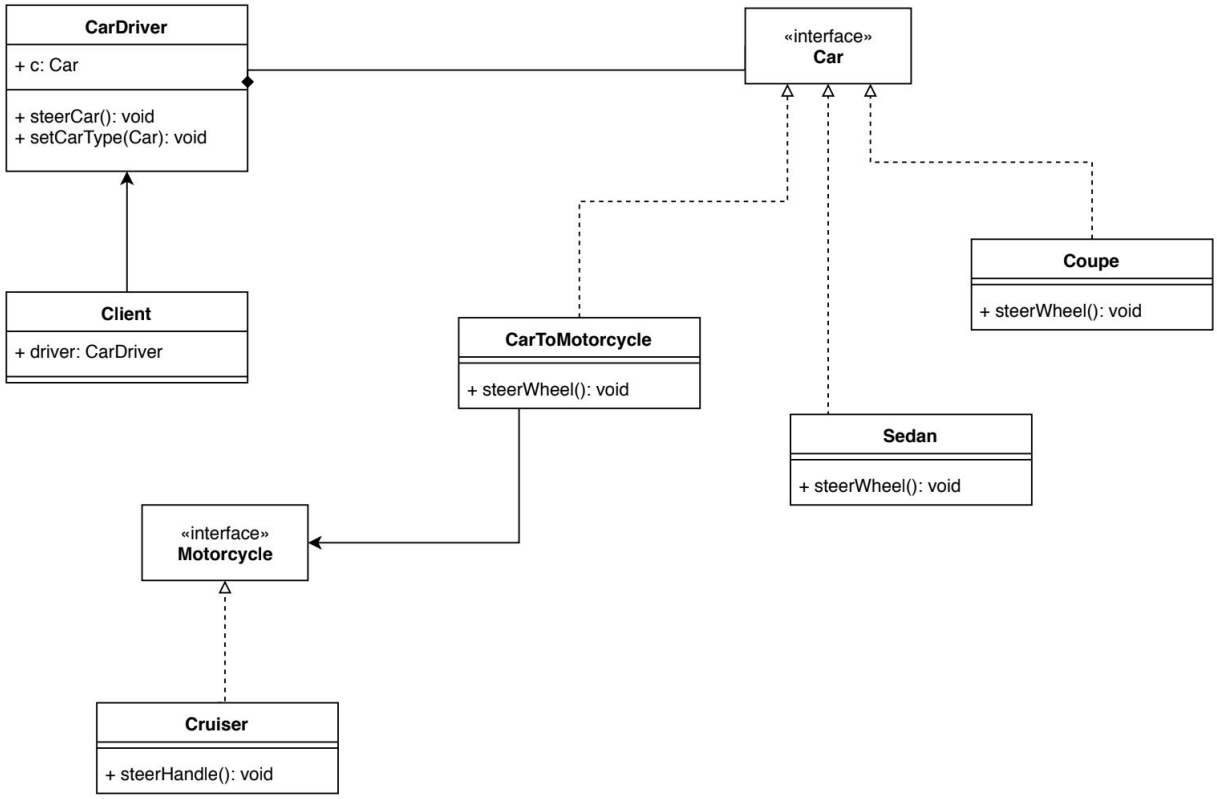
ESOF 322 Homework 3

Axel Ward
Cooper Strahan

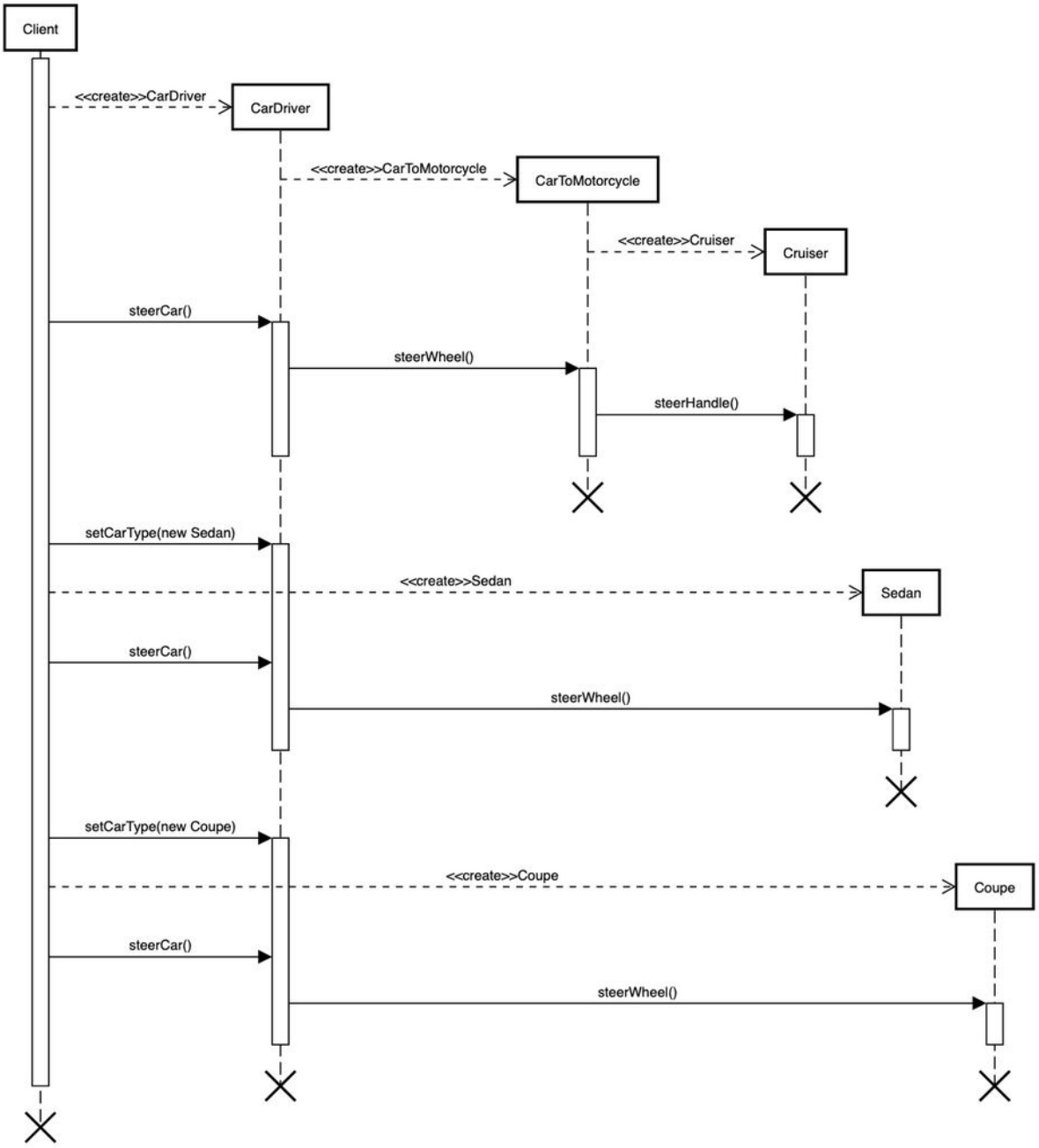
Part 1

1) Implementation of Strategy and the Adapter Pattern



The class that is shared between the two designs is the CarToMotorcycle class. It takes the steerWheel() function and properly converts it to a steerHandle() call.

