Gebze Technical University
Computer Engineering

CSE 222 - 2019 Spring

HOMEWORK 2 REPORT

OZAN ŞELTE
161044061

Course Assistant: Ayşe Şerbetçi Turan

1 INTRODUCTION

1.1 Problem Definition

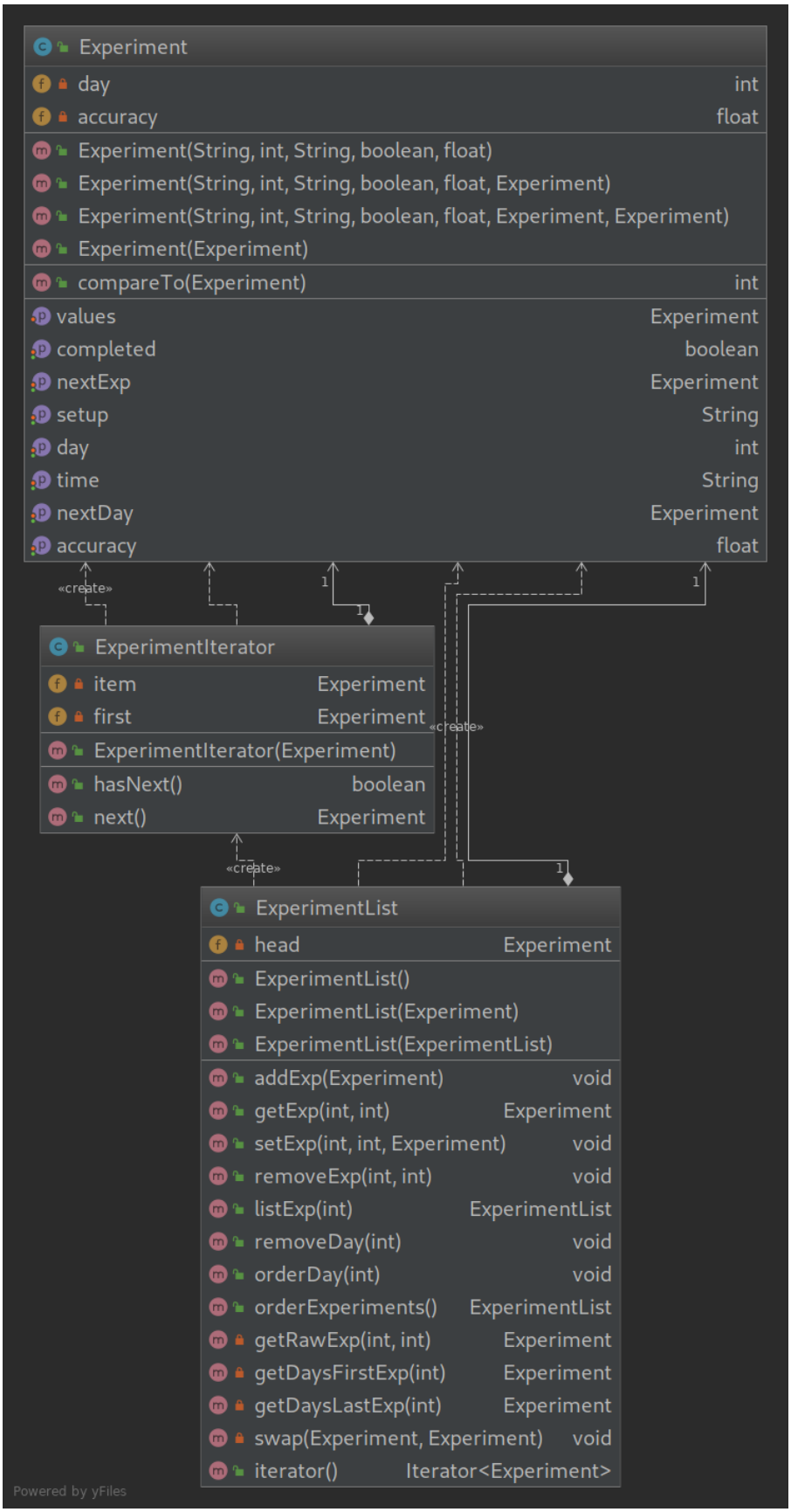
We have to create a structure to save Experiment Nodes for a machine learning application. The structure must be a Linked List and Java Collections library is prohibited. To make faster the algorithms we have to manage the Experiment's Day values too.

