Gebze Technical University
Computer Engineering

CSE 222
2017 Spring

HOMEWORK 05 REPORT

Yasin Açıkgöz
131044070

Course Assistant: Nurbanu Albayrak
Not: Bu ödevde 2 gün geç yükleme hakkı kullanılmıştır

1. Class Diagrams

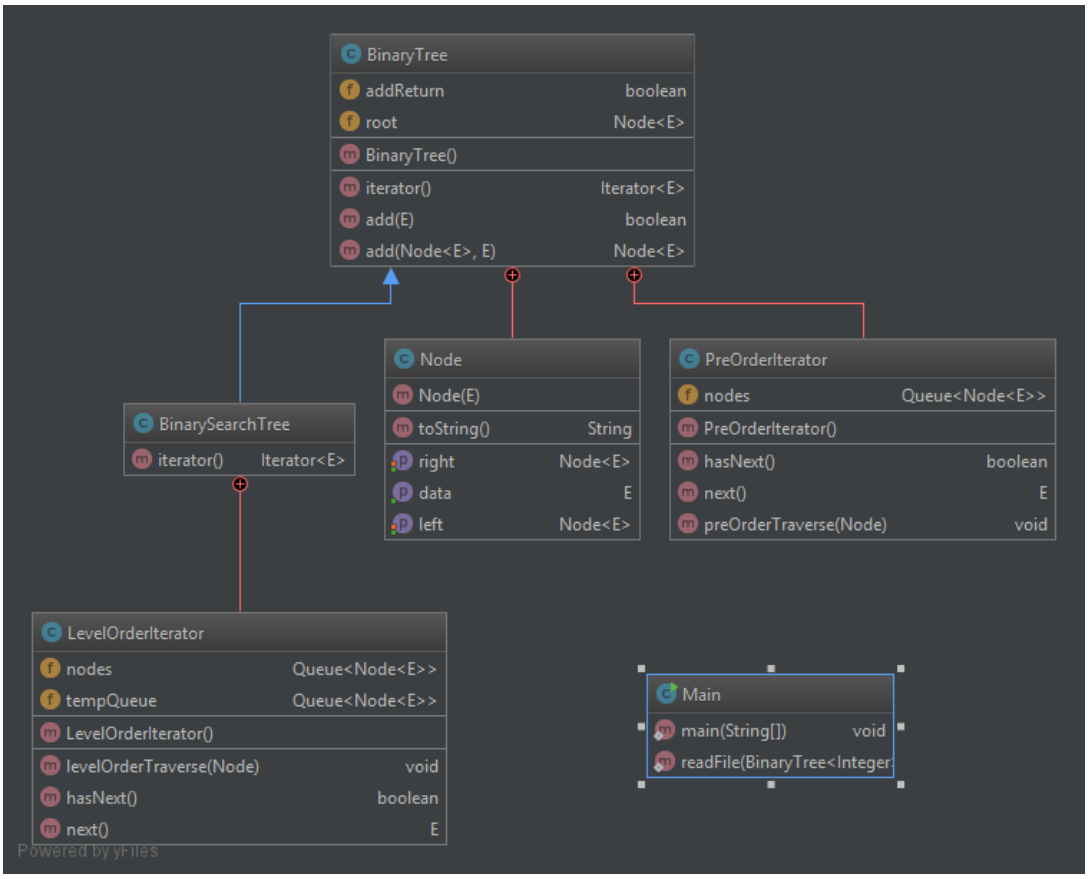

Figure 1 Class Diagram for Part1

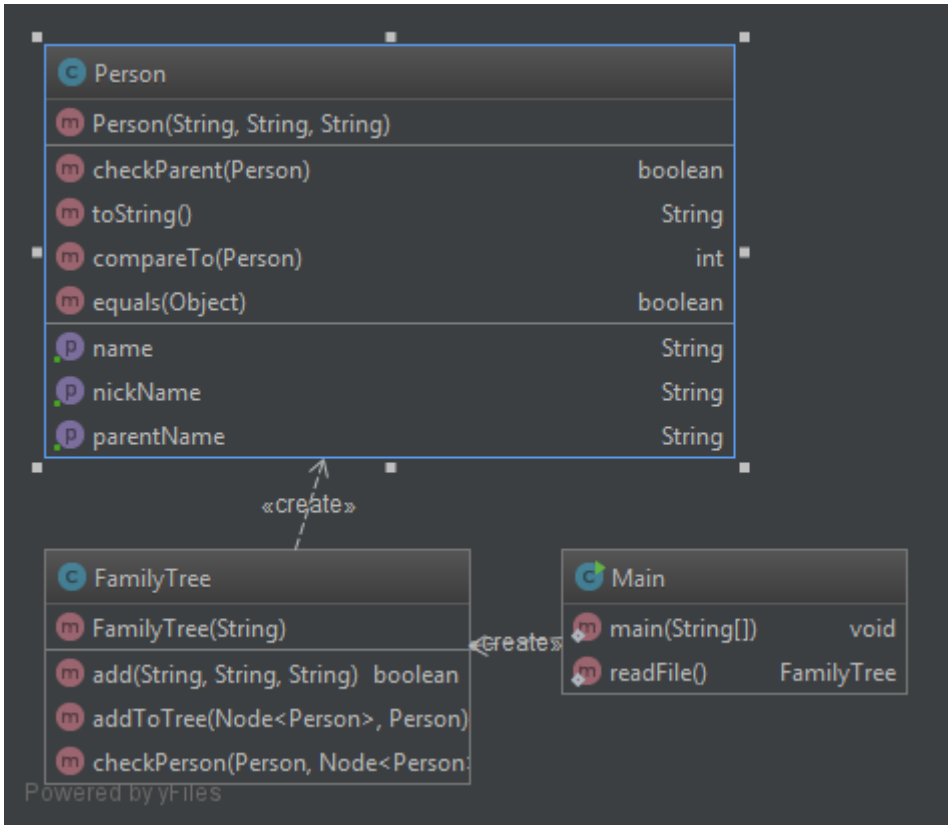

Figure 2 Class Diagram for Part2

2. Problem Solutions Approach

- **Part 1 – Pre Order Iterator**
  Ödevin birinci partının ilk kısmındaki pre order traverse problemini çözmek için ilk olarak kitaptaki pre order traverse algoritmasını inceledim. Ardından kitaptan yararlanarak oluşturduğum BinaryTree class'ının içerisinde **inner class olarak PreOrderIterator** sınıfı oluşturdum. Bu sınıfın içerisinde tree elemanlarını tutmak için bir queue oluşturdum. No parameter