Gebze Technical University
Computer Engineering

CSE 222 - 2018 Spring

HOMEWORK 4 REPORT

SEFA NADİR YILDIZ
131044031

Course Assistant:
Fatma Nur Esirci
Mehmet Burak Koca
Tuğbagül Altan Akın

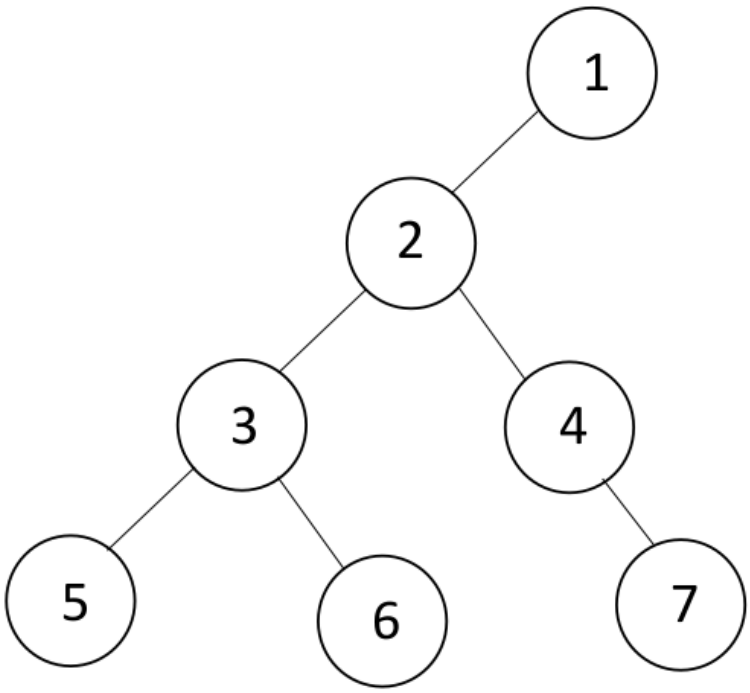
1 INTRODUCTION

1.1 Problem Definition

- **Part 1**

  Binary Tree sınıfı extend edilerek ağaç yapısında sol taraf çocuk, sağ taraf kardeş olacak şekilde yeni bir ağaç yapısı sınıfı yazılması istenmektedir. Bu yapıda ekleme ve arama methodları bulunmalıdır.

  Aşağıdaki örnekte görüldüğü gibi 1 numaralı node **root** olarak belirlenmştir.
  2, 4 ve 7 numaralı nodelar root'a çocuk olarak eklenmiştir. 3 ve 6 numaralı nodelar ise 3 numaralı node'a çoçuk olarak eklenmiştir. 5 numaralı node'da 3 numaralı node'un çoçuğu olarak görülmektedir.



1.2 System Requirements

PART 1

Öncelikle Binary Tree sınıfı bulunmalıdır. Bu sınıf **extends** etmek için kullanılacaktır.

Parent ve child eklemek için **add** methodu gerekmektedir. Ekleme işleminin başarılı olması durumunda **true**, başarısız olması durumunda false return etmelidir.

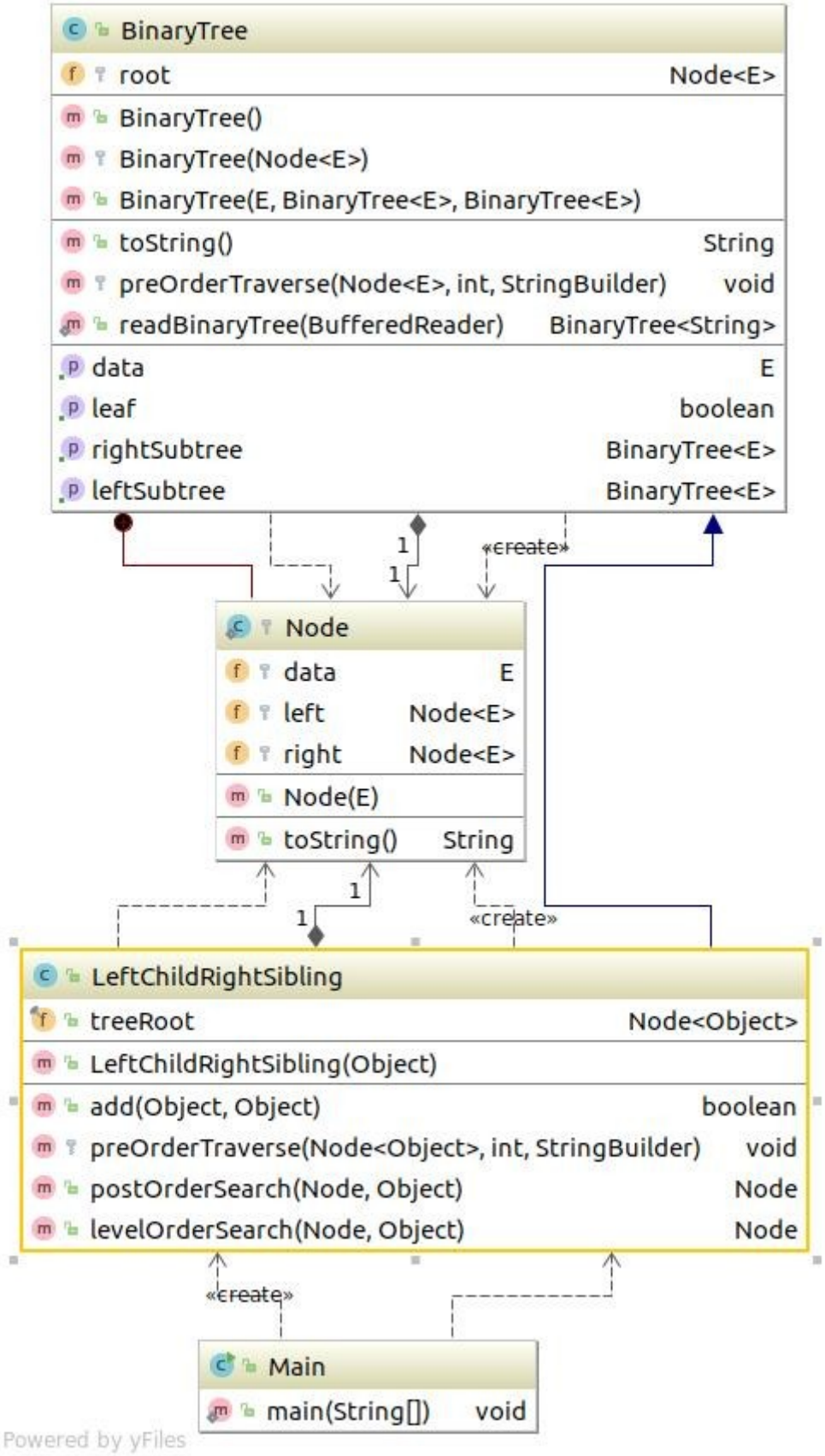
levelOrderSearch isimli bir arama methoduna ihtiyacı vardır. Bu method root'dan başlayarak onların çocuklarını gezecek şekilde bir arama yapmalıdır.

postOrderSearch isimli bir arama methoduna daha ihityacı vardır. Bu method ise en alttaki çocuktan başlayarak root'a doğru arama yapmalıdır.

preOrderTraverse methodu bu ağacı ekrana basmalıdır. Binary Tree sınıfından override edilerek yapılacaktır.