1.2 System Requirements

JDK, Experiment's data, Terminal

2 METHOD

2.1 Class Diagrams



2.2 Use Case Diagrams

Not required.

2.3 Algorithm Analysis

1 Experiment class' each function has T(n)=O(1) complexity.

2 ExperimentIterator class' each function has T(n)=O(1) complexity.

3 ExperimentList Class:

- ExperimentList(): T(n)=O(1).
- ExperimentList(Experiment): T(n)=O(1).
- ExperimentList(ExperimentList): Creates a new Experiment for each Experiment in given List and add it to the new List. T(n)=O(n), because List is already in a proper order.
- addExp(Experiment): It has three cases. If the Exp must be the head of the list T(n)=O(1). Must be not the lists head but a normal days first experiment it is going to be T(n)=O(n^2). Outside of these situations, it will be T(n)=O(n). Worst case, O(n^2)
- getExp(int, int): T(n)=O(n).
- setExp(int, int, Experiment): T(n)=O(n).
- removeExp(int, int): It has three cases. If the Exp is the head of the list T(n)=O(1). Not the lists head but a normal days first experiment it is going to be T(n)=O(n^2). Outside of these situations, it will be T(n)=O(n). Worst case, O(n^2).
- listExp(int): T(n)=O(n).
- removeDay(int): It uses removeExp(int,int) for the day's each Experiment. Worst case T(n)=O(n^3).
- orderDay(int): I used Bubble Sort Algorithm. T(n)=O(n^2).
- orderExperiments(): It calls orderDay(int) with all Experiments. T(n)=O(n^2).
- getRawExp(int, int): T(n)=O(n).
- getDaysFirstExp(int, int): T(n)=O(n).
- getDaysLastExp(int, int): T(n)=O(n).
- swap(Experiment, Experiment): T(n)=O(1).
- iterator(): T(n)=O(1).

2.4 Problem Solution Approach

In a generic LinkedList structure, List keeps a Node and the Node keeps a value. But in this homework the value is static, an Experiment object, so I could use a diffrent structure. A LinkedList which keeps an Experiment not a Node. The experiment is keeping the next experiment and next days references. To avoid illegal using situations, when List returning an Experiment to the user, the Experiment's is copying and assinging null to the new Experiments references.

3 RESULT

3.1 Test Cases

Day2 Exp1 – Day2 Exp2
Day4 Exp1 – Day4 Exp2
Day3 Exp1 – Day3 Exp2
Day1 Exp1 – Day1 Exp2
Day5 Exp1 – Day5 Exp2
I added two experiments for each day with unsorted **Accuracies** and **Completion Status**.

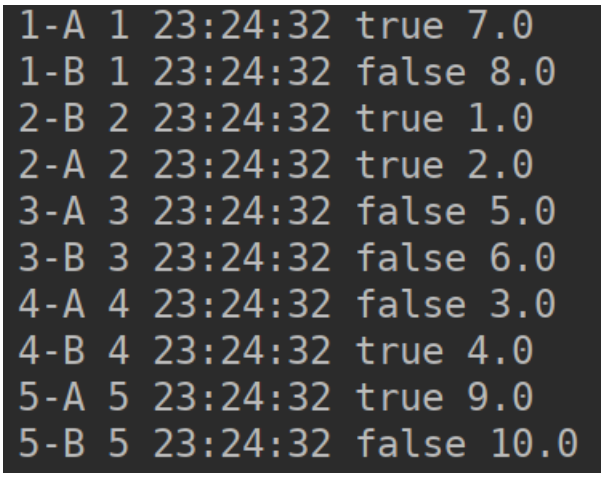
3.2 Running Results

Setup(String) Day(int) Time(String) Completed(Boolean) Accuracy(float)

