Q: How do I get the nodes out of the `List` I am passed in my constructor?
Q: How do I put `AssassinNode` into a `LinkedList`?

A: You do not have to and should not be trying to do either of these things. The `List` of `String` objects you are passed is just a way of giving you the names in a specific order. You need to access those names (not any nodes) and convert them into a chain of `AssassinNode` objects that represent the state of the game (who is stalking whom). The structures you make will look like "lists-that-are-linked", that is, they are lists comprised of nodes (`AssassinNode`) that are connected via `next` links, but they are not `LinkedList` objects (the Java collection). The assignment write-up states : "*You may not construct any arrays, `ArrayLists`, stacks, queues, sets, maps, or other data structures to solve this problem.*"

Q: How does an `AssassinNode` work?

A: The `AssassinNode` is just a structure that keeps track of two pieces of data (`name` and `killer`) and a link to another node. As all of its fields are public, you have access to them as you are writing your `AssassinManager` code and should modify them as is necessary to implement the behavior specified by the write-up

Q: How many `AssassinNode` objects can I use? It doesn't seem like I have enough variables to manipulate the lists.

A: The number of objects you may actually instantiate is the same as the number of names passed in you constructor. This means that you must only call `new AssassinNode` once for each name you are given. It does not mean that you cannot create temporary variables to point to these nodes (such as you do with `current` when traversing a list). You may use as many temporary variables as is required to manipulate the lists.

Q: How do I kill the first/last person?

A: Part of the problem you have to solve is recognizing the different cases of the list. Dealing with certain nodes is going to prove more difficult than dealing with others, it is up to you to recognize those cases and implement behavior specific to them.

Q: Can I use a circular list instead of a null-terminating list?

A: You **must not** use a circular list for this exercise, that is, **the last node in your list must have a null link**, not a link to the first node in your list. It is not needed and would likely cause you many more problems.