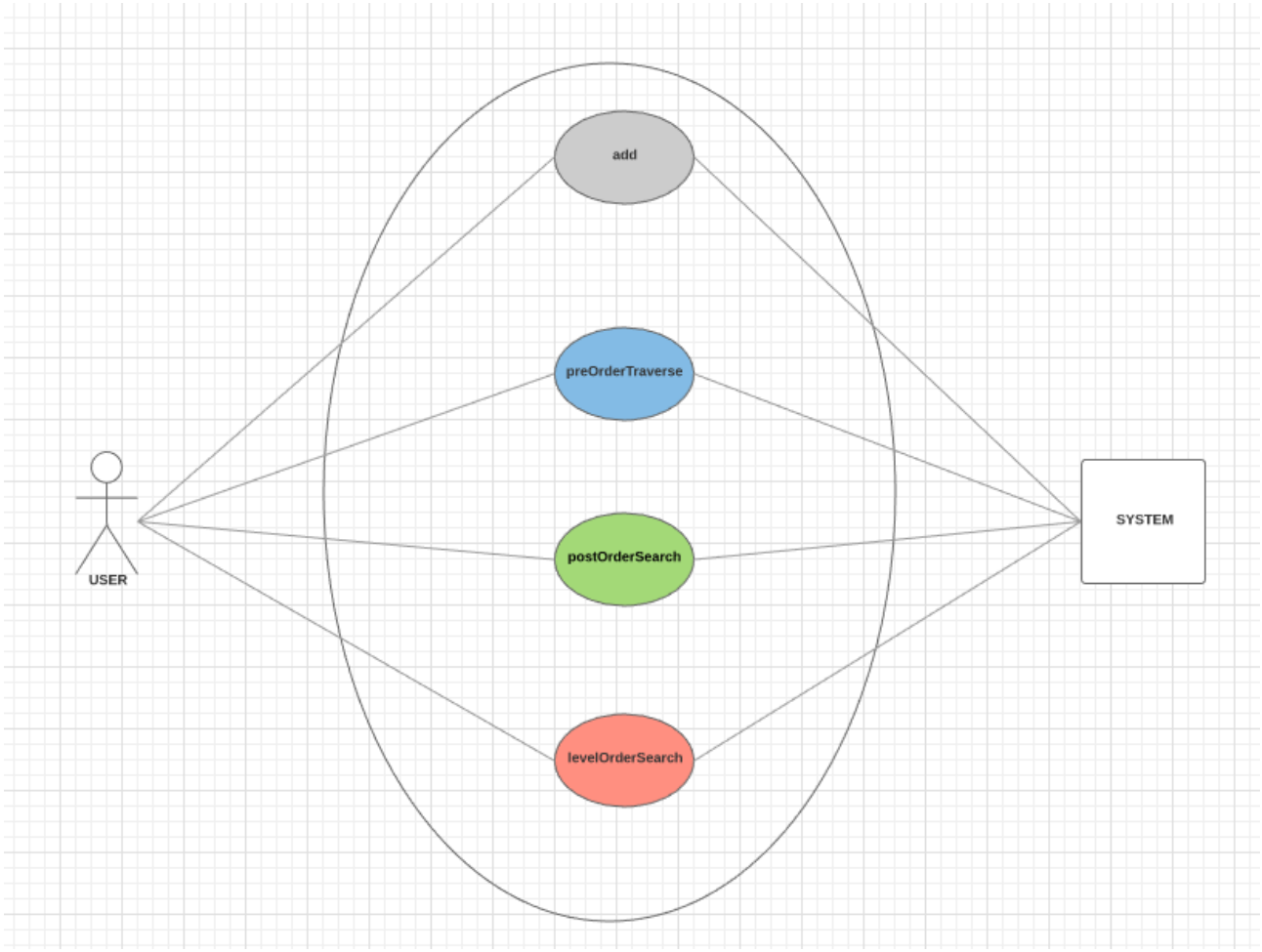
2 METHOD

2.1 Class Diagrams



2.2 Use Case Diagrams

PART1



2.3 Problem Solution Approach

PART 1

public class LeftChildRightSibling<Object> extends BinaryTree<Object> isimli bir generic bir sınıf oluşturdum. Binary Tree sınıfından bu sınıfı extend ettim. Ağacın başlangıcını tutabilmek için **public final Node<Object> treeRoot** isimli Node tipinde bir değişken oluşturdum.

Add methodumun için **public boolean add(Object addNode, Object addData)** isimli methodu oluşturdum. İlk parametre çocuk eklenecek olan node'un data bilgisidir. İkinci parametre ise yeni eklenecek olan node'un datasıdır.

Eklenecek olan node'un referansını **Node addToNode = levelOrderSearch(treeRoot, addNode)** methodunu çağırarak bulmaktayım ve eklenecek olan datayı bu node'a çoçuk veya kardeşlik durumuna göre eklemekteyim.

Arama methodlarından **levelOrderSearch(Node root, Object searchData)** methodunda parametre olarak ağacın root'unu ve aranacak datayı almaktayım. Root ksımından başlayarak ve çocukları gezerek bir arama yapmaktayım.

Öteki arama methodu olan **public Node postOrderSearch(Node root, Object searchData)** ise yine parametreden ağacın root'unu ve aranacak datayı almaktayım ama bu sefer arama yaparken en alt çocuğa kadar inip oradan başlayıp root'a doğru arama yapmaktayım.

Bu ağacı ekrana bastırmak için de **protected void preOrderTraverse(Node<Object> node, int depth, StringBuilder sb)** methodunu kullanmaktayım. Binary Tree class'ından override edilmiş bir methoddur. Parametreden ağacın root'unu, ağacın yükseklik bilgisini ve ağacı ekrana yazacağı StringBuilder değişkenini parametreden alır ve parent ve child ilişkisine göre ekranda ağacı gösterir.