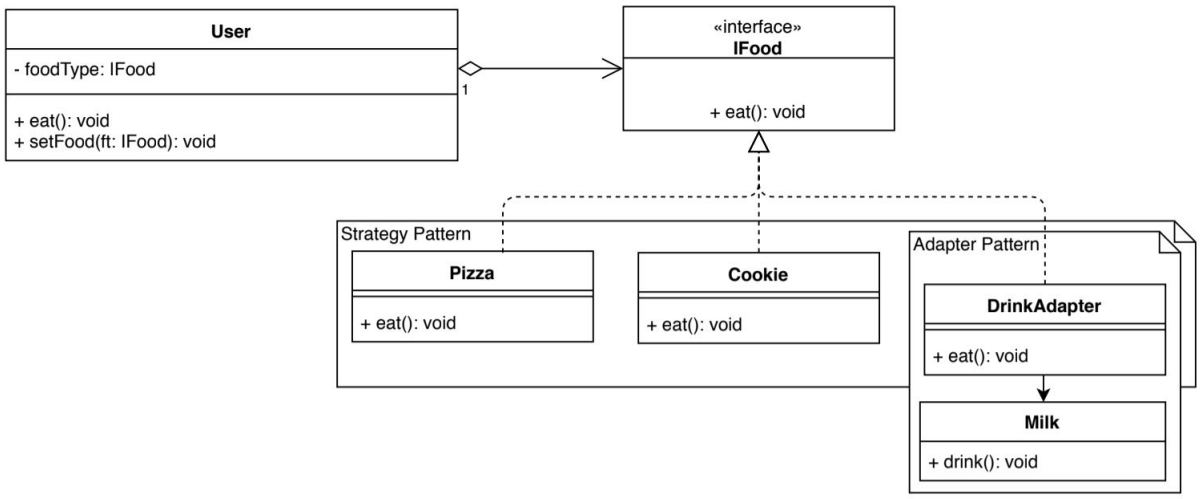
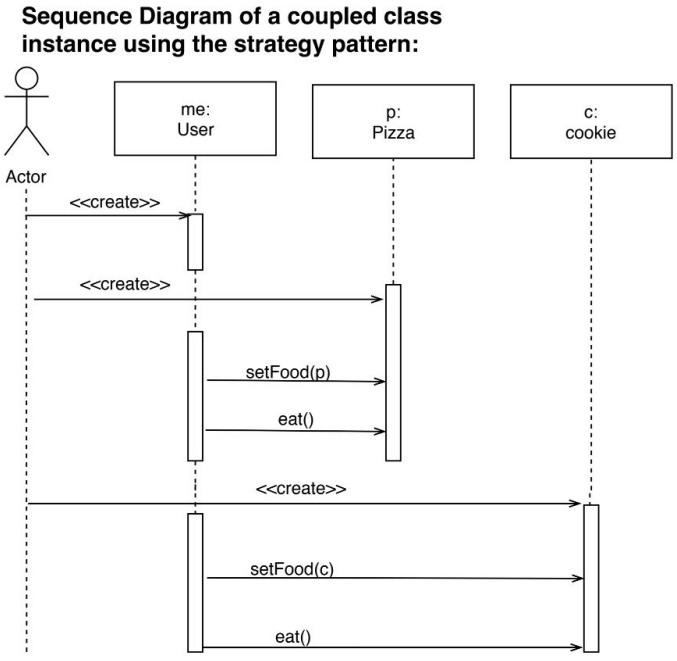
Forest Edwards and Michael Utt
ESOF 322
Homework 3

Exercise 1

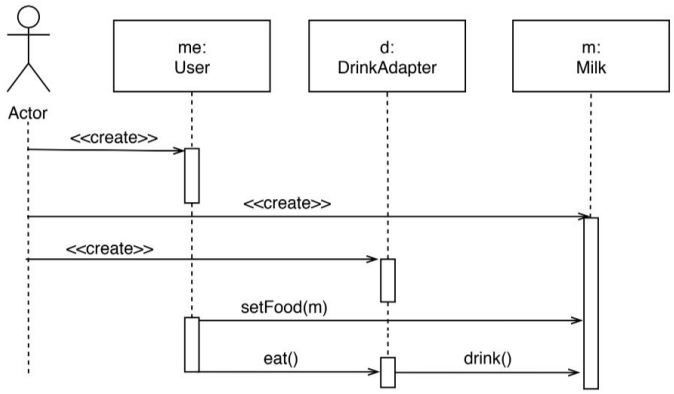
a)



b)



