Homework 16 - Keeping up with the Assassins

Authors: Hussain Miyaziwala, Jack Kelly, Vince Li, Andrew Chafos, Davis Williams, Shishir Bhat

Problem Description

Assassins is a live-action game in which players try to eliminate each other using mock weapons in an effort to become the last surviving player.

This game is popular on college campuses and is played by thousands of students all over the world. Game hosts start a round by assigning each player a unique target. Every player has target and are themselves a target for someone else. When a player eliminates a target, usually by marking them with a pen or some tape, they get their victim's target. The game ends when there is only one player remaining.

This sort of behavior can be modeled with a linked list! To be exact, this scenario requires a circularly linked list (as every player has a target and is a target), but for the sake of this homework we will **ONLY be asking you to implement a regular doubly linked list**.

Solution Description

Fill out the class called `MyLinkedList.java` that contains your doubly linked list implementation. We have provided an interface, `SimpleList.java`, that `MyLinkedList.java` must implement. **Look at the descriptions in `SimpleList.java` to understand what each method requires.** You may create additional methods and fields for your linked list as needed.

Your linked list must use the provided `Node` inner class to implement the list, and must update both the `head` and `tail` fields when appropriate.

Testing

For this assignment you will be expected to rigorously test your linked list implementation. We encourage you to write your own tester class and test various edge cases. Here is an example tester class:

```java
public class Test {
    public static void main(String[] args){
        MyLinkedList<Integer> mll = new MyLinkedList<>();
        for (int i = 0; i < 10; i++) {
           mll.add(i);
        }
        for (int i = 0; i < 10; i++) {
          System.out.println(mll.get(i));
        }
    }
}
```

You have also been provided with a tester file called `Tester.java` which contains more simple tests of every method.

Rubric

- [20] `add(int index, E element)`

    - [3] Correctly throws exceptions
    - [17] Functionality
- [12] `add(E element)`
    - [2] Correctly throws exceptions
    - [10] Functionality
- [12] `get(int index)`
    - [2] Correctly throws exceptions
    - [10] Functionality
- [12] `remove(int index)`
    - [2] Correctly throws exceptions
    - [10] Functionality
- [12] `removeElement(E element)`
    - [2] Correctly throws exceptions
    - [10] Functionality
- [12] `contains(E element)`
    - [2] Correctly throws exceptions
    - [10] Functionality
- [5] `isEmpty()`
- [5] `clear()`
- [5] `size()`
- [5] `toArray()`

Import Restrictions

You may not import anything for this homework assignment.

Feature Restrictions

There are a few features and methods in Java that overly simplify the concepts we are trying to teach or break our auto grader. For that reason, do not use any of the following in your final submission:

- var (the reserved keyword)
- System.exit

Checkstyle

For each of your homework assignments we will run checkstyle and deduct one point for every checkstyle error. For this homework the checkstyle cap is 100, meaning you can lose a maximum of 100 points on this assignment due to style errors.

If you encounter trouble running checkstyle, check Piazza for a solution and/or ask a TA as soon as you can! You can run checkstyle on your code by using the jar file found on Canvas (under Files/Resources). To check the style of your code, place the checkstyle jar file in the same folder as your java files, then run java -jar checkstyle-6.2.2.jar *.java. You will be responsible for running checkstyle on ALL of your code.

Collaboration

Collaboration Statement

To ensure that you acknowledge collaboration and give credit where credit is due, **we require that you place a collaboration statement as a comment at the top of at least one java file that you submit**. That collaboration statement should say either:

I worked on the homework assignment alone, using only course materials.

or

In order to help learn course concepts, I worked on the homework with [give the names of the people you worked with], discussed homework topics and issues with [provide names of people], and/or consulted related material that can be found at [cite any other materials not provided as course materials for CS 1331 that assisted your learning].

Recall that comments are special lines in Java that begin with `//`.

Turn-In Procedure

Submission

To submit, upload the files listed below to the corresponding assignment on Gradescope:

- MyLinkedList.java

Make sure you see the message stating "HW16 submitted successfully". From this point, Gradescope will run a basic autograder on your submission as discussed in the next section.

You can submit as many times as you want before the deadline, so feel free to resubmit as you make substantial progress on the homework. We will only grade your last submission: be sure to **submit every file each time you resubmit**.

Gradescope Autograder

For each submission, you will be able to see the results of a few basic test cases on your code. Each test typically corresponds to a rubric item, and the score returned represents the performance of your code on those rubric items only. If you fail a test, you can look at the output to determine what went wrong and resubmit once you have fixed the issue.

The Gradescope tests serve two main purposes:

1. Prevent upload mistakes (e.g. forgetting checkstyle, non-compiling code)
2. Provide basic formatting and usage validation

In other words, the test cases on Gradescope are by no means comprehensive. Be sure to thoroughly test your code by considering edge cases and writing your own test files. You also should avoid using Gradescope to compile, run, or checkstyle your code; you can do that locally on your machine.

Other portions of your assignment can also be graded by a TA once the submission deadline has passed, so the output on Gradescope may not necessarily reflect your grade for the assignment.

Important Notes (Don't Skip)

- Non-compiling files will receive a 0 for all associated rubric items
- Do not submit `.class` files.
- Test your code in addition to the basic checks on Gradescope
- Submit every file each time you resubmit
- Read the "Allowed Imports" and "Restricted Features" to avoid losing points
- Check on Piazza for all official clarifications