1. Questions?

2. Iterators, objects that embody control.

   (a) The `java.util.Iterator` interface requires three methods: `hasNext`, `next`, and `remove`.

   (b) Iterators maintain internal state that references another object, or the current state of a search. For example, in a `SinglyLinkedList`, this may be a pointer to a `Node` in the list. Abstractly, let’s call this state `current`; typically `current` is a reference to `unexplored territory`.

   (c) The `hasNext()` method returns `true` if there remain untraversed items. For example, in a `SinglyLinkedList` if `current` points to a node.

   (d) The `next()` method explores and returns the next item in the traversal. For example, in a `SinglyLinkedList`, the `next()` method would capture the `value()` associated with the `current` node. It would then move `current` to the next node. The value is returned.

   (e) The `remove()` method may allow removal of a value; my structures typically don’t provide this method.

   (f) The `AbstractIterator` class also provides several other methods, including `get`, a non-advancing version of `next`.

3. The `java.util.Iterable` interface is used for any class that has an `iterator()` method. This method simply returns an `Iterator` over objects of the class.

Notes: