Gebze Technical University
Computer Engineering

CSE 222
2017 Spring

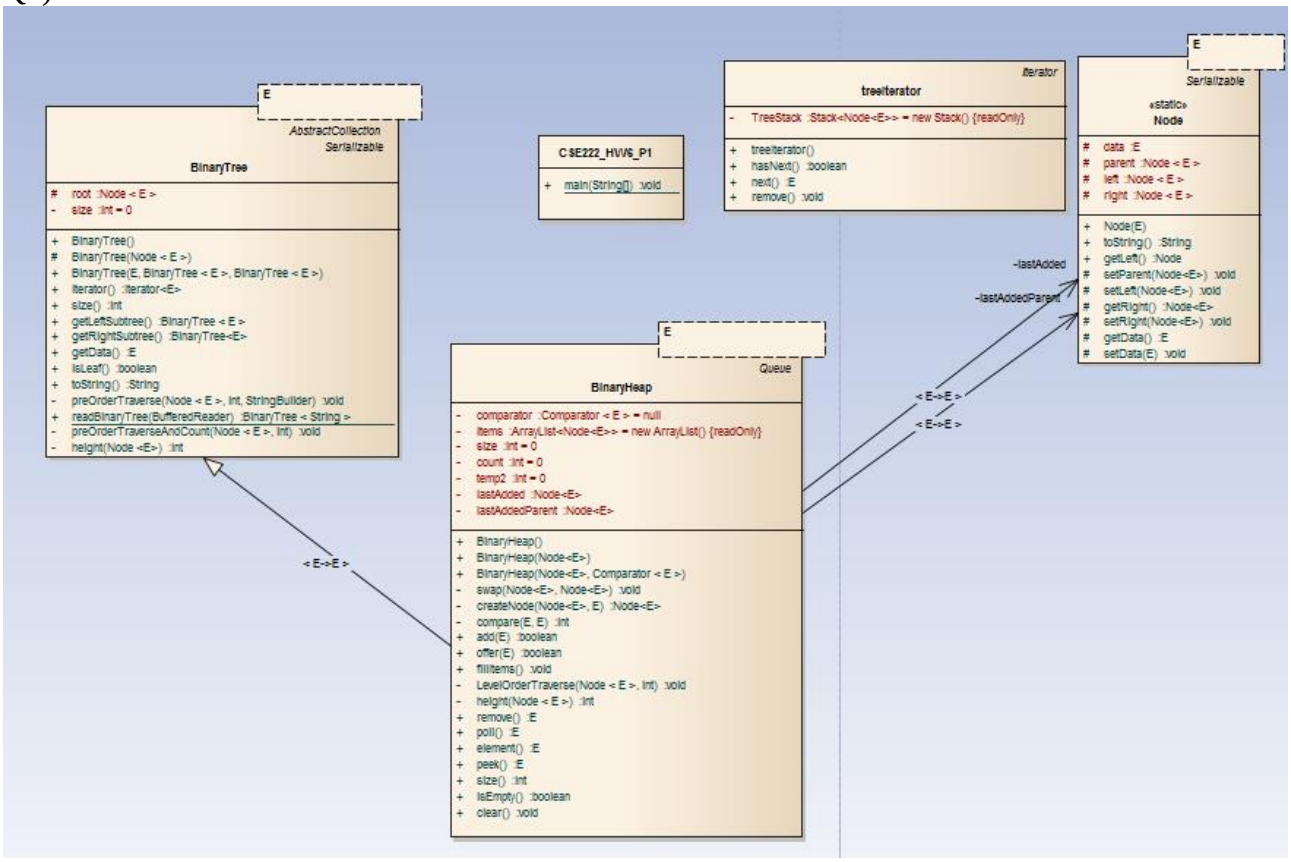
HOMEWORK 6 REPORT

