Rapor:

Part1:

Towers Of Hanoi metodu parametre olarak 1 tane disklerin sayısı ve 3 tane de stack almaktadır.Bu stacklerden biri kaynak stack diye adlandırdığımız ilk kule,diğeri diskleri değiştirmek için geçici olarak kullandığımız temp disk ve son olarak diğeri hedef stack diye adlandırdığımız üçüncü stack'imizdir. Oyunda üç kule ve farklı boyutlarda disklerden bulunmakta.Bir direkte en küçük disk yukarıda olacak şekilde, küçükten büyüğe direk üstünde dizilmiş olarak başlar.Ve eklenecek diskin altında kendinden küçük bir disk bulunmaz.Oyunun kuralları şu şekildedir:

Oyunun kuralları

- Her hamlede sadece bir disk taşınabilir.
- Her hamle en üstteki diski direkten alıp diğer bir direğe taşımaktan oluşur. Diğer direkte daha önceden diskler olabilir.
- Hiçbir disk kendisinden küçük bir diskin üzerine koyulamaz.
- Kaynak: https://tr.wikipedia.org/wiki/Hanoi_kuleleri

Oyuna başlamak için diskSize ı parametre olarak gönderince ilk olarak bu parametre ile gelen size kadar sayıyı ilk kulem olan kaynak kuleye dolduruyorum.En az kaç hamle olacağını bulmak için 2 üzeri n -1 ile hamle sayısını buluyorum.

Örnek:En kısa çözümler:

3 disk = 7 hareket

4 disk = 15 hareket

5 disk = 31 hareket

6 disk = 63 hareket

7 disk = 127 hareket

8 disk = 255 hareket

For döngüsü içinde hamle sayısı kadar döngüyü sürdürüyorum. Daha sonra disk size çift sayı olma durumunda ihtimaller 1.kuleden 2.kuleye veya tersi,1.kuleden 3.kuleye veya tersi yada 2.kuleden 3.kuleye diye bakıyorum.Eğer disk size tek sayı ise ihtimaller 1.kuleden 3.kuleye veya tersi, 1.kuleden 2.kuleye veya tersi yada 3.kuleden 2.kuleye diye bu sıra ile bakıyorum ve sonuca ulaşıyorum.

Part2:

Remove metodunu yazarken kitabin LinkedListRec sınıfı içerisinde bulunan add ve toString metotlarını direkt olarak kullandım.remove metodunu yazarken gönderilen elemanın linkedlistte tekrar edip etmediğine baktım.Bu sayede eğer gönderilen eleman duplicate değilse remove islemi yapmadan return false ile metodu bitiriyordum.Eğer gönderilen eleaman linkedlistte yoksa hata mesajı,eğer eleman linkedlistte duplicate olarak bulunuyorsa remove işlemi yapmaya o zaman başlıyorum.Önce wrapper metotta ilk elamana bakarak silinmek istenen elemansa silip private olan remove metoduna parametre olarak head.next göndererek çağırıyorum.İlk eleman silinmek istenen eleman değilse remove metoduna parametre olarak head gönderip öyle çağırıyorum.Private remove metodunda ise head null olana kadar silinen elemanı arayıp bulunca siliyorum.Bu metot recursive olarak çalışmaktadır.

Part3:

Sınıfımda bulunan iki tane List'in birleşimlerini,kesişimlerini ve 2.List 1.List'in alt kümesi mi diye bakan metotlar bulunmaktadır.Ödevde listlerin sıralı olarak gelmesi istenmiş.Eğer listler sıralı gelmediyse Constructor da Collections.sort metodu ile listeleri sıralıyorum.İlk olarak birleşimleri bulmak için yeni bir liste oluşturup List1 i içine kopyalıyorum.Sonra bu yeni liste ile List2 yi karşılaştırıp birbirlerinden farklı olan elemanlarıda yeni listeye atıyorum ve yeni listeyi return ediyorum.Sonra kesişimlerini bulmak için List1 in elemanlari ile List2 i nin elamanlarını teker teker karşılaştırıp ayni elamanları yeni bir listede tutup bu yeni listeyi return ediyorum.Son olarak 2.List 1.List'in alt kümesi mi diye bakarken bunun olma şartı olarak List2 nin tüm elamanlarının List1 de olması gerekir diye düşünüp List1 içinde List2 nin tüm elamanlarını aradım.Eğer bir elaman bile olsa

List1 de yoksa false return ettim.bu metotta true döndürmek için tek şans List2 nin tüm elamanlarının List1 de olamasidir.

NOT=>JUNİT TESTLER 3 SINIF İÇİNDE YAPILDI.ANCAK IDE KAYNAKLI SORUN NEDENİYLE CEVAPLARI (% KAÇ I DOĞRU ÇALIŞIYOR)GÖREMEDİM.

Muhammet Tayyip Çankaya

131044054