WESTERN UNIVERSITY
FACULTY OF ENGINEERING
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

SE2205b - Algorithms and Data Structures for Object-Oriented Design

Laboratory 1: ADT List

Due Date: January 31st , 2018

Student Name	Ryan Vickramasinghe
Student Number	250 923 282
Laboratory Date/Time	Wednesday, January 31, 2018, 2:30pm

Note:

Create a **private** GitHub student account

1. Go to GitHub's https://education.github.com/pack , click Get your pack.
2. Click Yes, I'm a student.
3. Type your name.
4. Verify your academic status.
5. Use the drop-down menu and click the desired email address.
6. Enter the school name.
7. Select your graduation year.
8. Describe how you plan to use GitHub.
9. Verify your application details, then click Submit Request.

If your application is approved, you'll receive a confirmation email. Applications are usually processed within a few days, but it may take longer during peak times, such as during the start of a new semester.

1 Goal

In this lab you will implement the Abstract Data Type (ADT) List using array, and then use them to simulate the interaction of the Hydra game. You will also gain new skills in JavaFX GUI applications.

2 Resources

- Notes: Unit 1.

3 Java Project Files

- NetBeans project file: Hydra.zip

4 Introduction

The ADT list is one of the basic tools for use in developing software applications. It is an ordered collection of objects that can be accessed based on their position. Before continuing the lab you should review the material in Unit 1. In particular, review the documentation of the interface ListInterface.java. While not all of the methods will be used in our applications, most of them will.

You will work on and complete a simulation for a fight with the mythical greek hydra. As legend goes, if you were to chop off the head of a hydra, two smaller heads would grow back in its place. In order for our fight to have an end, we will assume that once the size of the targeted head is small enough, no new heads will grow back in its place. The goal of this application is to determine the amount of work required to kill a hydra with a single head, when the size of the head is given as input.

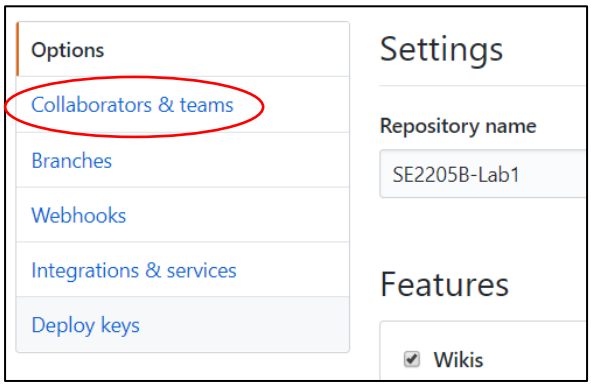
5 Create a new repository for Lab1

1. Download the lab1 file SE2205B-Lab1.zip from OWL into your course folder say "SE2205B".
2. Unzip this downloaded file to have a folder called "SE2205B-Lab1".
3. Open GitHub Desktop.
4. Run File→ Add local repository.
5. Click Choose and browse for the folder SE2205B-Lab1.
6. You will get a warning says "This directory does not appear to be a Git repository. Would you like to create a repository here instead?
7. Click "create a repository" and then click Create repository.
8. In the GitHub Desktop tool bar, click Publish repository, check the option "keep this code private", choose your GitHub account, and then click Publish repository.
9. Now a new repository called "SE2205B-Lab1" should appear into your GitHub account.
10. It is a mandatory to add the instructor and the TAs to this repository.
    a. In the GitHub account click the repository SE2205B-Lab1 and then click "Setting".

b. Click "Collaborators & teams" option.

c. At the bottom of the page and in the Collaborators section, enter the account id of the GitHub user your want to add and then click "Add collaborator".

d. You need to add the following accounts to your repo.
   - **aouda**
   - **kalhazmi2**
   - **rafadaguiar**
   - **rbarboza**



