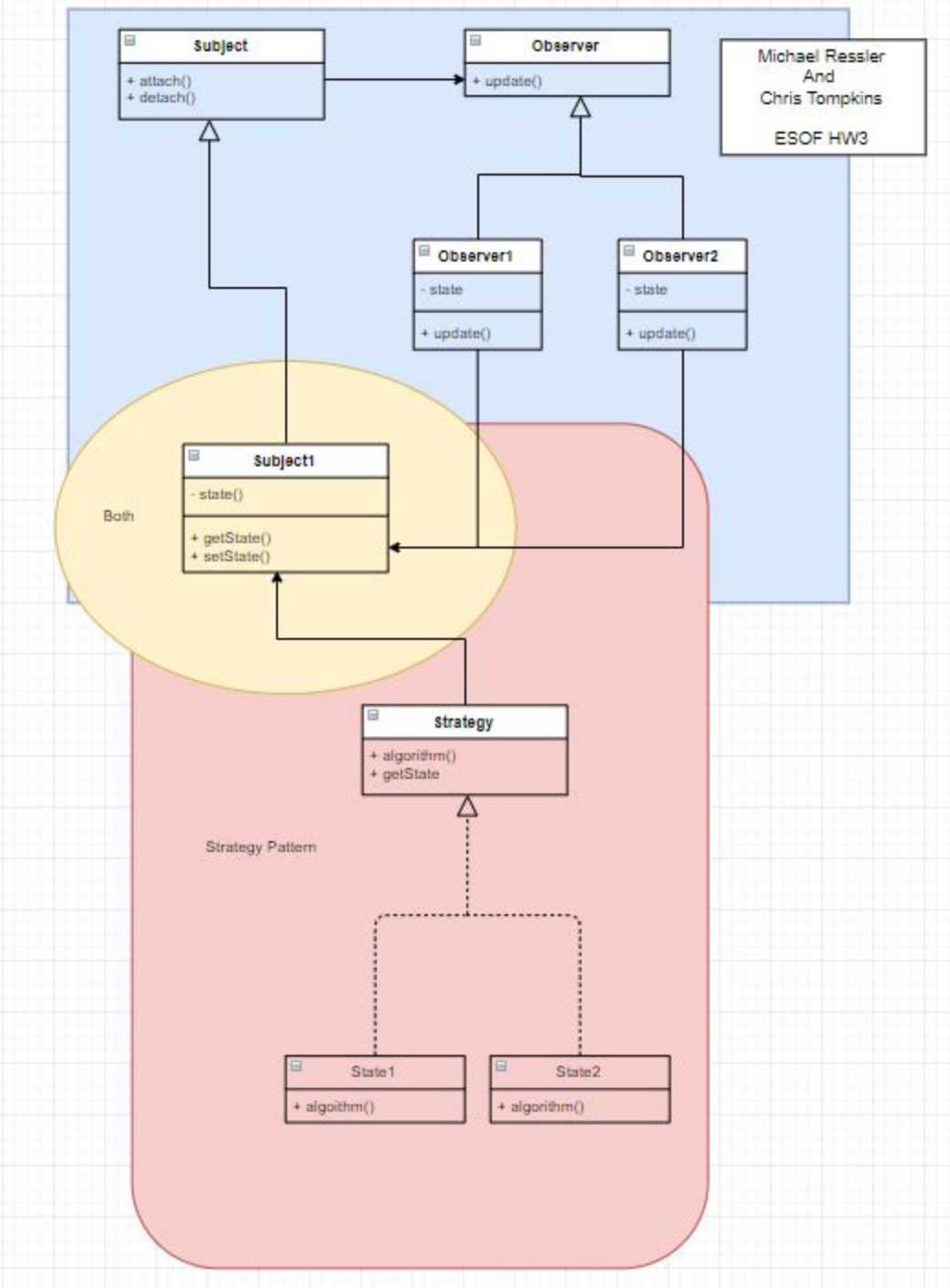
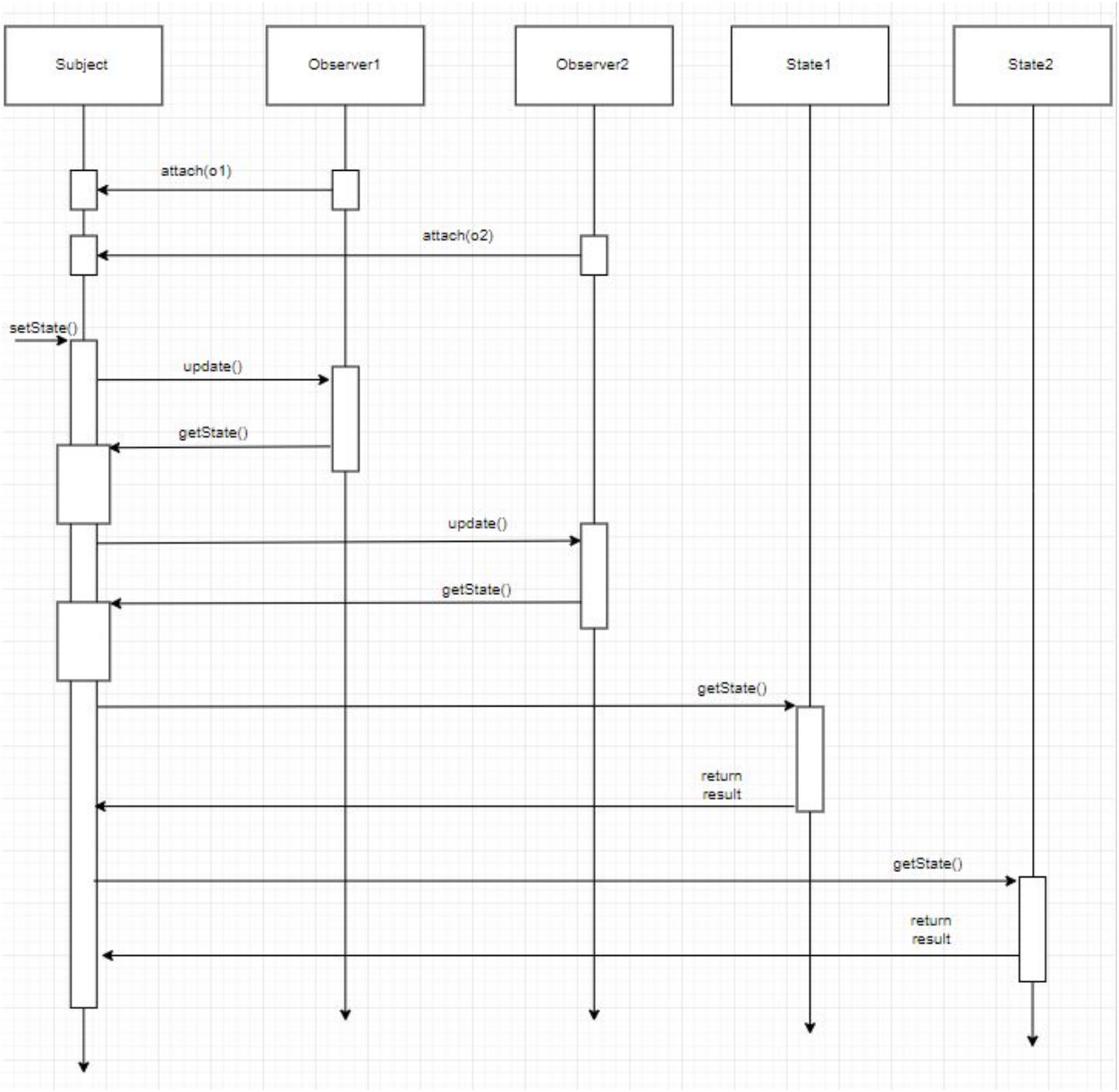
Michael Ressler and Chris Tompkins
ESOF 322 HW3
9/26/2019

