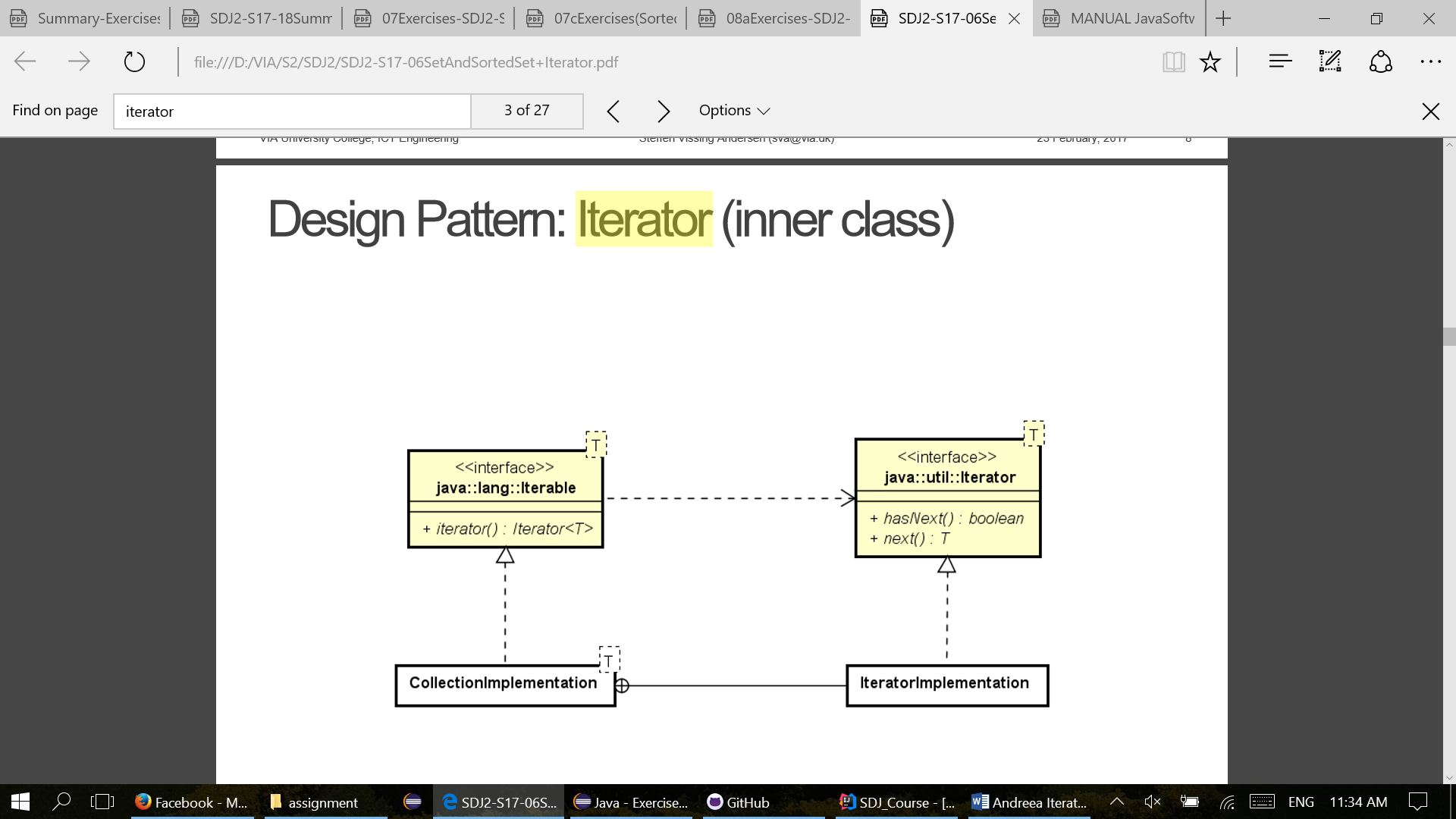
**Iterator design pattern**

**General UML diagram for iterator design pattern:**



**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 in turn. Most collections provide one or more ways to iterate over their elements.