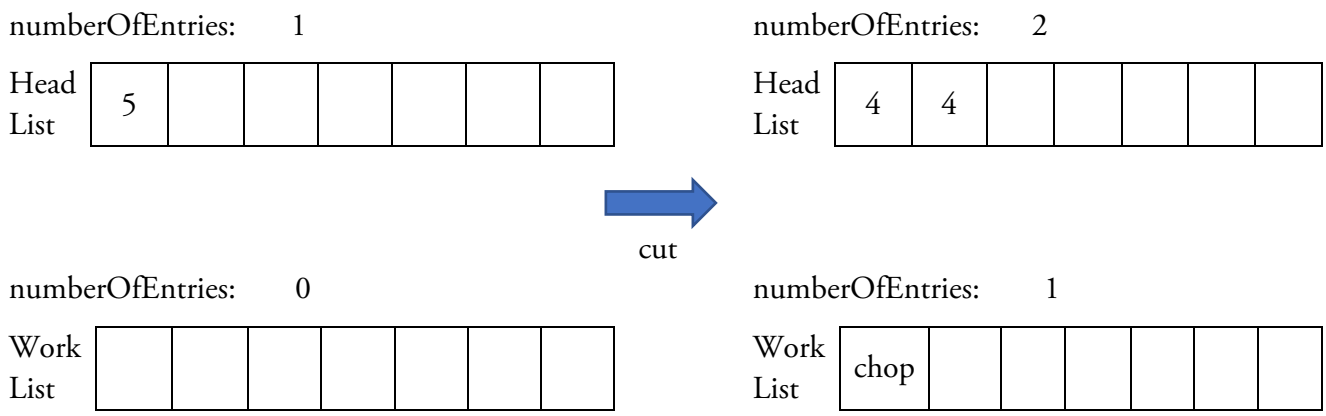
6 Pre-Lab Visualization

Hydra

We can view our hydra as a collection of heads, each of which has a size. To indicate the size we will use an integer value. Each time we cut off a head it disappear and then two new heads that are one size smaller will appear. For example, if we chop off a head of size 5, two heads of size 4 spring up instead. The exception to this rule is that a size 1 head does not grow back. (Fortunately for us, otherwise we would never finish.) A list is perfect to represent the state of the hydra as the fight continues. We need to know what heads the hydra currently has and what the size of each of the heads is, but they are in no particular order. In addition, there can be multiple heads of the same size.

We will use a second list to accumulate the answer to "How many cuts did it take to kill the hydra?". Each time we cut off a head, we will put the string "chop" into the list. Again, a list will work well. We don't care about the order of the strings in the list and we will certainly have duplicates. At the end of the simulation, the number of strings in the list will give us the answer to the question.

We want to visualize the process of the simulation as a series of steps and from that determine an algorithm. For example, if we start with one head of size 5, one cut results in the following transition.



Using the above as a model, complete the seven steps in the simulation for a hydra starting with a single head of size 3.

numberOfEntries:

Head List | 3 | | | | | | |

numberOfEntries:

Head List | 2 | 2 | | | | | |

cut

numberOfEntries:

Work List | | | | | | | |

numberOfEntries:

Work List | c | c | | | | | |

numberOfEntries:

Head List | 2 | 1 | 1 | | | | |

numberOfEntries:

Head List | 1 | 1 | 1 | 1 | | | |

cut

numberOfEntries:

Work List | c | c | | | | | |

numberOfEntries:

Work List | c | c | c | | | | |

numberOfEntries:

Head List | 1 | 1 | 1 | | | | |

numberOfEntries:

Head List | 1 | 1 | | | | | |

cut

numberOfEntries:

Work List | c | c | c | c | | | |

numberOfEntries:

Work List | c | c | c | c | c | | |

numberOfEntries:

Head List | 1 | | | | | | |

numberOfEntries:

Head List | | | | | | | |

cut

numberOfEntries:

Work List | c | c | c | c | c | c | |

numberOfEntries:

Work List | c | c | c | c | c | c | c |

Examine your sample simulation, and give an algorithm for what to do during a single step.