FÜRKAN YILDIZ
141044031

Course Assistant:NUR BANU ALBAYRAK

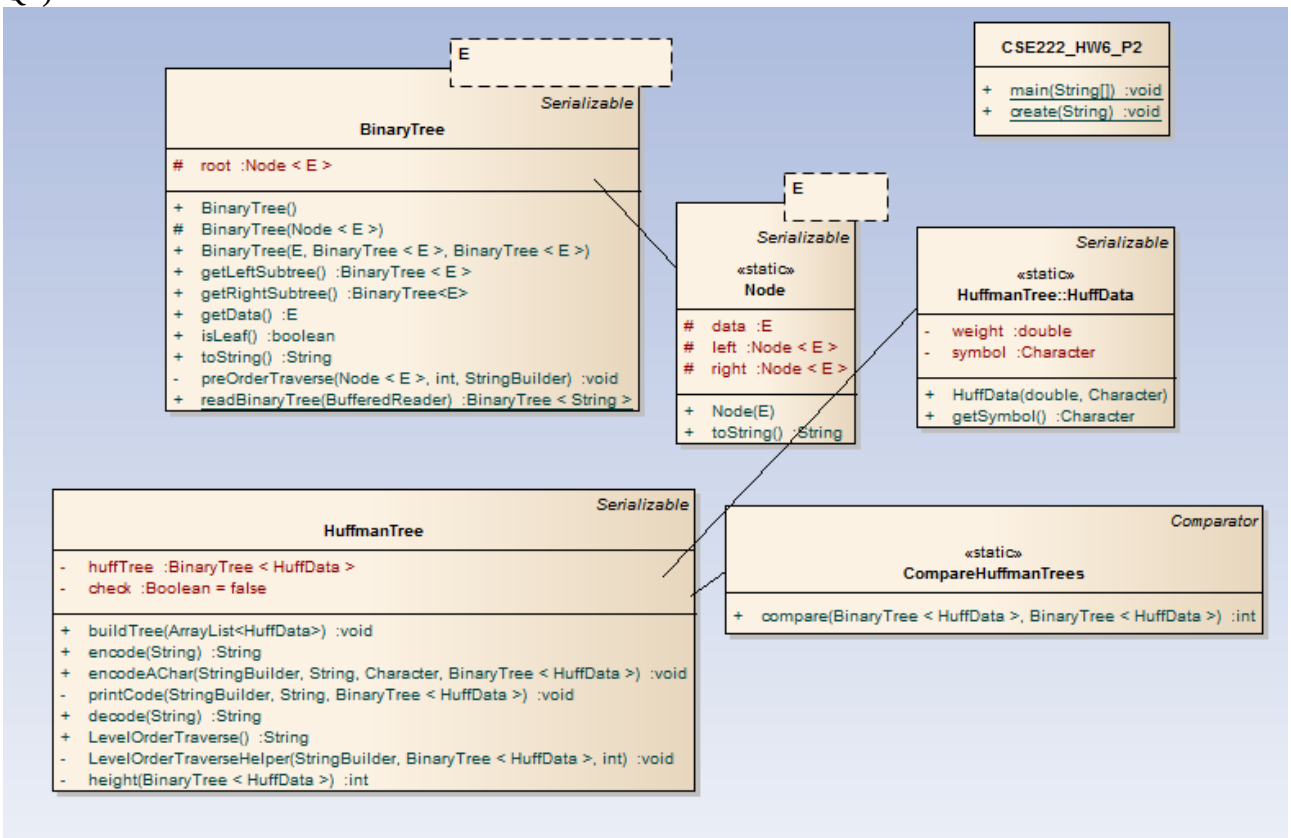
1. Class Diagrams

Q1)

