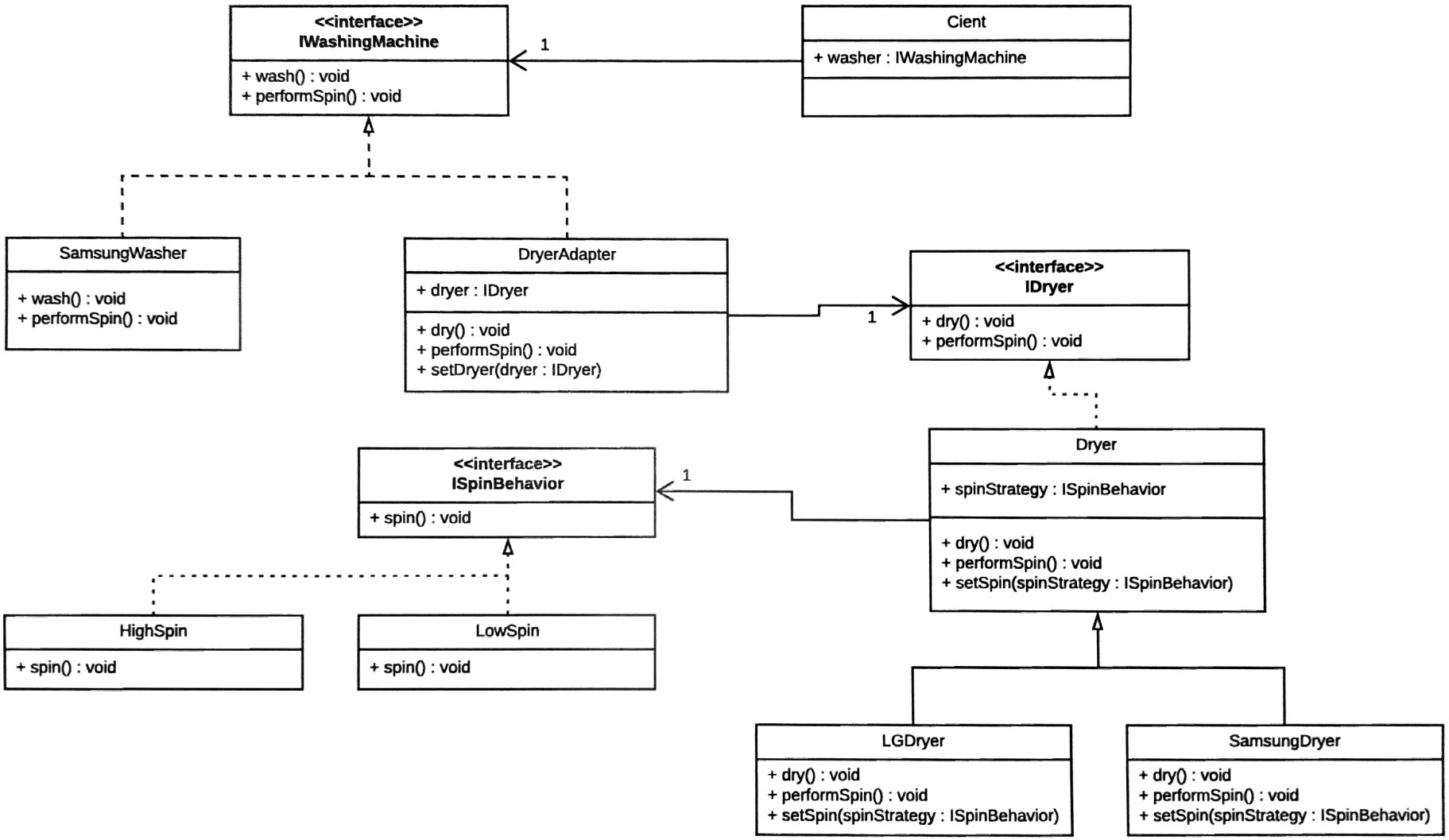
Conner Cross & Mason Dinardi

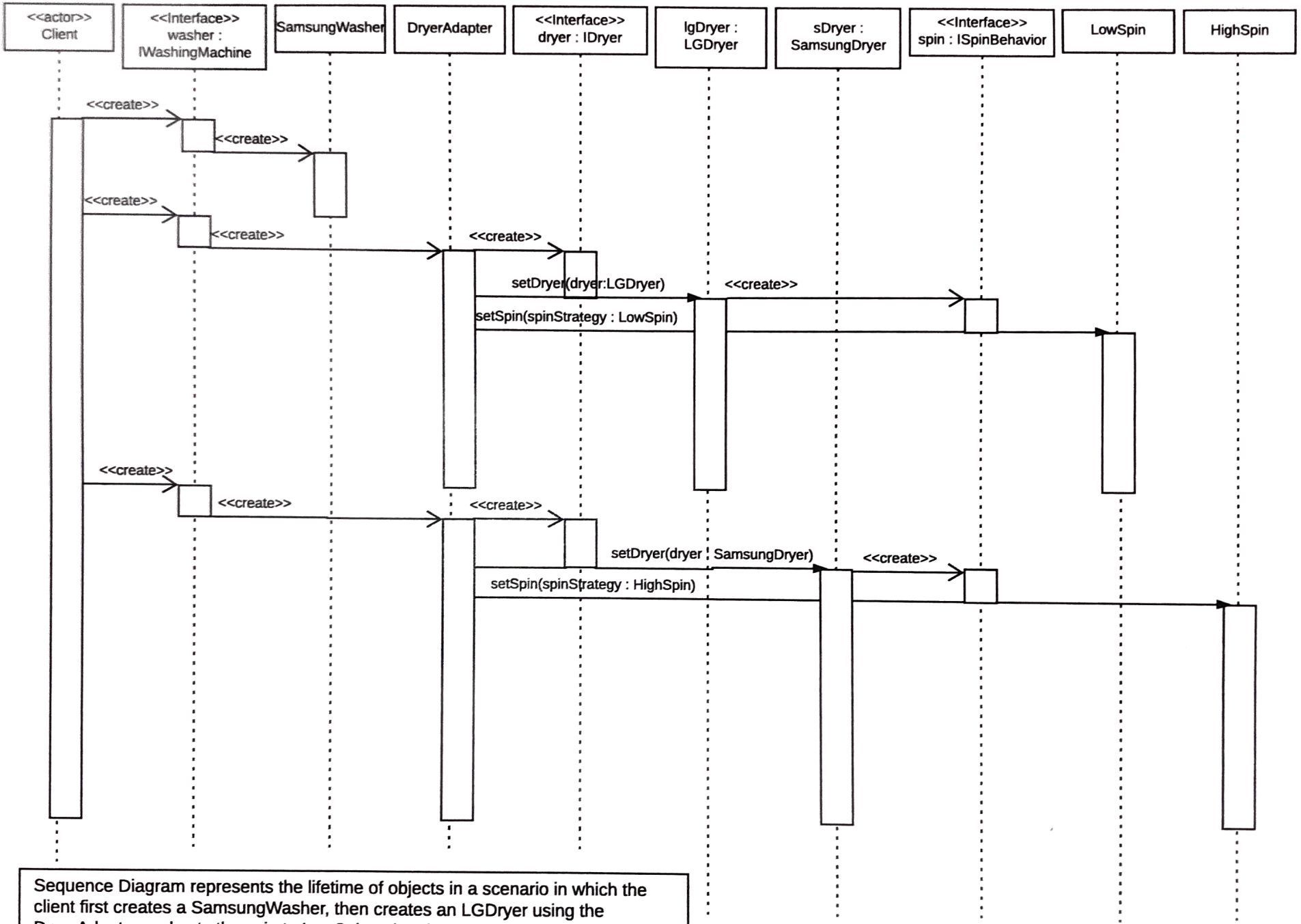
ESOF 322

Homework 3

September 26, 2019

Strategy and Adapter Patterns





Sequence Diagram represents the lifetime of objects in a scenario in which the client first creates a SamsungWasher, then creates an LGDryer using the DryerAdapter and sets the spin to LowSpin using the Strategy Pattern. Finally, the client creates a SamsungDryer and sets the spin to HighSpin.

PART 2

a.

