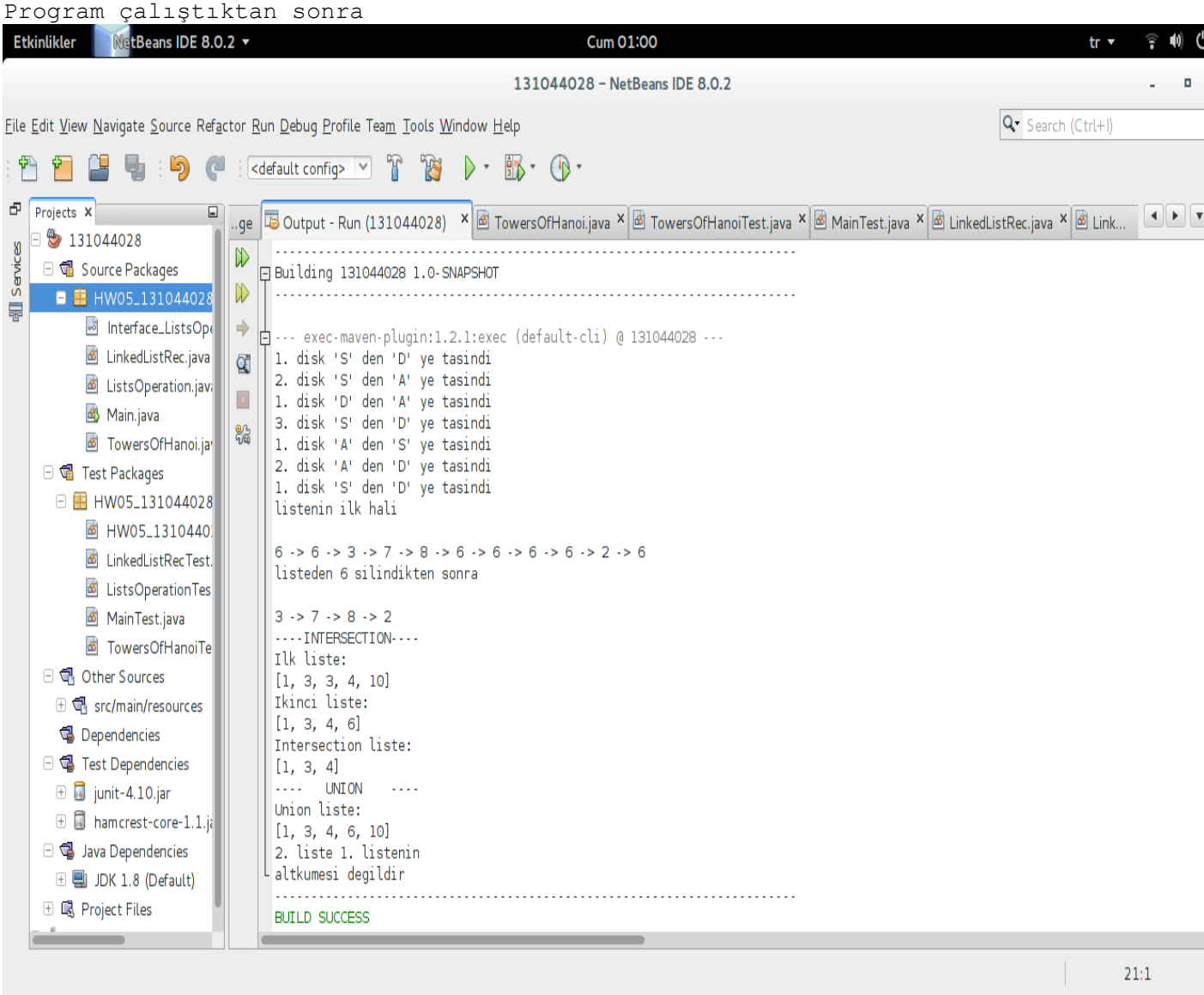
Mainde sıra sıra butun classları test ettim ancak junit testlerde çok ayrıntıya girdiğim için birer ornek yaptım.
İlk olarak 3 diskten oluşan towersOfHanoi 'nin hangi disk nereye taşındı onu ekrana basan classi cagirdim

İkinci olarak LinkedListRec classı oluşturup eleman ekledim daha sonra en çok tekrar eden 6 rakamını sildim ve 7 yerden 6yı sildi.

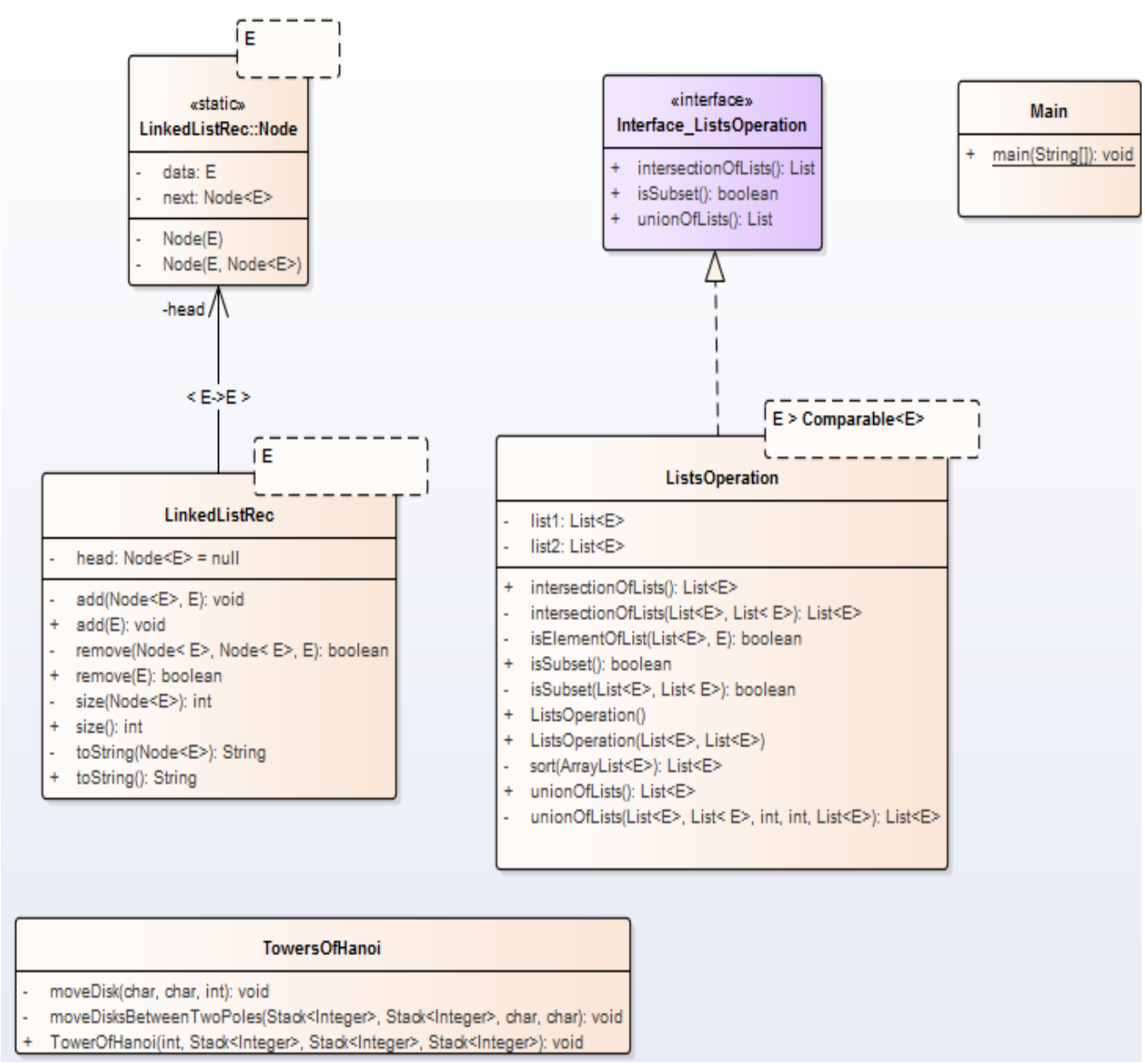
Son olarak 2 tane liste oluşturdum ve ListsOperation classina parametre olarak gonderdim.

sırayla intersection , union ve isSubset islemlerini yapip sonuçları ekranda görüntüledim.

Program çalıştıktan sonra



UML DIYAGRAMLARI



JUNIT TEST SONUCU

Butun testler başarılı