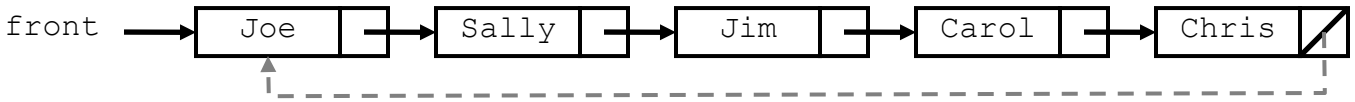
CSE 145
Assignment 2
Tag

This program focuses on implementing a linked list. Turn in a file named `TagManager.java` to Grade-it. You will need the support file `PlayerNode.java` found on the assignment page on Canvas.

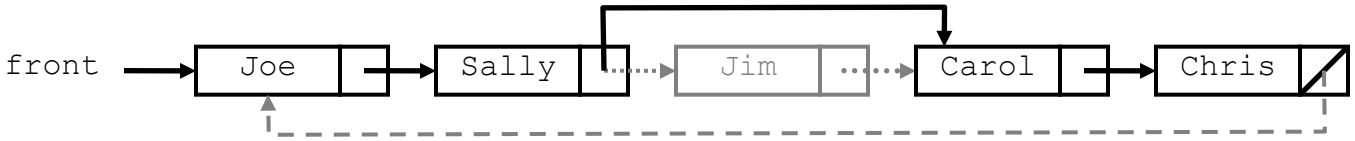
Program Description:

This program is going to manage a special version of a game of tag. Each person playing has a particular target that he/she is trying to tag. However, with this version of tag, each person knows only who they are trying to tag; they don't know who is trying to tag them, nor do they know whom the other people are trying to tag.

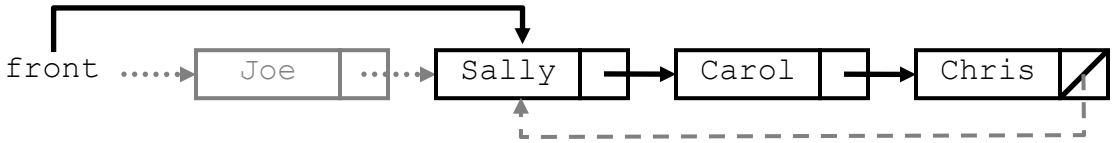
This game of tag is played as follows: You start out with a group of people who want to play the game. For example, let's say that there are five people playing named Carol, Chris, Jim, Joe, Sally. A circular chain of players (called the "game ring" in this program) is randomly established. For example, we might decide Joe should try to tag Sally, Sally should try to tag Jim, Jim should try to tag Carol, Carol should try to tag Chris, and Chris should try to tag Joe. (In the actual linked list that implements this game ring, Chris's `next` reference would be `null`. But conceptually we can think of it as though the next person after Chris is Joe, the front person in the list.) Here is a picture of this "game ring":



When someone is tagged, the links need to be changed to "skip" that person. For example, suppose that Jim is tagged by Sally. Sally needs a new person to tag, so we give her Jim's target: Carol. The game ring becomes:



If the first person in the game ring is tagged, the front of the list must adjust. If Chris tags Joe, the list becomes:



You will write a class `TagManager` that keeps track of who is trying to tag whom and the history of who tagged whom. You will maintain two linked lists: a list of people currently in the game (the "game ring") and a list of those who are out (the "history"). As people are tagged, you will move them from the game ring to the history by rearranging links between nodes. The game ends when only one node remains in the game ring, representing the winner.

Sample Log of Execution (user input underlined):

Your program should exactly reproduce the format and general behavior demonstrated in this log, although you may not exactly recreate this scenario because of the shuffling of the names that your main will perform.

```
Current game ring:
  Erica Kane is trying to tag Ruth Martin
  Ruth Martin is trying to tag Jackson Montgomery
  Jackson Montgomery trying to tag Bobby Warner
  Bobby Warner is trying to tag Joe Martin
  Joe Martin is trying to tag Anita Santos
  Anita Santos is trying to tag Tad Martin
  Tad Martin is trying to tag Phoebe Wallingford
  Phoebe Wallingford is trying to tag Erica Kane
Current history:

next person? Ruth Martin

Current game ring:
  Erica Kane is trying to tag Jackson Montgomery
  Jackson Montgomery is trying to tag Bobby Warner
  Bobby Warner is trying to tag Joe Martin
  Joe Martin is trying to tag Anita Santos
  Anita Santos is trying to tag Tad Martin
  Tad Martin is trying to tag Phoebe Wallingford
  Phoebe Wallingford is trying to tag Erica Kane
Current history:
  Ruth Martin was tagged by Erica Kane

next person? Ruth Martin
Ruth Martin is already out.

Current game ring:
  Erica Kane is trying to tag Jackson Montgomery
  Jackson Montgomery is trying to tag Bobby Warner
  Bobby Warner is trying to tag Joe Martin
  Joe Martin is trying to tag Anita Santos
  Anita Santos is trying to tag Tad Martin
  Tad Martin is trying to tag Phoebe Wallingford
  Phoebe Wallingford is trying to tag Erica Kane
Current history:
  Ruth Martin was tagged by Erica Kane

next person? bobby warner

Current game ring:
  Erica Kane is trying to tag Jackson Montgomery
  Jackson Montgomery is trying to tag Joe Martin
  Joe Martin is trying to tag Anita Santos
  Anita Santos is trying to tag Tad Martin
  Tad Martin is trying to tag Phoebe Wallingford
  Phoebe Wallingford is trying to tag Erica Kane
Current history:
  Bobby Warner was tagged by Jackson Montgomery
  Ruth Martin was tagged by Erica Kane

next person? ERICa kaNE

Current game ring:
  Jackson Montgomery is trying to tag Joe Martin
  Joe Martin is trying to tag Anita Santos
  Anita Santos is trying to tag Tad Martin
  Tad Martin is trying to tag Phoebe Wallingford
  Phoebe Wallingford is trying to tag Jackson Montgomery
Current history:
  Erica Kane was tagged by Phoebe Wallingford
  Bobby Warner was tagged by Jackson Montgomery
  Ruth Martin was tagged by Erica Kane

next person? ANITA SANTOS
```