Sequence Diagram of a coupled class instance using the adapter pattern:



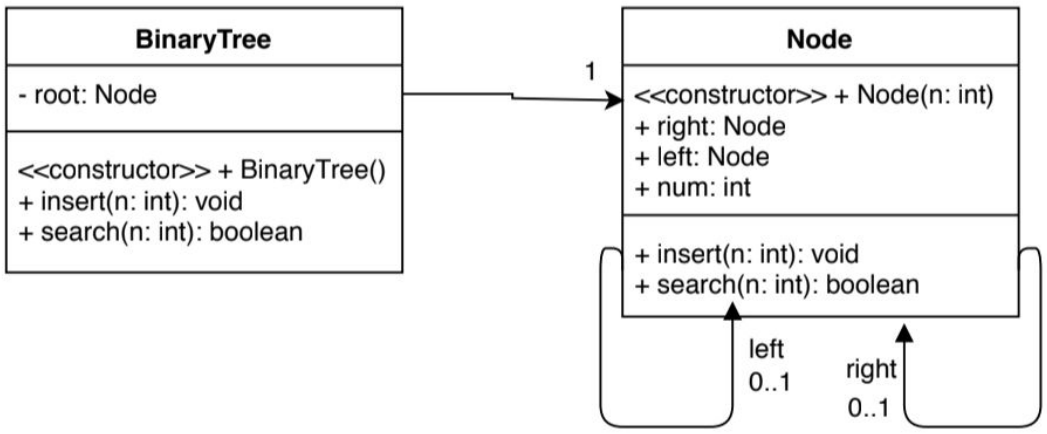
Exercise 2

a) Focus factor = Actual velocity/man days = 32/45 = 0.71.
New amount of man days = 45 + 15 + 15*0.8 = 72.
Estimated velocity = 72 * 0.71 = 51.12
The estimated velocity of this new team would be about 51 story points.

b) The company you are working for would have a set percentage for the expected first sprint focus factor. Generally 70% is what most companies use.

c) Another way to estimate story points is for the team of engineers to look through the backlog independently and to assign how long they think it would take them to complete each task. Then have the whole group together and discuss the differences and approach an appropriate compromise. This would be worse than poker because it would be harder to reach compromise numbers, especially higher numbers, when individuals can select any number instead of just a set of numbers like poker.

d)



e)

```java
/**
 *
 * @author forestedwards
 */
public class Driver {
    public static void main(String[] args) {
        // Simple example to show functionality of a binary tree:
        BinaryTree tree = new BinaryTree();
        tree.insert(5);
        tree.insert(7);
        tree.insert(4);
        tree.insert(2);
        tree.insert(1000);
        System.out.println(tree.search(1000));
        System.out.println(tree.search(77));
    }
}
```

