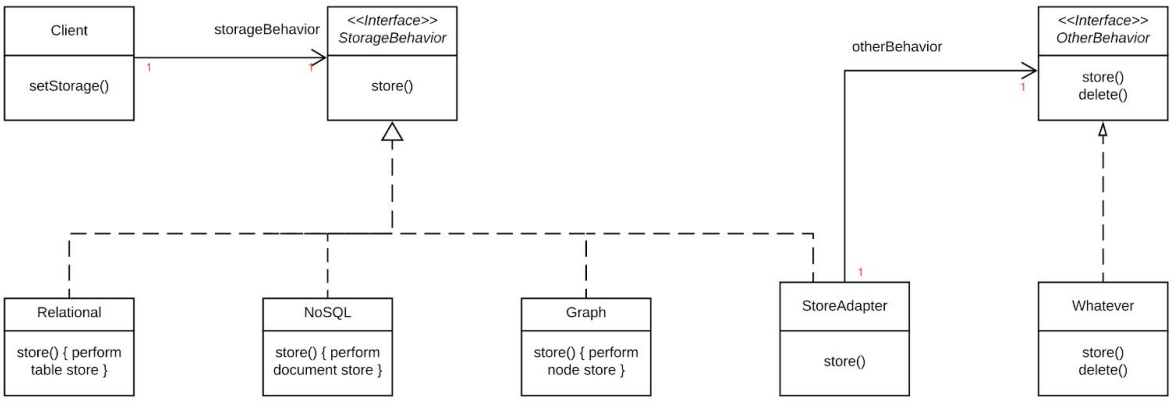
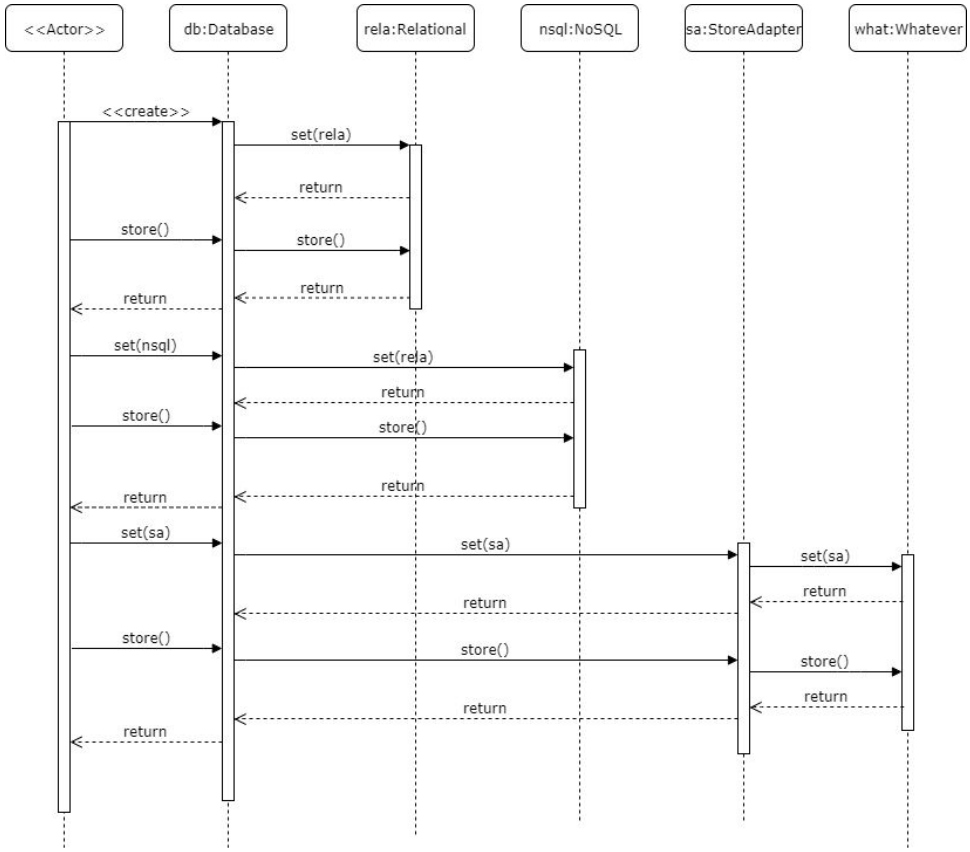
ESOF 322 Homework 3
Group member: Shengnan Zhou, Kyle Rathman

Exercise 1:

a) UML class diagram



b) UML sequence diagram



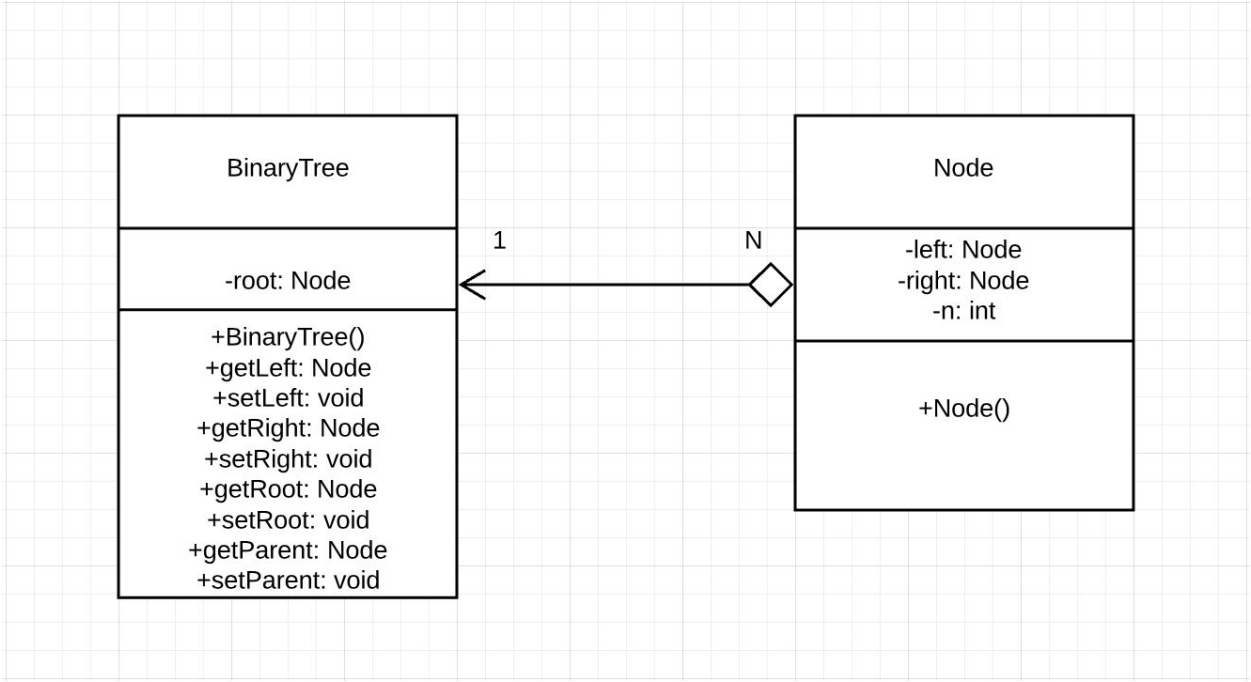
Exercise 2:

a) Velocity of 3-man team = 32 story points / sprint (45-man days)
New team available man days = 45 + 15 + 15 * 0.8 = 72-man days
Focus factor of previous team = 32/45 = 0.71
New focus factor = (3 * 0.71 + 2 * 0.7) / 5 = 0.706
Estimated velocity = 72 * 0.706 = 50.832

b) For a new team, the focus factor should be assumed to be about 70%. This prevents placing too much on a team that may not be up to the challenge. If they are capable of performing better, then the focus factor should naturally rise with subsequent sprints.

c) Estimate Story points:
Instead of using poker to estimate story points, we could also use the square of the list of integer: 1^2, 2^2, 3^2, 4^2, ... So the options of story points will be 1, 4, 9, 16, 25, 36,..... This method might be better to estimate middle sized project and it will be more detailed, but not for smaller sized project since there is not many numbers (2, 3, 5, etc.) But there will be more options for middle sized project (9, 16, 25, etc.)

d) UML class diagram of binary tree



e) Java implementation of binary tree

```
class Node {
    private int n;
    private Node parent;
    private Node left;
    private Node right;

    public Node(int value) {
        this.n = value;
    }
}


class BinaryTree {
    private Node root;

    public BinaryTree(int n) {
        root = new Node(n);
    }

    public BinaryTree() {
        root = null;
    }

    public Node getLeft() {
        return root.left;
    }

    public void setLeft(int n) {
        root.left = n;
    }

    public Node getRight() {
        return root.right;
    }

    public void setRight(int n) {
        root.right = n;
    }

    public Node getRoot() {
        return root;
```

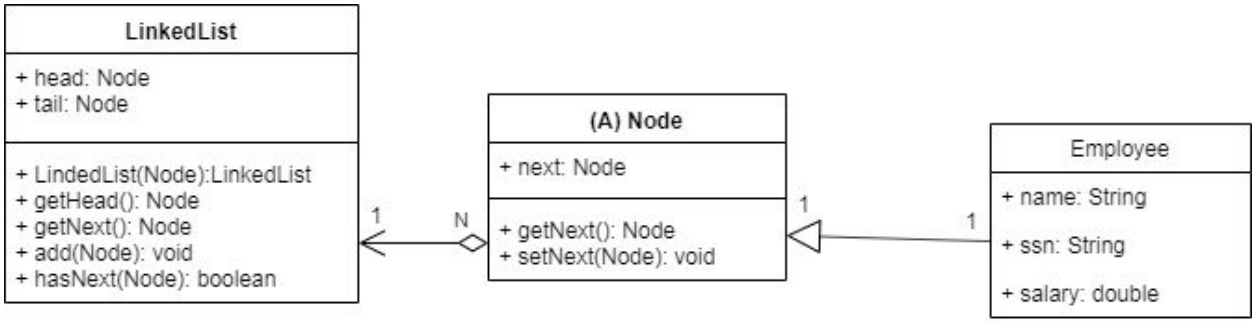
```
    }

    public void setRoot(int n) {
        root = n;
    }

    public Node getParent(Node n) {
        return n.parent
    }

    public void setParent(Node n, Node m) {
        n.parent = m
    }
}
```

f)



g)
```
class LinkedList{
    public Node head;
    public Node tail;

    public LinkedList(Node h){
        head = h;
        tail = head;
    }

    public Node getHead(){
        return head;
    }

    public Node getNext(){
        return head.getNext();
    }

    public void add(Node a){
```

```
        tail.setNext(a);
        tail = a;
    }

    public boolean hasNext(Node n){
        return n.getNext() != null;
    }
}

class Employee extends Node{
    public String name;
    public String ssn;
    public double salary;

    public Employee(String n, String sn, double s){
        name = n;
        ssn = sn;
        salary = s;
    }
}

abstract class Node{
    public Node next;
    public Node getNext(){return next;}
    public void setNext(Node n){}
}
```