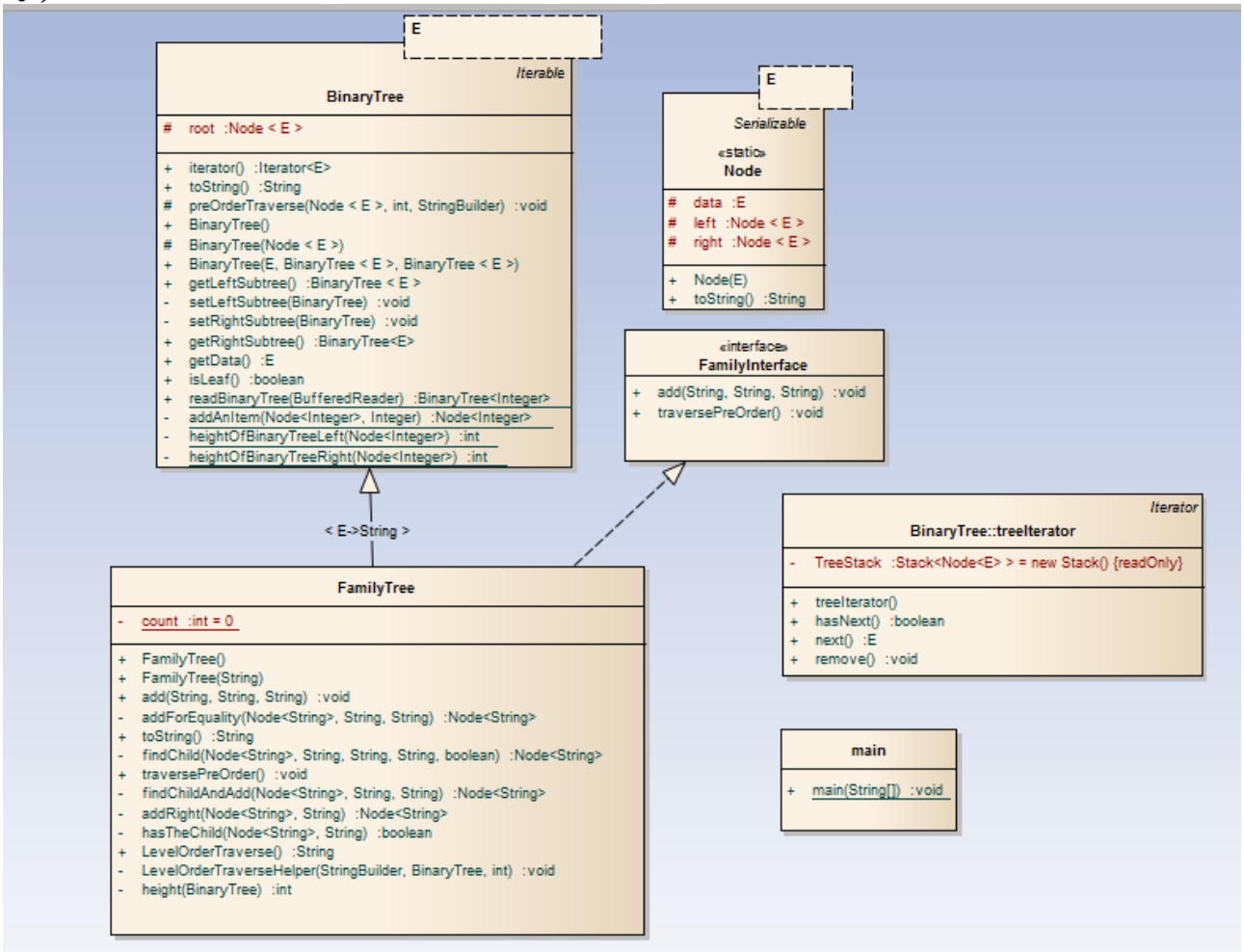


(ayrı olarak eklendi 1. Resim klasöree)

Q2)



Q3)



2. Problem Solutions Approach

Q1)

BinaryHeap'ı implement edebilmek için, kitaptaki BinaryTree kullanıldı ve BinaryTree classından extend, Queue'den implement edildi BinaryHeap. BinaryHeap'ı yazarken Queue interfacesindeki toArray, contains gibi methodları yukarıdan almak için BinaryTree, AbstractCollection'dan extend edildi ve iterator yazıldı BinaryTree için. BinaryHeap class tek bir kalıpta kalmaması için comparator eklendi, Constructurede kullanıcılar parametre olarak comparator verebilecek, böylece her objenin priority si farklı olabilecek. Eğer constructurede verilmez ise "compareTo" methodundan yararlanılacak. BinaryTree'de size hesaplanırken pre-order traverse yapılıp tek tek elemanlar'a bakılıyordu ve bu biraz zahmetli olduğundan dolayı, BinaryHeap'te size methodunun override edilmesine karar verildi. Bunun için bir private size data fieldi eklendi ve size eleman eklendiğinde arttırıldı, çıkartıldığında azaldıltı.