a) Draw a UML class diagram that clearly shows the coupling. Please identify which class(es) participates in multiple patterns.



b) Draw a UML Sequence diagram where you demonstrate the behavior of an instance of the coupled class from the perspective of one pattern, then from the perspective of the other pattern.



a) Assume during your team's last sprint, that they completed 32 stories points using a 3-person team working in sprints of 3 weeks for a total of 45-man-days. Calculate your team's estimated velocity for the next sprint if we still have 3-week sprints, but you now added 2 engineers to the team, and one of them can only work 80% of the time.

Focus Factor = 32/45
80% of 15 =12
-> 12 + 15 + 45 = 72 man-days

Estimated Velocity = 72 * 32/45 = 51.2 story points

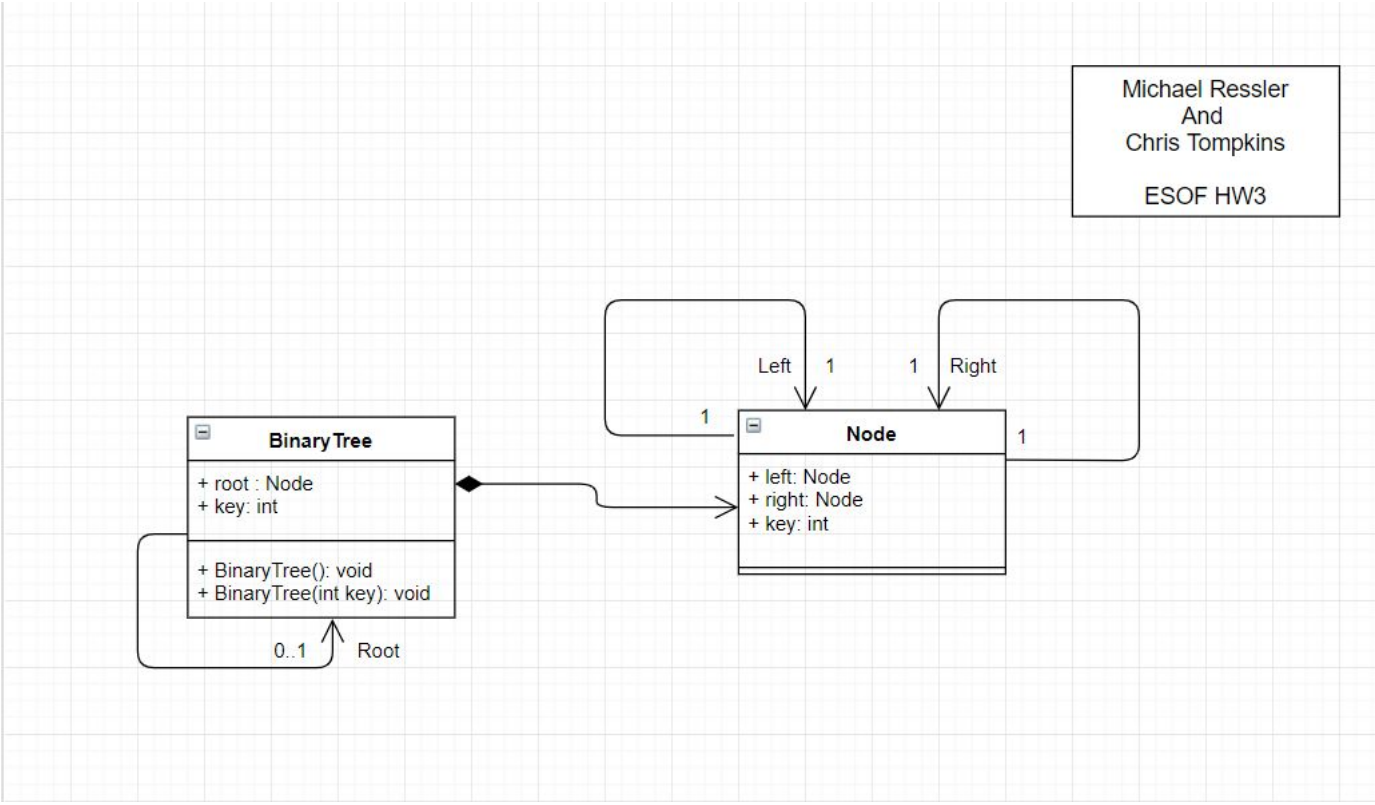
b)How would you estimate a focus factor for a brand-new team?

For a brand new team, a good rule of thumb is to go with 70% which is what most companies go with and is the industry average.

c)We looked at using poker using semi-Fibonacci sequences to estimate story points. Think of another way to estimate story points and explain them. Is it better or worse than poker?

