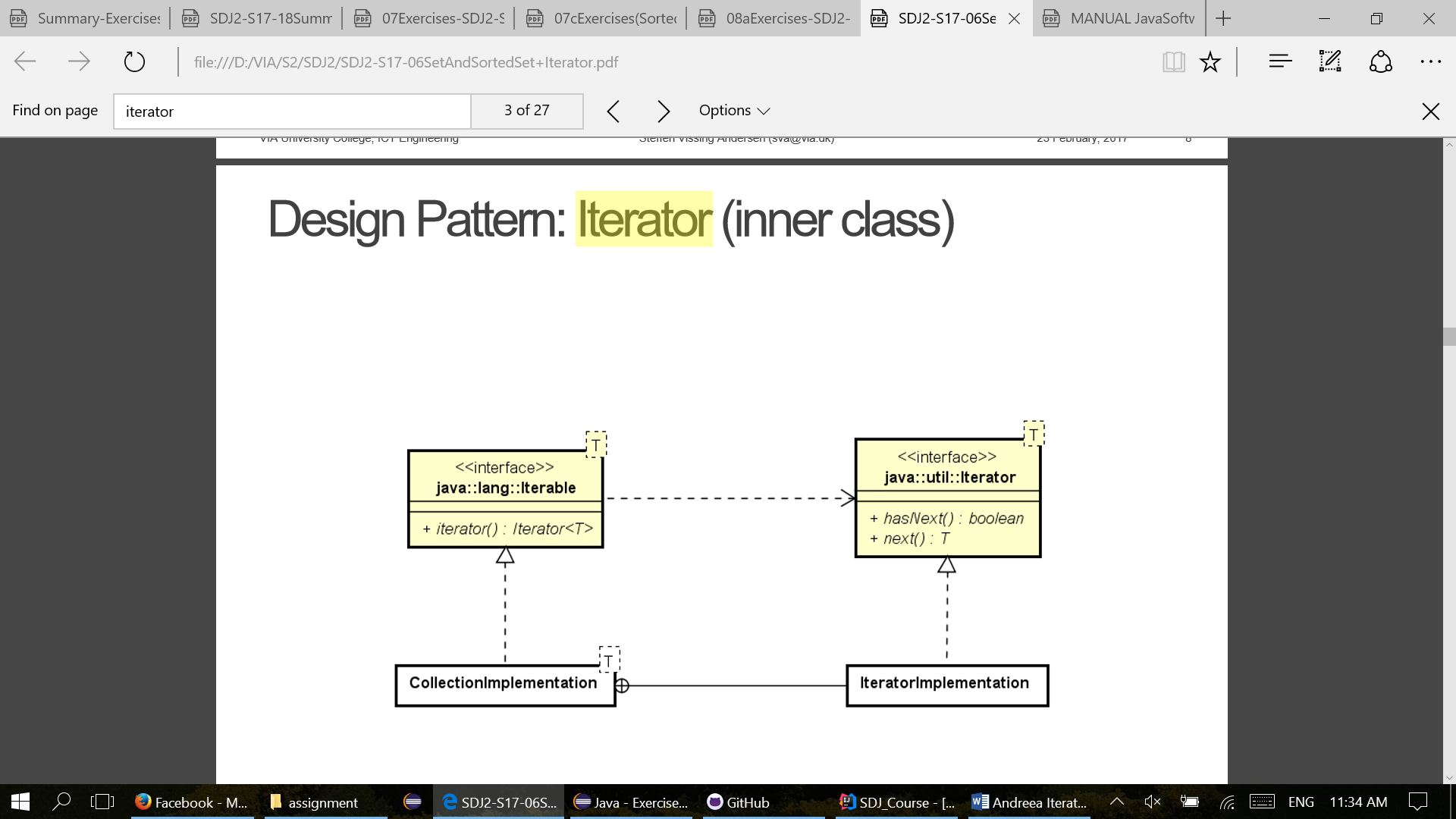
**Iterator**

**A general UML class diagram of an Iterator design pattern.**



**The overall purpose for the Iterator design pattern.**

An iterator is an object that provides the means to iterate over a collection. It provides methods that allow the user to acquire and use each element in a collection. Most collections provide one or more ways to iterate over their elements.

**Description of the diagram, purpose and relation.**

The Iterator interface is defined in the standard Java class library. The two primary abstract methods defined in the Iterator interface are:

* hasNext(): returns true if there are more elements in the iteration.
* Next(): returns the next element in the iteration.

There is no order in which an Iterator object delivers the elements from the collection. In the case of a list, there is a linear order to the elements, so the iterator would likely follow that order. In other cases, an iterator may follow a different order that makes sense for that collection.

