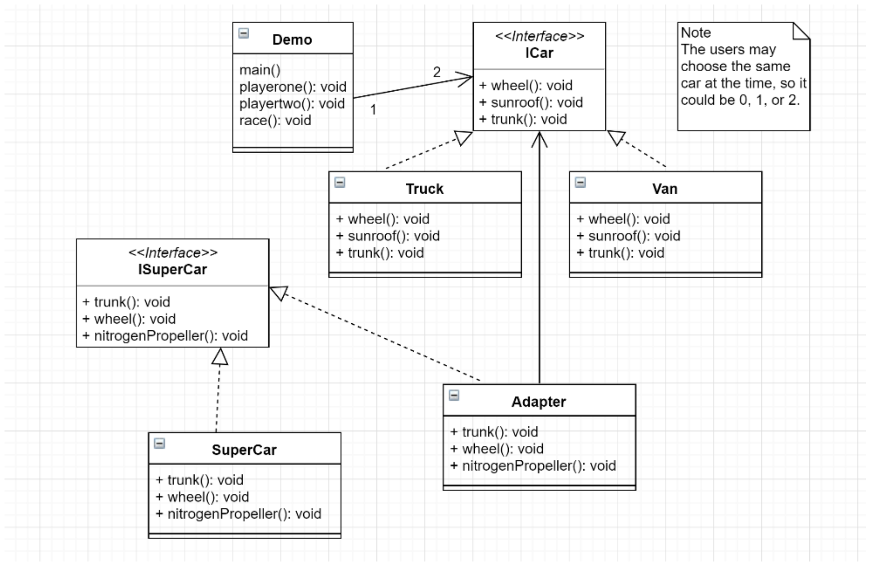
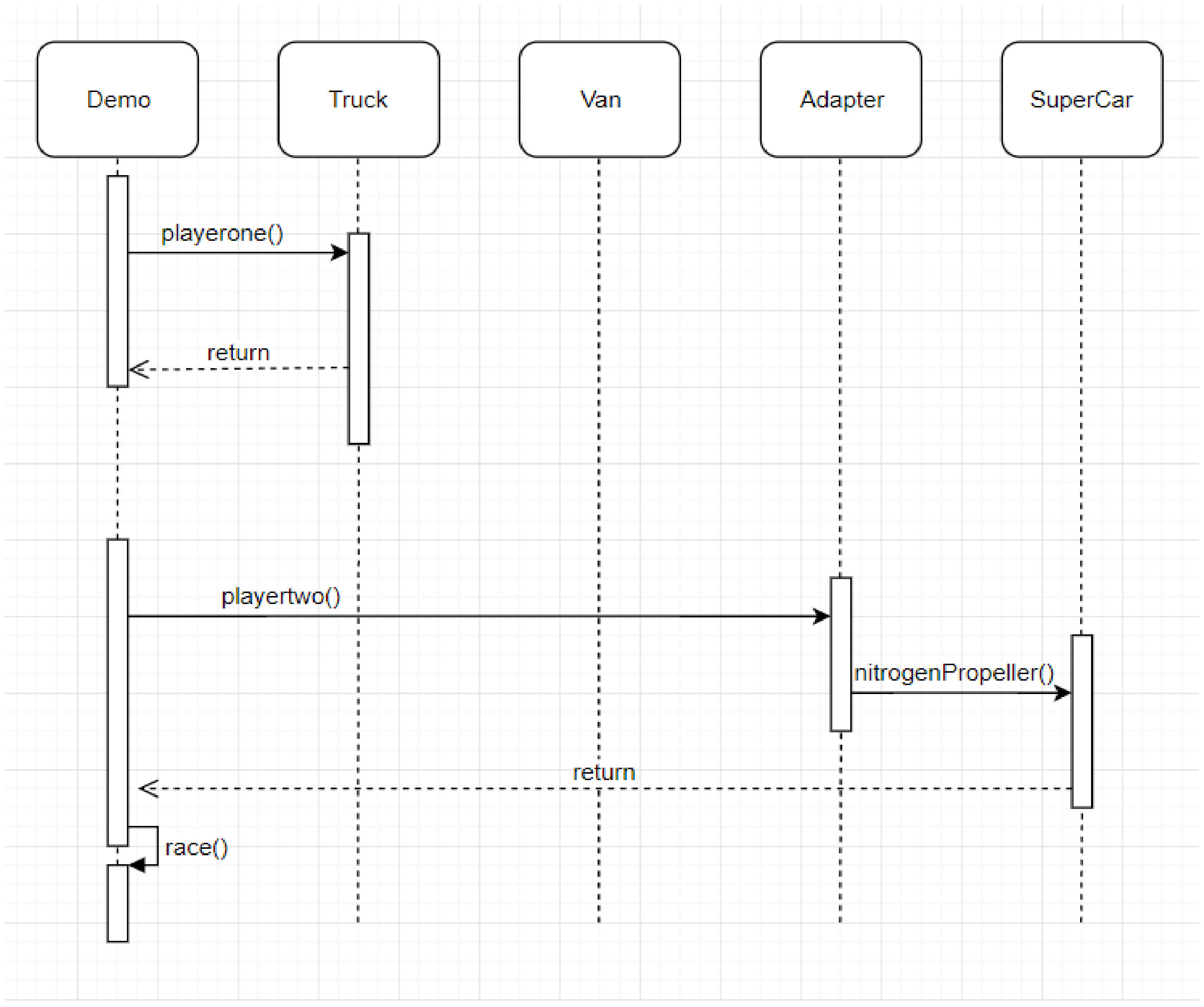
Hung-Kai Sun

ESOF

HW3

Exercise 1





Exercise 2

a.

