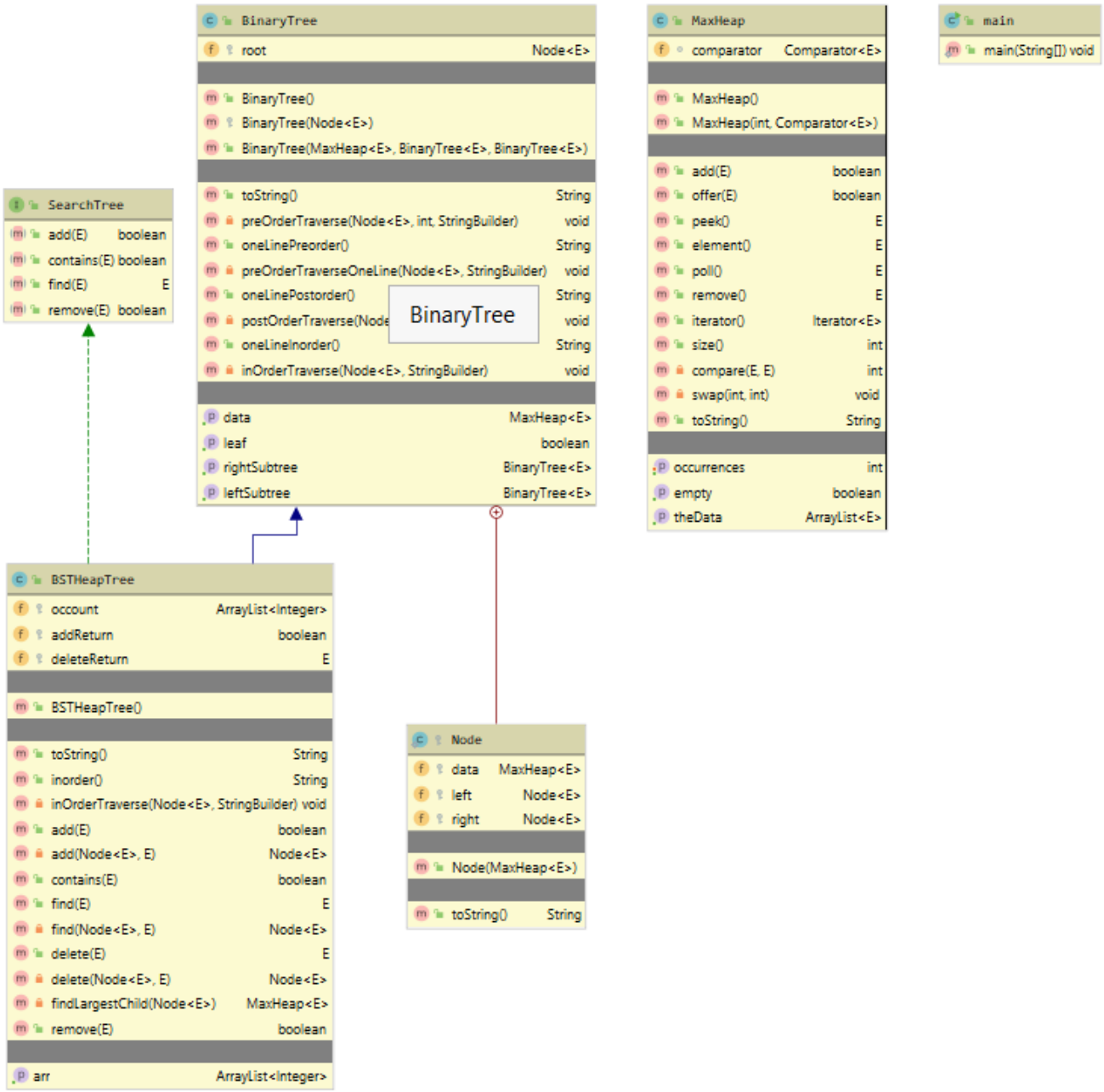
GTU Department of Computer Engineering
CSE 222/505 - Spring 2021
Homework 4 Report

Refik Orkun Arslan
151044063

Class Diagram



Problem Solutions Approach

In the binary tree system, we converted the node part to a heap structure, so the maxheap structure was found in the leaves, not variable types such as integer or string.I wrote accordingly, like search insertion and deletion, and implemented both in binary tree and heap structure

.