Add methodunda, parametre olarak verilen eleman null ise exception fırtalıyor, değil ise eleman tree'nin concreate kalması sağlanarak (createNode methoduyla) yeni eleman olarak ekleniyor. Bu eklenen yeni eleman data fieldimiz olan lastAdded (son eklenen eleman) ve lastAddedParent (son eklenen elemanın parenti) ile güncelleniyor. Ve ardından en son eklenen eleman ile onun parenti karşılaştırılıyor ta ki bu iki datafiledden biri null olana kadar (en üst root'a çıkmış olabiliriz ve o rootun parenti nulldur) ya da compare methodu true dönderene kadar (yani priority sıralaması bitene kadar).Kısaca özetlemek gerekirse bu methodta yapılan kısaca, verilen eleman concreate treede, concreate yapısını bozmayacak bir şekilde önce ekleniyor, ardından eklenen bu eleman

parentleri ile karşılaştırılıyor ve eklenen bu eleman ağaçta geçmesi gereken yere geçiyor. (Karşılaştırmalar priority için yapılıyor)

Offer, methodunda add'in aynısı yapılıyor.

Remove methodunda, eğer tree boş ise exception fırlatılıyor. Aksi durumda, root çıkartılıp, tree'nin en alt sağındaki elaman root ile yer değiştirip, rootun çocukları ile compare edilerek gerekli yer değiştirmeler yapılmalı. Bunu işlemlerde en sağ alttaki node'yi bulabilmek için, "fillItems" methodu yazıldı, bu methodda, tree'de level order olarak gezilerek, tüm nodeler, items data fieldine ekleniyor, bu eklemeler sonucu, en son eklenen node, bizim treemizdeki en sağ alttaki nodeye yani rootu silince onun yerine geçmesi gereken elemanı temsil ediyor. Bu elemanı bulduktan sonra, bu elamanı, root node'sine yazıyor ve bu elemanı eski olduğu yerden siliyoruz. Böylece silmemiz gereken root'u sildik ve onun yerine geçmesi gereken en sağ alttaki eleman geçmiş oldu. Şimdi yapmamız gereken root ile çocuklarını karşılaştıra karşılaştıra bu rootun bulunması gereken yere, rootu yerleştirmek. Bunu yapabilmek için "while(true) döngüsünde" önce çocuklar arasında bir karşılaştırma yapılıyor yazılan "compare" methodu yardımı ile, ardından bu çocuklardan minimum'u bulununca, root ile karşılaştırılıyor, çocukların en minimumu, roottan da küçük ise o çocuk ile root yer değiştiriyor ve ardından root güncelleniyor. Eğer root, çocuklarından daha küçük ise bu işlem son buluyor, eleman doğru yerine geçmiş oluyor. (küçük olan priority seçildi)

Poll methodu ise removeden farklı olarak sadece tree boş ise exception fırlatmak yerine null return ediyor.