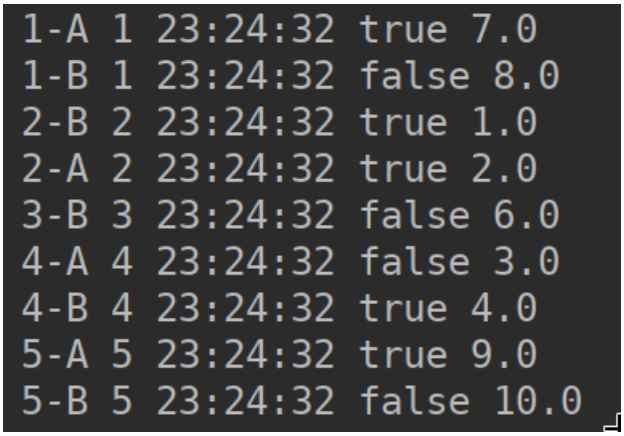
Initially, the list is as follows.

```
1-A 1 23:24:32 true 7.0
1-B 1 23:24:32 false 8.0
2-A 2 23:24:32 true 2.0
2-B 2 23:24:32 true 1.0
3-A 3 23:24:32 false 5.0
3-B 3 23:24:32 false 6.0
4-A 4 23:24:32 false 3.0
4-B 4 23:24:32 true 4.0
5-A 5 23:24:32 true 9.0
5-B 5 23:24:32 false 10.0
```

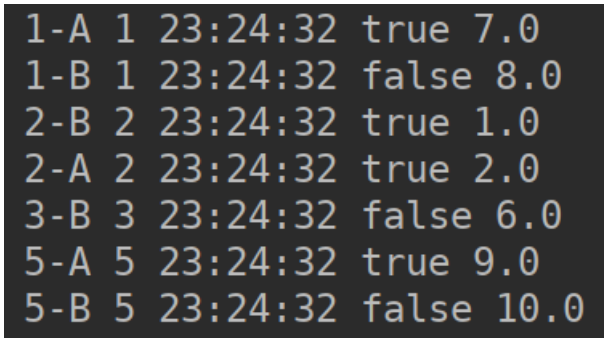
After Day-2 has ordered,



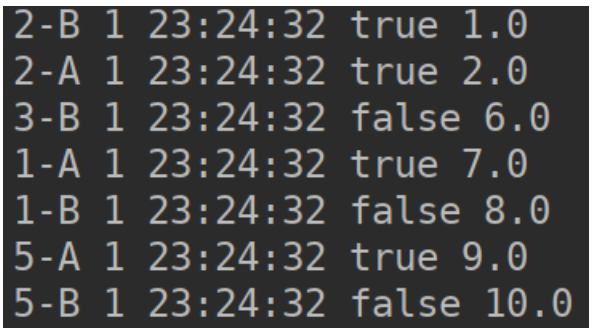
After Day-3 Exp-1 has removed,



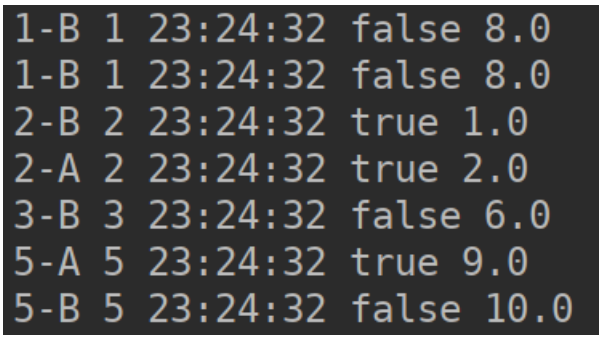
After Day-4s all experiments have deleted,



All elements have sorted,



Second Experiment has been read with getExp and assigned to the first Experiment with setExp(),



listExp() method called for Day-5