The first three people each can work 15-man days, and for the newcomers, one can work 15-man days, the other only can work 12-man days.
Focus factor (first three people): 32/45 = 0.711
Total man days (five people): 15 + 15 + 15 + 15 + 12 = 72
Team's estimated velocity = 0.711* 72 = 51.19, so is about 51 story points.

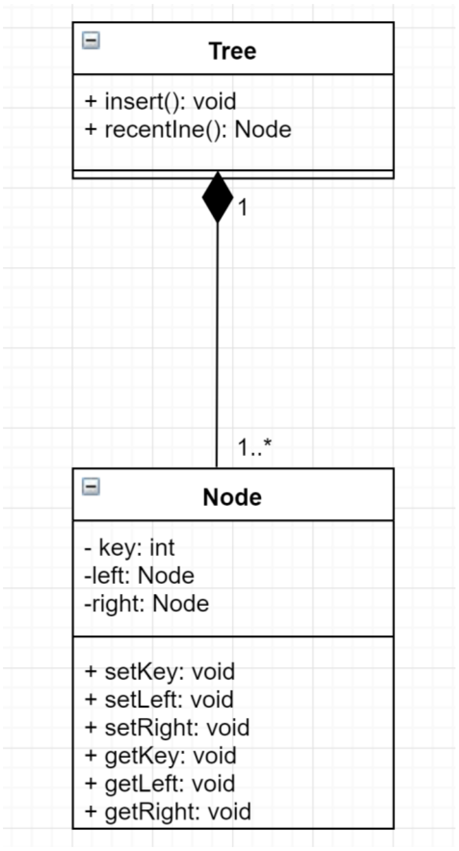
b.

The brand-new team's focus factor = 51/72 = 0.708

c.

Affinity Estimating, the benefits of Affinity Estimating is speed, it is faster than planning poker, it also considers the subjectivity of each team member's observation process. However, I can't tell which way is better. Even Affinity Estimating is faster, but it can't expect participants to estimate things they don't know, which means they still have to spend time discussing their estimated projects. Although Affinity Estimating saves a lot of time, you may lose accuracy compared to Planning Poker.

d.



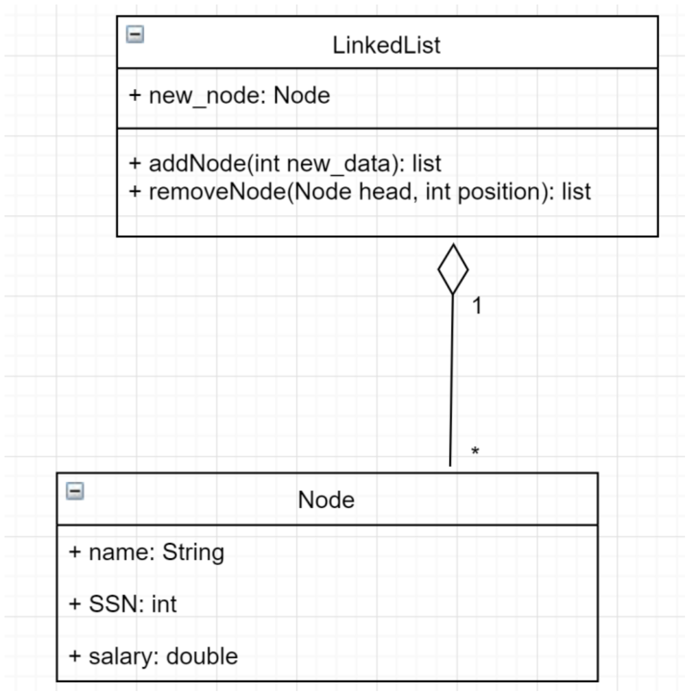
e.

```
class Node {

  int key;
  Node left, right;

  public Node(int item) {
        this.key = item;
        this.left = null;
        this.right = null;
  }

  public void setKey(int key) {
        this.key = key;
  }

  public void setLeft(Node left) {
        this.left = left;
  }
```

```
	public void setRight(Node right) {
		this.right = right;
	}

	public Integer getKey() {
		return this.key;
	}

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

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

	public class BinaryTree {

	// Root of Binary Tree
	Node root;

	BinaryTree() {
		root = null;
	}
	public void insert(int value) {
		this.root = this.recentInt(root, value);
	}

	public Node recentInt(Node roots, int node) {
		if (roots == null) {
			return new Node(node);
		}
		if (roots.getKey() < node) {
			roots.setRight(this.recentInt(roots.getRight(), node));
			return roots;
		} else {
			roots.setLeft(this.recentInt(roots.getLeft(), node));
```

```
                return roots;
            }
        }
}
```

f.



LinkedList
+ new_node: Node
+ addNode(int new_data): list
+ removeNode(Node head, int position): list
1
*
Node
+ name: String
+ SSN: int
+ salary: double

g.

```
public void addNode(int new_data)
{
        Node new_node = new Node(new_data);

        //Make next of new Node as head
        new_node.next = head;

        //Move the head to point to new Node
        head = new_node;
}
Node removeNode(Node head, int position) {
        if (head == null) {
                return null;
        } else if (position == 0) {
                return head.next;
        } else {
                Node n = head;
                for (int i = 0; i < position - 1; i++) {
                        n = n.next;
                }
                n.next = n.next.next;
                return head;
        }
}
public static Node {
        String name;
        Int SSN;
        Double salary;
}
```