2) Sequence Diagram

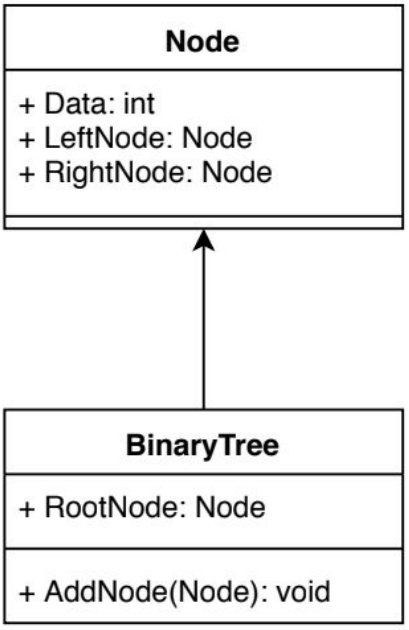


Part 2

1. (45 * .7167) + ((15 * .7) + ((15 * .8) * .7)) = 50.9 story points completed

2. For an employee that you have not previously worked with you estimate their focus factor at 70%. If the employee has worked under/with other managers in your company before you can ask them to help determine your estimate.

3. Another way to estimate story points time requirements would be to have each engineer estimate the time requirements a task is going to take them when they check the task out. This is worse than the poker example as you don't get an idea of what the time requirements before you start to work on them, and additionally other engineers may have input on the time requirement that you will never get to hear. The one improvement over the poker example is that people don't have to worry about working on the time requirements for story points they will not ever work on.

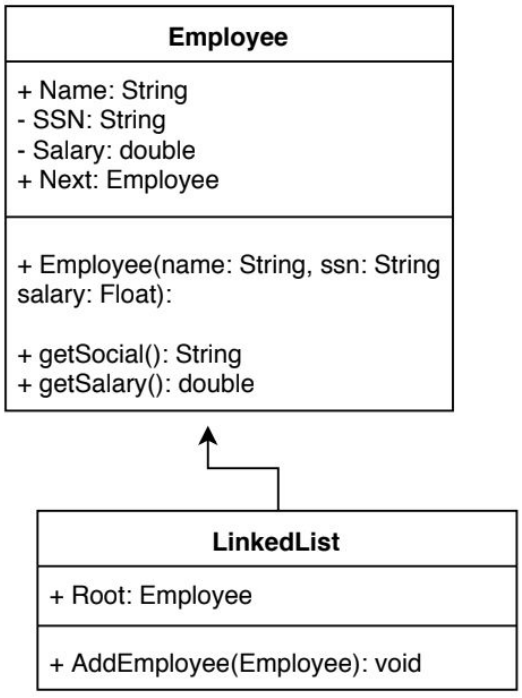


4.

5. Binary Tree Code:

```
//Create Structure for the Node Class
Class Node {
        Public int Data;
        Public Node LeftNode;
        Public Node RightNode;

        //Constructor requires you to create the node with an integer value
```

```
        Public Node(int inData) {
                this.Data = inData;
                this.LeftNode = null;
                this.RightNode = null;
        }
}

//Create Structure for the Binary Tree Class
Class BinaryTree {
        Public Node RootNode;

        //The Constructor is passed a Node which is then set to the root node
        Public BinaryTree(Node StartNode) {
                this.RootNode = StartNode;
        }


        //Attaches child nodes to the root node
        Public void addNode(Node n) {
                // Functionality Here
        }
}
```



6.

7. Linked List Code:

```
//Create Structure for the Employee class
Class Employee {
        Public String Name;
        Private String SSN;
        Private double Salary;
        Public Employee Next;

        //Constructor requires you to pass Name, SSN,
        //and Salary to create an employee
        Public Employee(String name, String ssn, double salary) {
                this.Name = name;
                this.SSN = ssn;
                this.Salary = salary;
                this.Next = null;
        }

        //Method to get Employee's Social Security number
        Public String getSocial() {
                return this.SSN;
        }

        //Method to get Employee's Salary
        Public double getSalary() {
                return this.Salary;
        }
}

//Structure for the Linked List Class
Class LinkedList {
        Public Employee Head;

        //Constructor creates a linked list when passed an employee with the employee
        //set to the head of the linked list
        Public LinkedList(Employee employee) {
                this.Head = employee;
        }
        //Attaches employees to Employee.Next
        Public void addEmployee(Employee e) {
                // Functionality Here
        }
}
```