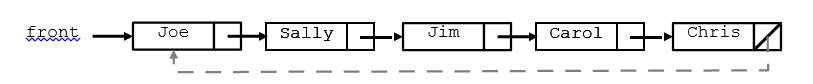
Assignment 12 – Tag Manager

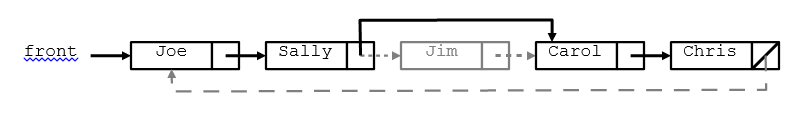
### Overview (what this program does)

"Tag" is a real-life game often played on the playground. In some versions of tag, each person playing has a particular target that he/she is trying to tag. Generally tagging a person means finding them on campus in public and acting on them in some way, such as saying, "You're tagged," or squirting them with a water gun, or touching them. One of the things that makes the game more interesting to play in real life is that initially 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.

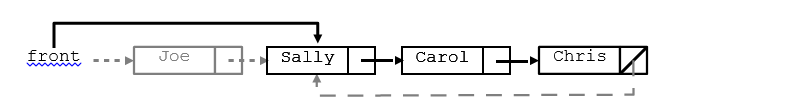
The game of tag is played as follows: You start out with a group of people who want to play the game. For example, let's say that there are five people playing named Carol, Chris, Jim, Joe, Sally. A circular chain of tag targets (called the "tag ring" in this program) is randomly established. For example, we might decide Joe should stalk Sally, Sally should stalk Jim, Jim should stalk Carol, Carol should stalk Chris, and Chris should stalk Joe. Here is a picture of this "tag 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 target, so we give her Jim's target: Carol. The tag ring becomes:



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



The game ends when only one person remains in the tag ring. That person is the declared the winner.

### Learning Objectives

* Practice previous concepts
* Implement a Linked List data structure in context

### External Requirements

Part of your program's score will come from its "external correctness." External correctness measures whether the output matches exactly what is expected. We are very picky about the output matching exactly. Every character and space must match. Use the **output comparison tool** to ensure that your output matches exactly. Programs that do not compile will receive no external correctness points. The specific requirements for external correctness are detailed below.

* You will be writing part of a program that will keep track of who is stalking whom and the history of who tagged whom. A sample of the program’s execution is shown on the next page. The bold and underlined text is the user’s input and will not be displayed that way in your console.
* As people are tagged, you will move them from the tag ring to the losers list.
* The game ends when only one node remains in the tag ring, representing the winner.