Teammate	Available Days
Person 1	15
Person 2	15
Person 3	15
Person 4	15
Person 5	12

Estimate the velocity by taking the available man days of the first 3 members plus the available man days of the 2 new members and multiply it by the focus factor of the last 3-week sprint.

Estimated Velocity = (45 + 27)(.71)

Estimated Velocity ≈ 51.12 Story Points

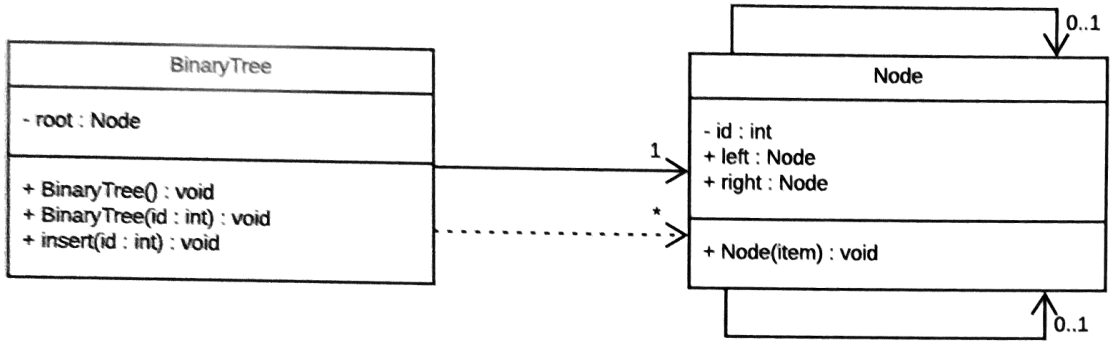
b.

Since it is a brand-new team, the team does not have an actual velocity therefore we use a focus factor of 70%.

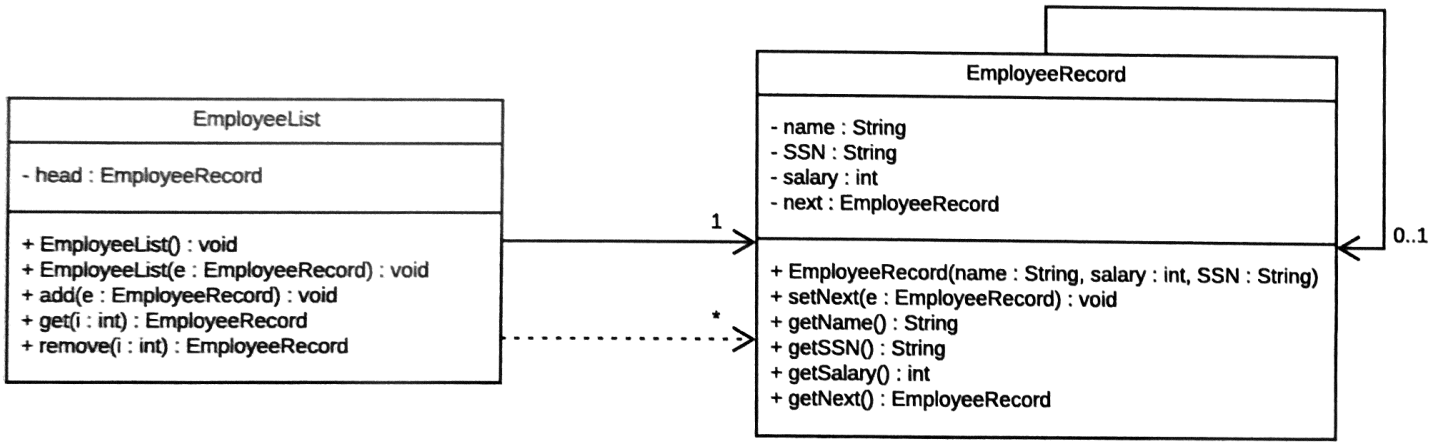
c.

Another potential way to estimate story points would be to place all items from the backlog into a random order on a table or board. The table/board can be scaled small to large and add Fibonacci numbers at the end of the exercise or it could be scaled with the Fibonacci numbers, which would be faster. Your team would then proceed to take turns moving an item up or down the scale. This sequence would repeat until the team agrees that everything is in an agreeable place. This method could be better than planning poker if you have a smaller team, because it can be quicker ordering the items from the backlog. However, with a large group this could take quite some time, considering the amount of items in the backlog would be larger, and there would be more disagreements.

BinaryTree UML



EmployeeList UML (EmployeeRecord)



```
1   // -------------------------Binary Tree Code-------------------------
2
3   public class BinaryTree  {
4       private Node root;
5
6       public void BinaryTree(int id) {
7           root = new Node(id);
8       }
9
10      public void BinaryTree() {
11          root = null;
12      }
13
14      public void insert(int id) {
15          Node temp = root;
16          while(temp != null){
17              temp = root.right;
18          }
19          temp = new Node(id);
20      }
21  }
22
23
24  class Node {
25      private int id;
26
27      Node left;
28      Node right;
29
30      public Node(int item) {
31          id = item;
32          left = null;
33          right = null;
34      }
35  }
36
37  // -------------------------End Binary Tree Code-------------------------
38
39
40
41  // -------------------------LinkedList Code-------------------------
42
43  // Our implementation of a LinkedList of type EmployeeRecord
44
45  public class EmployeeList {
46      private EmployeeRecord head;
47
48      public EmployeeList() {}
49
50      public EmployeeList(EmployeeRecord head) {
51          this.head = head;
52      }
53
54      public void add(EmployeeRecord e) {
55          if (this.head == null){
56              this.head = e;
57          } else {
58              EmployeeRecord temp = head;
59              while (temp.getNext() != null) {
60                  temp = temp.getNext();
61              }
62
63              temp.setNext(e);
64          }
65      }
66
```

```
67      public EmployeeRecord get(int i){
68          int count = 0;
69          EmployeeRecord e = head;
70
71          while(count < i) {
72              e = e.getNext();
73              count++;
74          }
75          return e;
76      }
77
78      public EmployeeRecord remove(int i) {
79          EmployeeRecord prev = this.get(i -1);
80          EmployeeRecord removed = prev.getNext();
81          prev.setNext(removed.getNext());
82
83          return removed;
84      }
85
86  }
87
88
89  class EmployeeRecord {
90      private String name;
91      private int salary;
92      private String SSN;
93      private EmployeeRecord next;
94
95      public EmployeeRecord(String name, int salary, String SSN) {
96          this.name = name;
97          this.salary = salary;
98          this.SSN = SSN;
99      }
100
101     public void setNext(EmployeeRecord e) {
102         this.next = e;
103     }
104
105     public String getName() {
106         return name;
107     }
108
109     public int getSalary() {
110         return salary;
111     }
112
113     public String getSSN() {
114         return SSN;
115     }
116
117     public EmployeeRecord getNext() {
118         return next;
119     }
120 }
121
122 // -------------------------End LinkedList Code-------------------------
```