constructor'ında BinaryTree class'ında bulunan Node class'ının root data member'ını parametre olarak alan bir preOrderTraverse metodu yazdım. Bu metod aşağıdaki recursive mantıkla çalışmaktadır. Root node'u queue'ya koyar, ardından node'un sol alt ağacı ve sağ alt ağacı için ayrı ayrı recursive çağrı yapar.

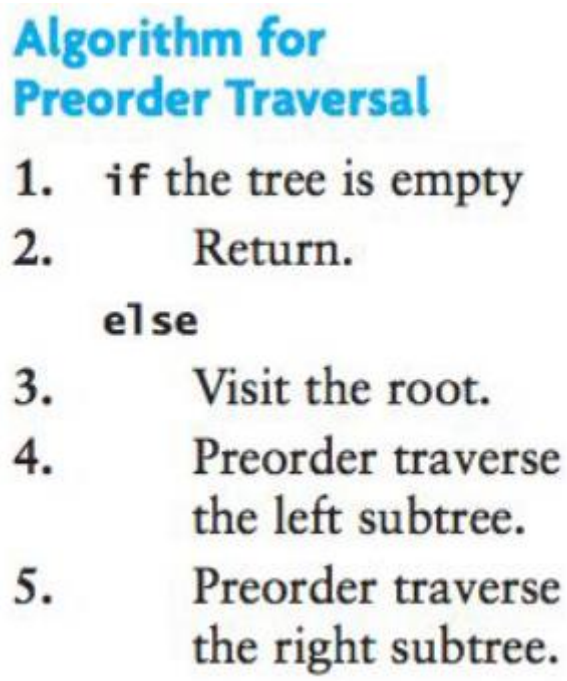

Figure 3 Kitaptaki Pre Order Traverse Algoritması

- **Part 1 – Level Order Iterator**

  Level Order Traverse probleminde ise Pre Order Traverse probleminde olduğu gibi BinarySearchTree sınıfının içerisinde **inner class olarak LevelOrderIterator** sınıfı oluşturdum. Bu sınıf içerisinde node'ları tutan bir adet queue objesi oluşturdum. levelOrderTraverse metodunun içerisinde döngü yardımıyla her seviyedeki node'ları ayrı ayrı bu queue'ya ekledim.