(continued)

```
Current game ring:
  Jackson Montgomery is trying to tag Joe Martin
  Joe Martin is trying to tag Tad Martin
  Tad Martin is trying to tag Phoebe Wallingford
  Phoebe Wallingford is trying to tag Jackson Montgomery
Current history:
  Anita Santos was tagged by Joe Martin
  Erica Kane was tagged by Phoebe Wallingford
  Bobby Warner was tagged by Jackson Montgomery
  Ruth Martin was tagged by Erica Kane

next person? phoebe wallingford

Current game ring:
  Jackson Montgomery is trying to tag Joe Martin
  Joe Martin is trying to tag Tad Martin
  Tad Martin is trying to tag Jackson Montgomery
Current history:
  Phoebe Wallingford was tagged by Tad Martin
  Anita Santos was tagged by Joe Martin
  Erica Kane was tagged by Phoebe Wallingford
  Bobby Warner was tagged by Jackson Montgomery
  Ruth Martin was tagged by Erica Kane

next person? Joe Martin

Current game ring:
  Jackson Montgomery is trying to tag Tad Martin
  Tad Martin is trying to tag Jackson Montgomery
Current history:
  Joe Martin was tagged by Jackson Montgomery
  Phoebe Wallingford was tagged by Tad Martin
  Anita Santos was tagged by Joe Martin
  Erica Kane was tagged by Phoebe Wallingford
  Bobby Warner was tagged by Jackson Montgomery
  Ruth Martin was tagged by Erica Kane

next person? jackson montgomery

Game was won by Tad Martin
Final history is as follows:
  Jackson Montgomery was tagged by Tad Martin
  Joe Martin was tagged by Jackson Montgomery
  Phoebe Wallingford was tagged by Tad Martin
  Anita Santos was tagged by Joe Martin
  Erica Kane was tagged by Phoebe Wallingford
  Bobby Warner was tagged by Jackson Montgomery
  Ruth Martin was tagged by Erica Kane
```

Implementation Details:

You must use our node class `PlayerNode` (provided on Canvas) which has the following implementation:

```
public class PlayerNode {
    public String name;         // this person's name
    public String tagger;       // name of who tagged this person (null if still playing)
    public PlayerNode next;   // next node in the list

    public PlayerNode(String name) { ... }
    public PlayerNode(String name, PlayerNode next) { ... }
}
```

For this assignment we are going to specify exactly what fields you can have in your Tag`Manager` class. You must have exactly the following two fields; you are not allowed to have any others:

- a reference to the front node of the game ring
- a reference to the front node of the history (`null` if empty)

Implementation Details (continued):

Your class should have the following public methods.

`public TagManager(List<String> names)`

In this constructor you should initialize a new tag manager over the given list of people. Your constructor should not save the `List<String>` itself as a field, nor modify the list; but instead it should build your own game ring of linked nodes that contains these names in the same order. For example, if the list contains `["John", "Sally", "Fred"]`, the initial game ring should represent that John is trying to tag Sally who is trying to tag Fred who is trying to tag John (in that order). You may assume that the names are non-empty, non-`null` strings and that there are no duplicates.

You should throw an `IllegalArgumentException` if the list is `null` or has a size of 0.

`public void printGameRing()`

In this method you should print the names of the people in the game ring, one per line, indented by two spaces, as "**name** is trying to tag **name**". The behavior is unspecified if the game is over. For the names on the first page, the initial output is:

```
  Joe is trying to tag Sally
  Sally is trying to tag Jim
  Jim is trying to tag Carol
  Carol is trying to tag Chris
  Chris is trying to tag Joe
```

`public void printHistory()`

In this method you should print the names of the people in the history, one per line, with each line indented by two spaces, with output of the form "**name** was tagged by **name**". It should print the names in the opposite of the order in which they were tagged (most recently tagged first, then next more recently tagged, and so on). It should produce no output if the history is empty. For example, from the previous names, if Jim is tagged, then Chris, then Carol, the output is:

```
  Carol was tagged by Sally
  Chris was tagged by Carol
  Jim was tagged by Sally
```

`public boolean gameRingContains(String name)`

In this method you should return `true` if the given name is in the current game ring and `false` otherwise. It should ignore case in comparing names; for example, if passed `"salLY"`, it should match a node with a name of `"Sally"`.

`public boolean historyContains(String name)`

In this method you should return `true` if the given name is in the current history and `false` otherwise. It should ignore case in comparing names; for example, if passed `"CaRoL"`, it should match a node with a name of `"Carol"`.

`public boolean isGameOver()`

In this method you should return `true` if the game is over (i.e., if the game ring has just one person) and `false` otherwise.

`public String winner()`

In this method you should return the name of the winner of the game, or `null` if the game is not over.

`public void tag(String name)`

In this method you should record the tagging of the person with the given name, transferring the person from the game ring to the front of the history. This operation should not change the relative order of the game ring (i.e., the links of who is trying to tag whom should stay the same other than the person who is being tagged/removed). Ignore case in comparing names. A node remembers who tagged the person in its `tagger` field. It is your code's responsibility to set that field's value.

You should throw an `IllegalStateException` if the game is over, or an `IllegalArgumentException` if the given name is not part of the game ring (if both of these conditions are true, the `IllegalStateException` takes precedence).

The `tag` method is the hardest one, so write it last. Use the jGRASP debugger and `println` statements liberally to debug problems in your code. You will likely have a lot of `NullPointerException` errors, infinite loops, etc. and will have a very hard time tracking them down unless you are comfortable with debugging techniques and jGRASP.

For full credit, every method should loop over the list at most one time. You should not need to loop over the game ring more than one time for any method.

jGRASP's Debugger:

In jGRASP's debugger you can use a structure viewer to see what your list looks like by dragging one of your fields from the debug window outside the window. By default the viewer won't show you the `name` in each node (it'll show a "?" mark instead). Fix this by clicking the wrench icon, then in the "Value Expressions" box, type: **`_node_.name`**

Click OK, and you should see the names in the nodes. You can also drag the Width scrollbar to see the names better.

Other Constraints and Tips:

This is meant to be an exercise in linked list manipulation. As a result, you must adhere to the following rules:

- You must use the `PlayerNode` class for your lists. You are not allowed to modify it.
- You may not construct any arrays, `ArrayList`s, `LinkedList`s, stacks, queues, sets, maps, or other data structures; you must use linked nodes. You may not modify the list of `String`s passed to your constructor.
- If there are *N* names in the list of `String`s passed to your constructor, you should create exactly *N* new `PlayerNode` objects in your constructor. As people are tagged, you have to move their node from the game ring to the history by changing references, without creating any new node objects.

Your constructor will create the initial game ring of nodes, and then your class may not create any more nodes for the rest of the program. You are allowed to declare as many local variables of type `PlayerNode` (like `current` from lecture) as you like. `PlayerNode` variables are not node objects and therefore don't count against the limit of *n* nodes.

You should write some of your own testing code. This will be in your main class that your write. You should test each of your methods and think of each of the special cases with LinkedLists.

A word of caution: Some students try to store the game ring in a "circular" linked list, with the list's final element storing a `next` reference back to the front element. But we discourage you from implementing the program in this way; we strongly suggest that you follow the normal convention of having `null` in the `next` field of the last node. Most novices find it difficult to work with a circular list; it is easy to end up with infinite loops or other bugs. There is no need to use a circular list, because you can always get back to the front via the fields of your `TagManager`. If you feel strongly that you want to use a circular list, you are allowed to do so, but it is likely to make the program harder to write.

Style Guidelines and Grading:

Part of your grade will come from appropriately utilizing linked lists and nodes as described previously. Redundancy is another major grading focus; some methods are very similar, and you should avoid repeated logic as much as possible.

You should follow good general style guidelines such as: making fields `private` and avoiding unnecessary fields; appropriately using control structures like loops and `if`/`else` statements; properly using indentation, good variable names and proper types; and not having any lines of code longer than 100 characters. Do not use recursion on this assignment.

Comment your code descriptively in your own words at the top of your class, each method, and on complex sections of your code. Comments should explain each method's behavior, parameters, return, pre/post-conditions, and exceptions. For reference, our solution is around 130 lines long (60 "substantive" lines).

Extra Credit Opportunities (Optional):

You can choose to do as many or as few of these as you would like. Each one completed successfully and in good style will earn some extra credit points towards this assignment.

1. For extra credit, you may take on the added challenge of solving this as a circular list instead of the traditional LinkedList that we have looked at in class. You must make sure that all of the methods still work as intended while using the circular list.

Grading:

Style	10
Constructor	10
printGameRing	12.5
printHistory	12.5
gameRingContains	7.5
historyContains	7.5
isGameOver and winner	5
tag	15
Overall program functionality	20
Total:	**100**