**Description of general UML diagram:**

* The Iterator interface is defined in the Java standard class library. The two primary abstract methods defined in the Iterator interface are:
  + hasNext, which returns true if there are more elements in the iteration.
  + next, which returns the next element in the iteration.
* There is no assumption about the 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.

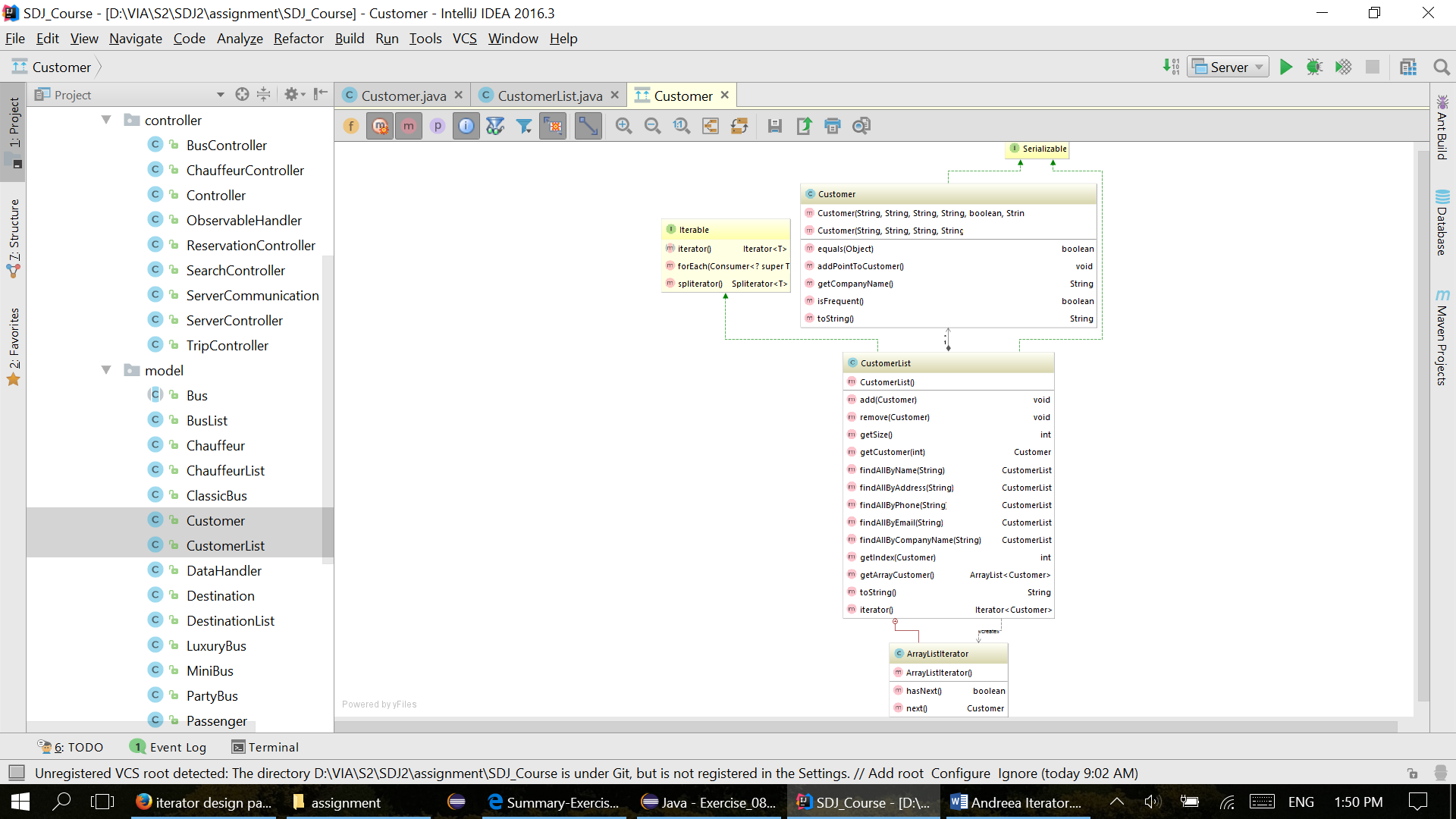
**Usage of iterator design pattern:**

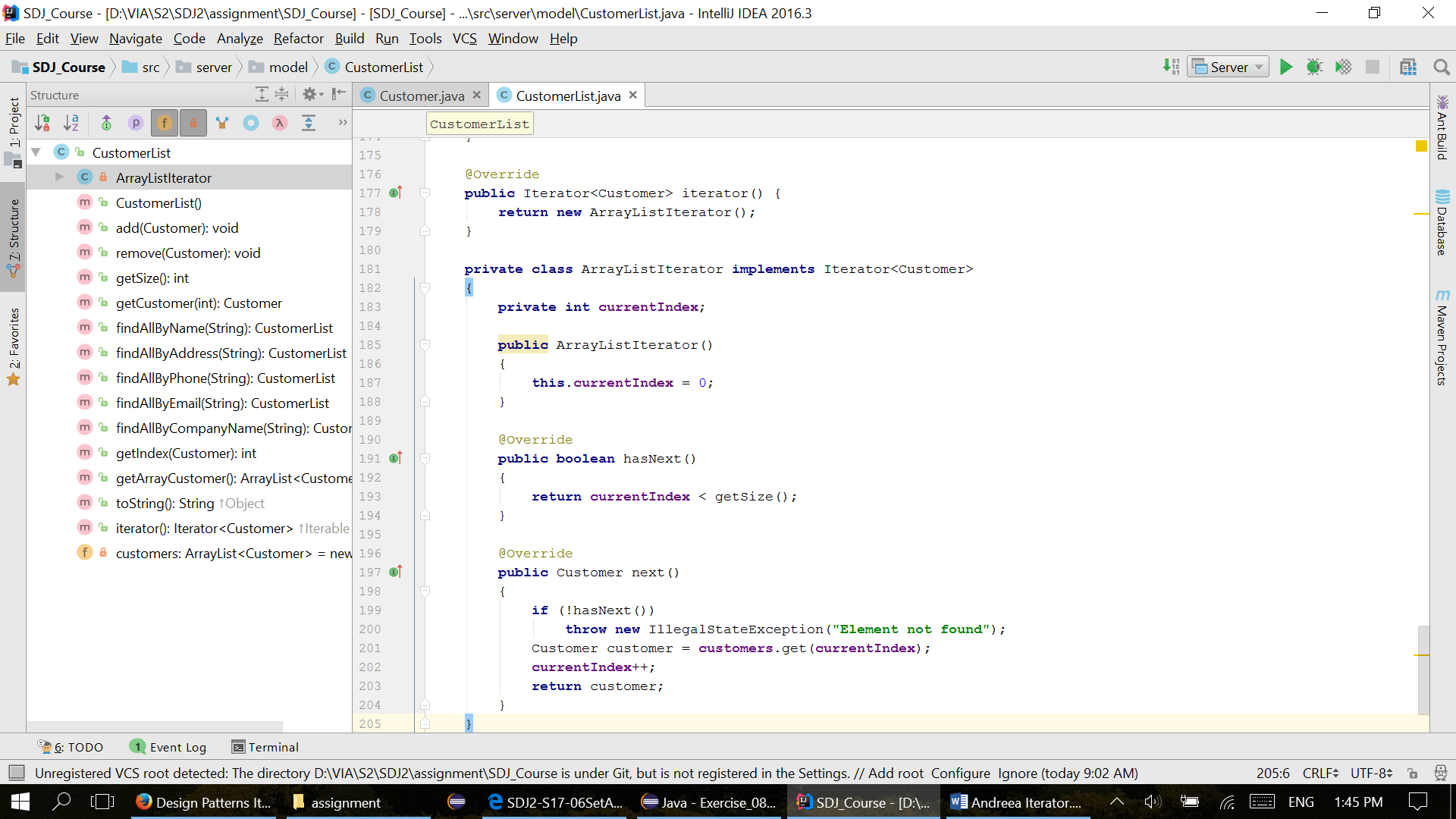
* Including an Iterator method in the collection, it makes the collection Iterable or in other words, it implements the Iterable interface. The iterator method returns an object that implements the Iterator. The user can then interact with that object, using the hasNext and next methods, to access the elements in the list.

**Remarks for iterator design pattern:**

* 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.

**Our implementation:**



**Code examples:**