A different method that I've heard about is the T-shirt size method. Items are estimated into sizes such as XS, S, M, L, XL where XS could represent the variables of time (1-3 weeks), people (1-2) and cost. They could go up to XL of 18+ months, 20+ people and high cost. It's a quick and informal method to estimate the size of a backlog. It can be better than planning poker because unlike poker, it can be done for large teams and large amounts of items. It allows open discussion, however voting anonymously in poker can be beneficial to a team so everyone's input is included.

d)Draw a UML class diagram of a binary tree. Each node contains an integer.



e)Provide the corresponding object-oriented code that implements your binary tree design from part d.

```
/* Michael Ressler And Chris Tompkins
 * ESOF 322 / Homework #3
 * 9/26/19
 */

public class Node {

    public int key;
    public Node left, right;

    public Node(int item)
    {
        /* new node created with integer
         * sent from BinaryTree with children set to null
         */
        key = item;
        left = right = null;
    }
}
```

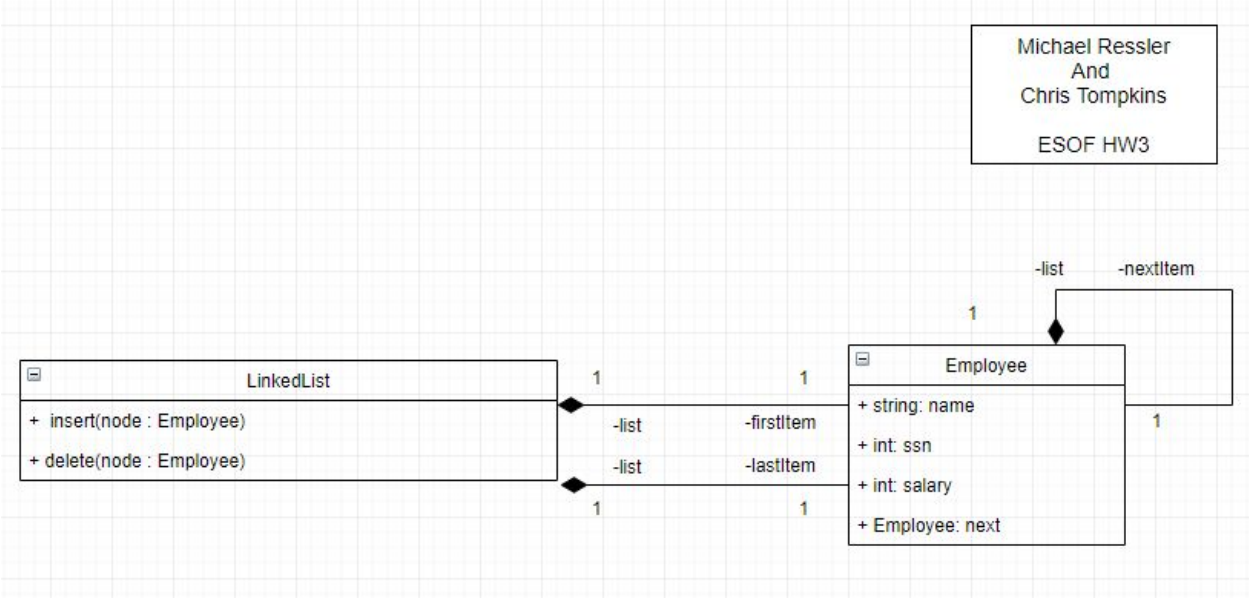
```
/* Michael Ressler And Chris Tompkins
 * ESOF 322 / Homework #3
 * 9/26/19
 */

public class BinaryTree {
    // Root of Binary Tree
    Node root;

    // Constructors
    BinaryTree(int key)
    {
        root = new Node(key);
    }

    BinaryTree()
    {
        root = null;
    }
}
```

f)Draw a UML class diagram of a linked list that contains Employee records as data. An Employee record has a name, a social security number, and a salary.



g)Provide the corresponding object-oriented code that implements your linked list from part f.

```
//Chris Tompkins & Michael Ressler
//
//
public class LinkedList {

	Employee head;

	static class Employee {

		int ssn;
		int salary;
		String name;
		Employee next;

		Employee(int s, int p, String n) {
			ssn = s;
			salary = p;
			name = n;
			next = null;
		}
	}
```

```
public static LinkedList insert(LinkedList list, int ssn, int salary, String name) {
        Employee new_node = new Employee(ssn, salary, name);
        new_node.next = null;

        if (list.head == null) {
                list.head = new_node;
        } else {
                Employee last = list.head;
                while (last.next != null) {
                        last = last.next;
                }

                last.next = new_node;
        }

        return list;
}

void deleteNode(int position) {
        if (head == null)
                return;

        Employee temp = head;

        if (position == 0) {
                head = temp.next;
                return;
        }

        for (int i = 0; temp != null && i < position - 1; i++)
                temp = temp.next;

        if (temp == null || temp.next == null)
                return;

        Employee next = temp.next.next;

        temp.next = next;
}
```