Post-order traverse output little part

```
[4327, 4343, 4356, 4358, 4373, 4380, 4370, 4376]
[4337, 4386, 4339, 4412, 4388, 4353, 4357, 4413]
[]
[4340, 4364, 4349, 4381, 4400, 4371, 4376, 4404]
[4366, 4402, 4366, 4415]
[4501, 4850, 4600, 4911, 4932, 4957, 4635, 4913]
[4431, 4449, 4445, 4455, 4464, 4473, 4477, 4491]
[4419, 4427, 4422]
[4441, 4475, 4448, 4478, 4478, 4487, 4472, 4484]
[]
[4453, 4466, 4483, 4472, 4486, 4495, 4494, 4481]
[4452]
[4458, 4465, 4472, 4467, 4491, 4487, 4495, 4483]
[4467, 4473, 4468, 4497, 4474]
[4590, 4611, 4687, 4829, 4925, 4900, 4775, 4990]
[4502, 4508, 4503, 4523, 4510, 4526, 4542, 4554]
[4504, 4549, 4516, 4569, 4565, 4538, 4534, 4579]
[4506, 4540, 4573, 4540, 4555, 4577, 4577, 4588]
[4508, 4523, 4528, 4539, 4546, 4548, 4542, 4563]
[4533]
[4694, 4754, 4772, 4829, 4800, 4804, 4941, 4915]
[4593, 4603, 4598, 4614, 4689, 4657, 4654, 4685]
[4596, 4597, 4636, 4679, 4679, 4688, 4664, 4692]
[4605, 4607, 4615, 4651, 4618, 4692, 4620, 4685]
[4633, 4663, 4662, 4668, 4692, 4687, 4674, 4671]
[]
[4637, 4664, 4664, 4682, 4681, 4668]
[4696, 4799, 4864, 4946, 4928, 4905, 4920, 4975]
[4744, 4750, 4967, 4758, 4858, 4982, 4968, 4970]
[4700, 4700, 4733, 4710, 4703, 4736, 4736, 4723]
[4718, 4720, 4720, 4726, 4731, 4743, 4737, 4740]
[]
[4732]
[4777, 4847, 4919, 4848, 4883, 4989, 4922, 4917]
```

```
BSTHeapTree<Integer> BST=new BSTHeapTree<~>();
    BST.occount= new ArrayList<Integer>();
    Random rand = new Random();
    int upperbound = 5000;
    int int_random;
    for(int i=0;i<3000;i++)
    {
        int_random = rand.nextInt(upperbound);
        BST.add(int_random);
        BST.arr.add(int_random);

    }



    Collections.sort(BST.arr);
    Object o = BST.arr.get(0);
    int n = 1;
    for (int i = 1; i < 3000; i++) {
        Object t = BST.arr.get(i);

        if (o.equals(t)) {
            n++;
        } else {

            n = 1;
            o = t;
        }
        BST.occount.add(n-1);

      // System.out.println(BST.oneLinePreorder());

  for(int i=10;i<110;i++)
  {
      if(BST.find(i) !=0)
      {
          System.out.println("find "+i+" "+" occurence "+ BST.find(i));

      }

  }
  for(int i=0;i<100;i++)
  {
      int_random = rand.nextInt( bound: 3000);
      System.out.println(BST.remove(int_random));

  }
```

```
find 13  occurence 1
find 15  occurence 1
find 22  occurence 1
find 23  occurence 2
find 24  occurence 3
find 30  occurence 1
find 43  occurence 1
find 51  occurence 1
find 53  occurence 1
find 62  occurence 1
find 68  occurence 1
find 71  occurence 1
find 83  occurence 1
find 86  occurence 1
find 101  occurence 1
find 104  occurence 1
false
false
false
false
false
false
false
false
false
false
false
false
false
false
false
false
false
false
false
```