- Ardından iterator objesi için oluşturduğum **next( )** metodunda bu queue'dan poll metodunu kullanarak ilk elemanı çektim ve bunu return ettim**. hasNext( )** metodunda ise queue'nun boş olup olmadığını kontrol ettim. BinaryTree sınıfının iterator metodunda ise PreOrderIterator sınıfının no parameter constructor'ıyla oluşturduğum objeyi, BinarySearchTree sınıfının iterator metodunda ise LevelOrderIterator sınıfının no parameter constructor'ıyla oluşturduğum objeyi return ettim.

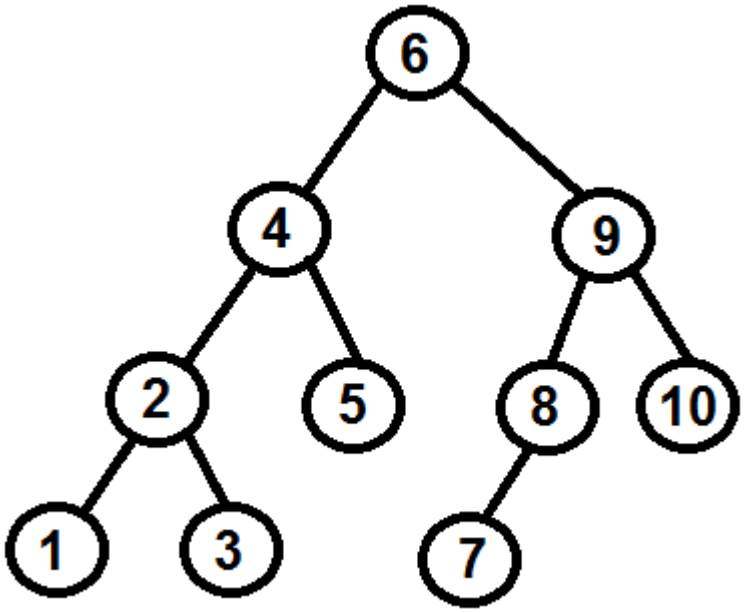

Figure 4 Test Kısmında Oluşturulan Binary Tree

- **Part 2 – Family Tree**

  Ödevin ikinci kısmındaki Family Tree probleminin çözümü için öncelikle bir **Person sınıfı** oluşturdum. Bu sınıfa Person'ın isim, parent ve nick bilgisini tuttum ve Person objelerini kıyaslamak için equals metodunu override ettim.

  Ardından FamilyTree sınıfının içerisinde oluşturduğum Person objelerini kullandım. FamilyTree sınıfının 3 parametre alan add metodunda yeni bir Person objesi oluşturdum ve ilk olarak bu objenin aynısının FamilyTree'de olup olmadığını **checkPerson( )** metoduyla kontrol ettim. Ardından eğer bu eleman ağaçta yoksa recursive çalışan **addToTree( )** metodunu kullanarak bu elemanı ağaca ekledim. Bu metod bir adet node ve Person objesi almaktadır. Metodun base case'i person node'unun null olmasıdır. Eğer person'ın parent'ıyla, kendi ismi aynı ise ve bu node'un sol çocuğu boş değil ise recursive fonksiyon içerisinde döngü yardımıyla node'un üzerinde sağa doğru giderek (kardeşler üzerinde) elemanı ekledim. Eğer ki bu node'un sol çocuğu boş ise buraya ekleme yaptım. Son olarak true return ettim ve recursive çağrıyı sağ ve sol çocuklar için tekrar yaptım.