Q2)
Gelen stringi, encode edebilmek için, stringin her elemanı (her karakteri) tek tek encode edilmeli. Bu sebeple verilen bir karakteri encode eden bir recursive method yazıldı "encodeAChar" bu method, tree'nin tüm leaf'lerini kontrol ediyor, eğer aradığımız karakter o leafta ise o leafin kodunu parametre olarak aldığı stringBuildere ekliyor. Kodu bulma işlemini ise yine bir parametre ile yapıyor, parametre başta boş string ile gönderiliyor ve her solda arama yapıldığında stringe "0", sağda arama yapıkdığında ise stringe "1" ekleniyor, böylece bulunan leafin kodu elde ediliyor. Ayrıca eğer aranan karakterin kodu bulundu ise method çağırılmadan önce false olarak atanan "check" datafiledi, true yapılıyor. Böylelikle aranan karakterin Huffman tree'de olup olmadığı kontrol ediliyor. Eğer "encodeAChar" mehodunu çağırdıktan sonra "check" hala false ise, aranan karakter tree'de yoktur, bu durumda ise NoSuchElement exceptionu fırlatılıyor. Bir karakter bu şekilde "encodeAChar" methodu ile şifrelendikten sonra,bir stringe ekleniyor ve encode etmek istediğimiz stringin diğer elemanı "encodeAchar" methoduna yollanıyor, böylece şifrelenmek string tek tek karakterlere ayrılarak şifrelenmiş ve stringe eklenmiş oluyor ve bu string return ediliyor. Eğer tree, şifrelenmek istenen elemanı içermiyor ise exception fırlatılıyor.

Q3)
Ağacı level level traverse edebilmek için önce bir leveldeki tüm elemanları soldan başlayarak kendisine gelen StringBuilder parametresine ekleyen method yazıldı. Bu method bir level parametreside alıyor ve gelen parametreye göre ağacın derinliğine, leveline inerek oradaki tüm nodeleri stringBuildere ekliyor. Levellere inmek için recursive method olarak tekrar aynı fonksiyon level sayısı 1 azaltılarak çağırıldı ve önce sol taraf için çağırıldı böylece istenilen levelde sol taraftan başlayarak eleman stringe yazılıyor. Istenilen level'deki elemanları stringe atmayı başarabildikten sonra, ağacın yüksekliğini öğrendikten sonra tüm yükseklikler için tek tek bu methodu çağırsak problemi çözmüş oluruz. Ağacın yüksekliğini hesaplayabilmek için, "heigh"

methodu yazıldı bu method roottan başlayarak nodeleri geziyor sağ ve sol taraftan sağ veya solda null nodeye kadar ilerliyor ve ardından sağ veya soldan yüksekliği büyük olanı return ediyor. Heihtide bulduktan sonra, her level deki elemanları stringe ekleyen “LevelOrderTraverseHelper” methodumuzu 0’dan başlayarak ağacın yüksekliğine kadar tüm leveller için çağırılarak problem çözüldü.

3. Test Cases

Q1)

14 elemanlı bir integer heap oluşturuldu, bu heap’ten tek tek elemanlar çıkartıldı, her elemanın çıkartılmasında treenin yeni hali ve size’ı ekrana bastırıldı. Eğer tree boşaltıkran sonra pol yapılırsa null, remove yapılrısa exception fırlatır.
Q2)
“freq.txt” isimli bir txt dosyası oluşturuldu bu dosyada alfabenin bazı farkleri ve onların frekansları var (boşluk ‘ ‘ ile ifade edildi) bu dosyadan okuma yapılarak bir Huffman Tree oluşturuluyor ve oluşturulan bu Huffman treeye göre “furkan yildiz gtu” stringi encode methoduna veriliyor ve “1100110000100101100001110100111111100011011010110101110110110000101111110000111010000001” şeklinde bir şifre elde ediliyor, Huffman tree’de kontol ettiğimizde veya decode ile bu şifreyi çözmek istediğimizde doğru bir şekilde çözülmüş halini “furkan yildiz “gtu” stringini alıyoruz.

Eğer dosyada olmayan bir değer girersek mesela dosyada ç,ş,ı gibi Türkçe karakterler yokken bu karakterleri içeren bir stringi encode etmeye çalışırsak örneğin “çarşamba” gibi program bize girdiğimiz karakterin tree’de yer almadığına yönelik bir hata verecektir.

Q3)
“test.txt” adında bir file oluşturuldu, bu file da, bir ailenin akrabalık bağları yer alıyor. Bu file’ya göre Family Tree oluşturuldu ve oluşturulan bu family tree, “LevelOrderTraverse” methodu çağırılarak test edildi.