`Algorithm Single Step` (headList, workList)

```
Input list headList of integers, list workList of strings
Output 1
if headList[1] > 1 then
 for i := headList[1] - 1 to 0 do headList.add(i)
 headList.remove(1)
else headList[1].remove(1)
worklist.add("chop")
return 1;
```

Given your previous algorithm, come up with an algorithm that performs the simulation. Don't forget to do initialization and report the result. (follow the format given in Unit 1)

`Algorithm Hydra` (headList, workList)

```
Input list headList of integers, list workList of strings
while !headList.isEmpty do AlgorithmSingleStep
print "hydra killed in " + workList.getLength() + " steps."
```

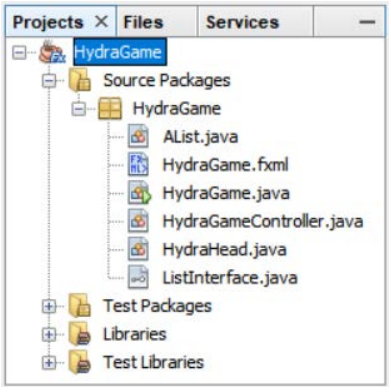
Use GitHub desktop to commit your work.

1. Click the changes tab in the left sidebar to see a list of the files that have been changed or added since the last commit.
2. Use the checkboxes to indicate which files should be part of the commit. In this activity, you'll select the "SE2205B Lab1 – Lists.pdf" file.
3. Type your commit message (Pre-lab checkpoint) in the Summary field.
4. You will notice that GitHub Desktop has already populated the commit button with the current branch. Simply click the button to commit your changes
5. Click Push origin in the menu bar to push this change to your GitHub account.

Checkpoint Once you completed the pre-lab visualizations, commit and push this workbook to your GitHub account.

7 Directed Lab Work

7.1 ADT List implementation



1. Open the NetBeans IDE and import the project "Hydra.zip". When you import it successfully you will have all files shown in the figure.
2. The class ListInterface.java is given and has all list method we discussed in Class. One method has been added to the interface, "getPosition".
3. The class AList is given as an placeholder to add the implementation of all interface methods. Simply copy the implementation code for AList given in Unit 1 and paste it into AList.java file.

4. Now, you need to implement the method getPosition(T anEntry). This method should return an integer value that represent the index of a given entry.
5. Test your project (Run→ Test Poject) and fix any error (if any). Do not processed until you have a successful build.
6. Pieces of the Hydra game simulation already exist and are in HydraGame.fxml, HydraGame.java, HydraGameController.java, and HydraHead.java. Take a look at that code now if you have not done so already.
7. Run the application and try different head size values and have fun.

Checkpoint In completing section 7.1, commit and push the following files to your GitHub account: ListInterface.java, and AList.java .

7.2 Hydra Client Modification

1. So far, Hydra simulation is working fine, however it is not yet complete. You need to use another list named "worklist" to accumulate the answer to "How many cuts did it take to kill the hydra?".
2. Modify the HydraGameController.java file, so that, each time you cut off a head, you will put the string "chop" into this list. At the end of the simulation, instead of displaying the message ""Good Job! - Play Again", you need to display the message "Good Job, you have made ### times of cuts", where ### is the number of strings in the worklist.
3. Run the application and try different head size values and check the output.

Checkpoint In completing section 7.2, commit and push HydraGameController.java file to your GitHub account.

---

8 Hand in

1. Using NetBeans IDE, export the Hydra project to ZIP and name it yourUwoId_SE2205B_Lab1.zip. For example, if yourUWOId is aouda then name the archive file as aouda_SE2205B_Lab1.zip. Use File → Export Project → to ZIP, then enter the zip file name. (do not forget to add the extension .zip)
2. Submit (1) this zip file along with (2) your solution workbook (this PDF file), using OWL assignment link at the due date mentioned above