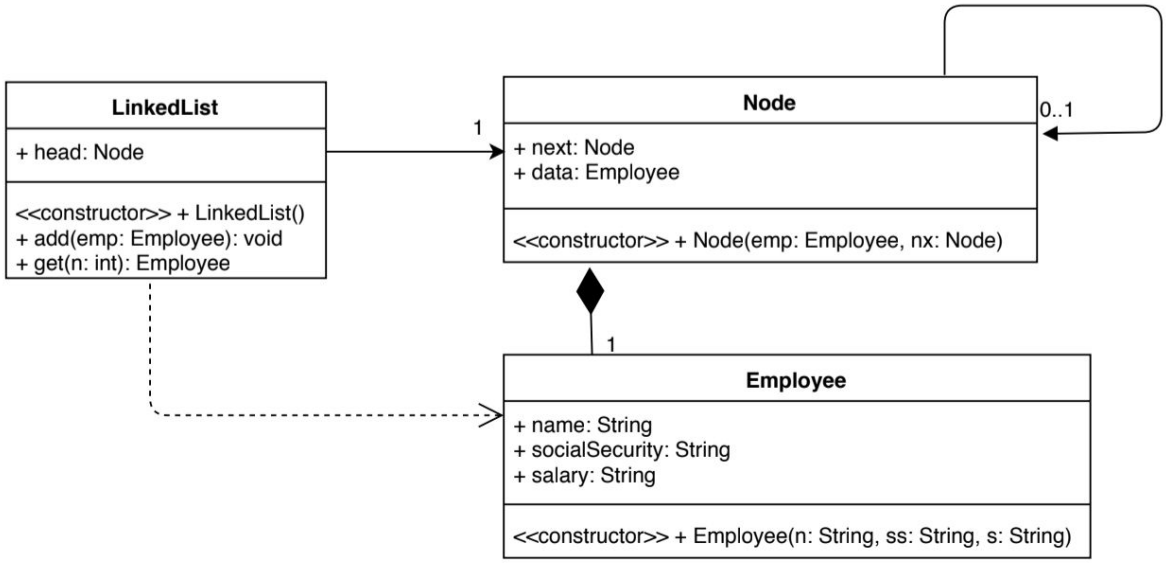
```java
/**
 *
 * @author forestedwards
 */
public class BinaryTree {
    public Node root;
    public BinaryTree() {
        root = null; // Creates an empty tree
    }
    public void insert(int n) {
        if(root == null) { // If tree is empty, make root a new node with n.
            root = new Node(n);
            return;
        }
        root.insert(n); // calls insert method in Node class.
    }
    public boolean search(int n) {
        return root.search(n); // calls search method in node class, starting with the root.
    }
}
```

```java
public class Node {
    public int num; // Stores the data
    public Node left;
    public Node right;
    public Node(int n) {
        num = n;
        left = null;
        right = null;
    }
    public void insert(int n) {
        if(n > num) { // n is greater than current node, add the new node to the right.
            if(right == null) { // right node is empty? replace it with a new node
                right = new Node(n);
            }
            else {
                right.insert(n); // Recursive call to insert the node in this right subtree
            }
        }
        else { // n is less than current node, add the new new node to the left
            if(left == null) { // left node is empty? replace it with a new node.
                left = new Node(n);
            }
            else {
                left.insert(n); // Recursive call to insert node in this left subtree
            }

        }
    }

    public boolean search(int n) {
        if(num == n) // Value is found! returns true
            return true;
        else if(n > num) { // value is greater than the value in the current node, search right subtree
            if(right == null) return false; // Right subtree is null, value does not exist in the tree, returns false.
            return right.search(n); // Recursive call to right subtree
        }
        else { // Value is greater than the value in the current node, search left subtree
            if(left == null) return false; // Left subtree is null, value does not exist in the tree, returns false.
            return left.search(n); // Recursive call to left subtree
        }
    }
}
```

f)



g)

```java
/**
 *
 * @author forestedwards
 */
public class Driver {
    public static void main(String[] args) {
        LinkedList list = new LinkedList();
        list.add(new Employee("Forest Edwards", "518-47-9593", "$1,000,000"));
        list.add(new Employee("Michael Utt", "557-65-6323", "$20,000"));
        list.add(new Employee("Hugh Janus", "434-76-2435", "$666,666"));
        Employee e = list.get(2);
        System.out.println(e.name + " " + e.socialSecurity + " " + e.salary);
    }
}
```

```java
public class LinkedList {
    public Node head;
    public LinkedList() {
        head = null;
    }
    public void add(Employee emp) {
        if(head == null) { // If linked list is empty, replace head with a new node containing the employee
            head = new Node(emp, null);
        }
        else { // Add a new node to the head of the list, and set as the new head.
            Node temp = new Node(emp, head);
            head = temp;
        }
    }
    public Employee get(int n) {
        Node iterator = head;
        int i = 0;
        while(i < n) { // loops through list until the correct node is found, or the end of the list is reached.
            if(iterator.next == null) {
                return null;
            }
            iterator = iterator.next;
            i++;
        }
        return iterator.data; // Returns the node containing the desired data.
    }
}
```

```java
public class Node {
    public Node next;
    public Employee data;
    public Node(Employee emp, Node nx) { // Each node MUST have an employee and a next.
        data = emp;
        next = nx;
    }
}
```