LL – Assassin

**Program Details**

“Assassin” is a game often played on college campuses. Each person playing has a particular target that he/she is trying to “assassinate.” Generally, “assassinating” a person means finding them on campus in public and acting on them in some way (e.g. saying “You’re dead,” 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 assassinating; they don’t know who is trying to assassinate them, nor do they know whom the other people are trying to assassinate.

**Rules**

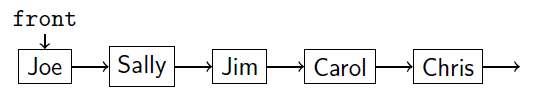
• You start out with a group of people who want to play the game

• A circular chain of assassination targets (called the “kill ring” in this program) is established.

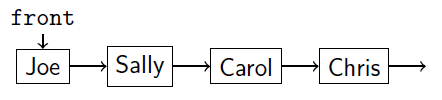
• When someone is assassinated, the links need to be changed to “skip” that person.

**Example**

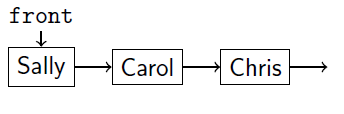
Let’s walk through an example with five people playing: Carol, Chris, Jim, Joe, Sally. We might decide Joe should stalk Sally, Sally should stalk Jim, Jim should stalk Carol, Carol should stalk Chris, and Chris should stalk Joe. In the actual linked list that implements this kill ring, Chris’s next reference 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 “kill ring”:



Then, suppose Sally assassinates Jim. Sally needs a new target, so we give her Jim’s target: Carol. The kill ring becomes:



If the first person in the kill ring is assassinated, the front of the list must adjust. If Chris kills Joe, the list becomes:



In this assignment, you will write a class AssassinManager that keeps track of who is stalking whom and the history of who killed whom. You will maintain two linked lists:

• a list of people currently alive (the “kill ring”) and

• a list of those who are dead (the “graveyard”).

As people are killed, you will move them from the kill ring to the graveyard by rearranging links between nodes. The game ends when only one node remains in the kill ring, representing the winner. A client program called AssassinMain has been written for you. It reads a file of names, shuffles the names, and constructs an object of your class AssassinManager. This main program then asks the user for the names of each victim to kill until there is just one player left alive (at which point the game is over and the last remaining player wins). AssassinMain calls methods of the AssassinManager class to carry out the tasks involved in administering the game.

**AssassinManager**

You must use the AssassinNode class which we have provided to you as a private inner class. AssassinManager should have two fields, a reference to the front node of the kill ring and a reference to the front node of the graveyard (null if empty).

*public* ***AssassinManager****(List<String> names)*

This constructor should initialize a new assassin manager over the given list of people. Note that you should not save the list parameter itself as a field, nor modify the list. Instead, you should build your own kill ring of list nodes that contains these names in the same order.

If the list is null or empty, you should throw an IllegalArgumentException. For example, if the given list contains ["John", "Sally", "Fred"], your initial kill 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. Do not change the list that is passed it.

*public String* ***killRing****()*

This method should return the names of the people in the kill ring, one per line, indented by two spaces, as “X is stalking Y ”. If the game is over, then instead print “X won the game!”.

For example, using the names in the example game above, the String is:

>> Joe is stalking Sally

>> Sally is stalking Jim

>> Jim is stalking Carol

>> Carol is stalking Chris

>> Chris is stalking Joe

*public String* ***graveyard****()*

This method should return the names of the people in the graveyard, one per line, with each line indented by two spaces, with output of the form “name was killed by name”. It should return the names in the opposite of the order in which they were killed (most recently killed first, then next more recently killed, and so on). It should produce no output if the graveyard is empty.

For example, using the names from above, if Jim is killed, then Chris, then Carol, the String is:

>> Carol was killed by Sally

>> Chris was killed by Carol

>> Jim was killed by Sally

*public boolean* ***killRingContains****(String name)*

This method should return true if the given name is in the current kill ring and false otherwise. It should ignore case in comparing names; so, “salLY” should match a node with a name of “Sally”.

*public boolean* ***graveyardContains****(String name)*

This method should return true if the given name is in the current graveyard and false otherwise. It should ignore case in comparing names; so, “CaRoL” should match a node with a name of “Carol”.

*public boolean* ***isGameOver****()*

This method should return true if the game is over (the kill ring has one person) and false otherwise.

*public String* ***winner****()*

This method should return the name of the winner of the game, or null if the game is not over.

*public void* ***kill****(String name)*

This method should record the assassination of the person with the given name, transferring the person from the kill ring to the front of the graveyard. This operation should not change the relative order of the kill ring (i.e. the links of who is killing whom should stay the same other than the person who is being killed). This method should ignore case in comparing names.

A node remembers who killed the person in its killer field, and you must set the value of this field. You should throw an IllegalStateException if the game is over, or throw an

IllegalArgumentException if the given name is not part of the kill ring. If both of these conditions are true, the IllegalStateException takes precedence.

The kill method is the hardest one, so write it last. Use the 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.

**Constraints**

• You may not construct any arrays, ArrayLists, LinkedLists, Stacks, Queues, or other data

structures; you must use list 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 AssassinNode objects in your constructor. As people are killed, you have to move their node from the kill ring to the graveyard by changing references, without creating any new node objects.

• Your constructor will create the initial kill 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 AssassinNode (like current from lecture) as you like. AssassinNode 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. AssassinMain 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.

**Sample Execution**

>> Welcome to Assassin Manager

>>

>> What name file do you want to use this time? names3.txt

>> Do you want the names shuffled? (y/n)? n

>>

>> Current kill ring:

>> Athos is stalking Porthos

>> Porthos is stalking Aramis

>> Aramis is stalking Athos

>> Current graveyard:

>>

>> next victim? Aramis

>>

>> Current kill ring:

>> Athos is stalking Porthos

>> Porthos is stalking Athos

>> Current graveyard:

>> Aramis was killed by Porthos

>>

>> next victim? Athos

>>

>> Game was won by Porthos

>> Final graveyard is as follows:

>> Athos was killed by Porthos

>> Aramis was killed by Porthos