**The use of an Iterator design pattern.**

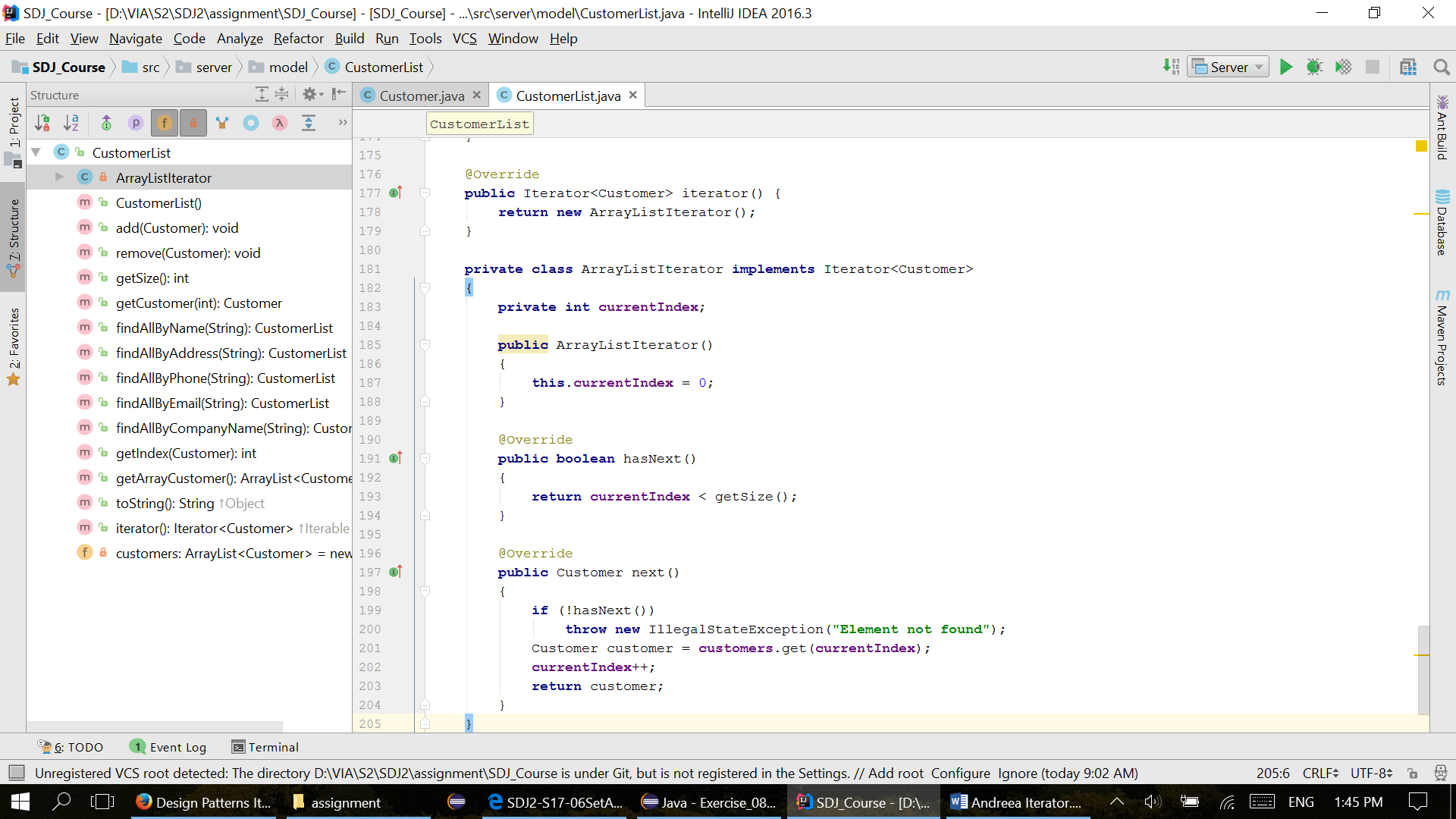
Implementing the Iterable interface makes the collection Iterable. The iterator method of the interface returns an object that implements the Iterator. The user can then interact with that object, using the hasNext and next methods, to access the elements in the list.

**Overall remarks to implementation.**

- Iterator method boolean hasNext() - returns true if the iteration has more elements. (In other words, returns true if next would return an element rather than throwing an exception.)

* Iterator method E next() - returns the next element in the iteration.
* Throws: – NoSuchElementException-iteration has no more elements.

**Code fractions directly related to the pattern.**



**UML class diagram of the Iterator design pattern we implemented.**