|  |  |
| --- | --- |
| Current tag ring:  Erica Kane is stalking Ruth Martin  Ruth Martin is stalking Jackson Montgomery  Jackson Montgomery is stalking Bobby Warner Bobby Warner is stalking Joe Martin  Joe Martin is stalking Anita Santos  Anita Santos is stalking Tad Martin  Tad Martin is stalking Phoebe Wallingford  Phoebe Wallingford is stalking Erica Kane  Current losers:  next victim? **Ruth Martin**  Current tag ring:  Erica Kane is stalking Jackson Montgomery Jackson Montgomery is stalking Bobby Warner Bobby Warner is stalking Joe Martin  Joe Martin is stalking Anita Santos  Anita Santos is stalking Tad Martin  Tad Martin is stalking Phoebe Wallingford  Phoebe Wallingford is stalking Erica Kane  Current losers:  Ruth Martin was tagged by Erica Kane  next victim? **Ruth Martin**  Ruth Martin is already tagged.  Current tag ring:  Erica Kane is stalking Jackson Montgomery Jackson Montgomery is stalking Bobby Warner Bobby Warner is stalking Joe Martin  Joe Martin is stalking Anita Santos  Anita Santos is stalking Tad Martin  Tad Martin is stalking Phoebe Wallingford  Phoebe Wallingford is stalking Erica Kane  Current losers:  Ruth Martin was taged by Erica Kane  next victim? **bobby warner**  Current tag ring:  Erica Kane is stalking Jackson Montgomery Jackson Montgomery is stalking Joe Martin Joe Martin is stalking Anita Santos  Anita Santos is stalking Tad Martin  Tad Martin is stalking Phoebe Wallingford  Phoebe Wallingford is stalking Erica Kane  Current losers:  Bobby Warner was tagged by Jackson Montgomery  Ruth Martin was tagged by Erica Kane  next victim? **ERICa kaNE**  Current tag ring:  Jackson Montgomery is stalking Joe Martin  Joe Martin is stalking Anita Santos  Anita Santos is stalking Tad Martin  Tad Martin is stalking Phoebe Wallingford  Phoebe Wallingford is stalking Jackson Montgomery  Current losers:  Erica Kane was tagged by Phoebe Wallingford Bobby Warner was tagged by Jackson Montgomery Ruth Martin was tagged by Erica Kane  next victim? **ANITA SANTOS** | *(continued)*  Current tag ring:  Jackson Montgomery is stalking Joe Martin  Joe Martin is stalking Tad Martin  Tad Martin is stalking Phoebe Wallingford  Phoebe Wallingford is stalking Jackson Montgomery  Current losers:  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 victim? **phoebe wallingford**  Current tag ring:  Jackson Montgomery is stalking Joe Martin  Joe Martin is stalking Tad Martin  Tad Martin is stalking Jackson Montgomery  Current losers:  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 victim? **Barack Obama**  Unknown person.  Current tag ring:  Jackson Montgomery is stalking Joe Martin  Joe Martin is stalking Tad Martin  Tad Martin is stalking Jackson Montgomery  Current losers:  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 victim? **Joe Martin**  Current tagdea ring:  Jackson Montgomery is stalking Tad Martin  Tad Martin is stalking Jackson Montgomery  Current losers:  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 victim? **jackson montgomery**  Game was won by Tad Martin  Final losers are 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 |

### Internal Requirements

Internal correctness means that your program uses the programming elements and structure that are detailed in the list of requirements below.

* Your files must be called TagManager.java that will keep track of who is stalking whom and the history of who tagged whom.
* A client program has been written for you called TagMain. It reads a file of names, shuffles the names, and constructs an object of your class TagManager. This main program then asks the user for the names of each victim to tag until there is just one player left untagged (at which point the game is over and the last remaining player wins). TagMain calls methods of the TagManager class to carry out the tasks involved in administering the game.
* Your TagManager will maintain two linked lists: a list of people currently untagged (the "tag ring") and a list of those who have already been tagged (the "losers").
* You may only have two fields in your TagManager object:
  + a reference to the front node of the tag ring
  + a reference to the front node of the losers (null if empty)
* The two linked lists are each (separately) comprised of TagNode objects.
* You must use our node class TagNode (provided on the course website) which has the following implementation:

public class TagNode

{

public String **name**; // this person's name

public String **tagger**; // name of who tagged this person

// (null if untagged)

public TagNode **next**; // next node in the list

public TagNode(String name) { **...** }

public TagNode(String name, TagNode next) { **...** }

}

* You are not allowed to modify TagNode class.
* Your TagManager is to implement the methods listed in the box on the next page.
* **You may not construct any arrays, ArrayLists, LinkedLists, Stacks, Queues, Sets, Maps, or other data structures; you must use linked nodes.**
* You may not modify the list of Strings passed to your constructor.
* If there are N names in the list of Strings passed to your constructor, you should create exactly N new TagNode objects in your constructor, and then your class may not create any more new nodes for the rest of the program.
* You are allowed to declare as many local variables of type TagNode as you like. TagNode variables are not new node objects and therefore don't count against the limit of *n* nodes.

**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 tag ring of linked nodes that contains these names in the same order. For example, if the list contains ["John", "Sally", "Fred"], the initial tag ring should represent that John is stalking Sally who is stalking Fred who is stalking 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 printTagRing()**

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

Joe is stalking Sally

Sally is stalking Jim

Jim is stalking Carol

Carol is stalking Chris

Chris is stalking Joe

**public void printLosers()**

In this method you should print the names of the people in the ‘losers’, 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 ‘losers’ 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 tagRingContains(String name)**

In this method you should return true if the given name is in the current tag 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 losersContains(String name)**

In this method you should return true if the given name is in the current ‘losers’ list 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 tag 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 tag ring to the front of the ‘losers’ list. This operation should not change the relative order of the tag ring (i.e., the links of who is tagging 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 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 tag ring (if both of these conditions are true, the IllegalStateException takes precedence).

* As people are tagged, you have to move their node from the tag ring to the losers list by changing references, without creating any new node objects.
* A word of caution: Some students try to store the tag ring in a "circular" linked list, with the list's final element storing a next reference back to the front element. We HIGHLY 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.
* For full credit, every method's runtime should be at worst O(N), where N is the number of people in your linked lists.

### Style Requirements

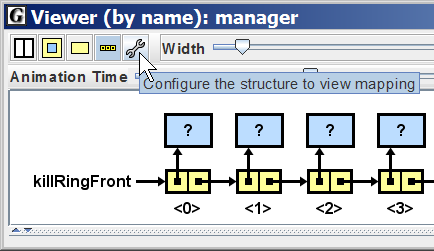
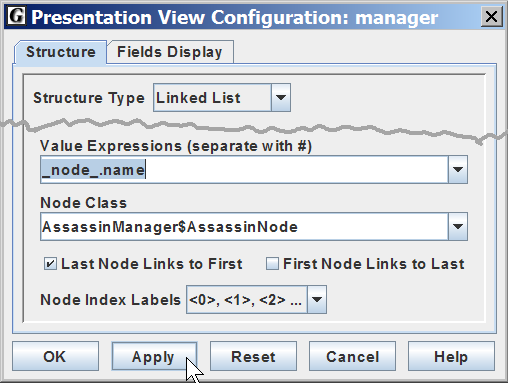
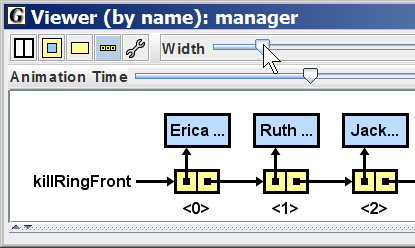
For EACH of the files you turn in you must have:

* A comment header for your class.
* All class variables and methods must have associated Javadoc comments.
* Tricky code and hard-coded values must be explained with in-line comments.
* Code is neatly indented and spaced.
* Give meaningful names to methods/variables, and follow Java's naming standards.
* Limit the lengths of all lines in your program to fewer than 100 characters.
* Remember, we will also deduct for magic numbers.
* Also remember that each class constant requires its own separate Javadoc statement.

### Development Recommendations

### Developing your code in stages and knowing how to test your solutions will be critical to your success.

* 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.
* 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.



* You should write some of your own testing code. TagMain requires every method to be written in order to compile, and it never generates any of the exceptions you have to handle, so it is not exhaustive.

### Milestone 1

To demonstrate you understand what is expected for this assignment you will write the constructor and the printTagRing method. This will demonstrate you know how to traverse a linked list. Turn this into the Tag Manager Milestone 1 GradeIt link

Note: To test your code you should comment out parts of the "Main" program, or you can create stub methods as necessary.

### Milestone 2

For Milestone 2 complete the TagManager.java meeting all internal and external requirements and turn it into the GradeIt link provided on the course website.

### Grading

You will be graded on the program’s "external correctness" (whether the program compiles and produces exactly the expected output), its "internal correctness" (whether your source code follows the stylistic guidelines in this document) as well as its style and documentation.

#### Milestone 1:

 Milestone 1 (2 points)

#### Milestone 2:

9 points - external correctness

2 points – input

7 points - output  
6 points - internal design and efficiency  
3 points - style and documentation

### Optional Features

There are no optional features for this program

### Challenges

* There are no current challenges for this assignment.