Assignment 11

1. Replace sets with collections. equals() no longer satisfies its contract.
   * Explain why there is a problem.
   * **Collections can contain both Sets and Lists which have very different definitions of equals() method. Sets do not allow duplicates and have no order whereas Lists have order and can have duplicates. This causes the equals method to fail symmetric test. Equals will also fail reflexive and transitive tests as well because whatever collection is wrapped was calling its equals() method on the InstrumentedCollection and indicating that they were not equal because the wrapping class is a Collection, not an instance of the wrapped object. For example if the InstrumentedCollection was instantiated with a List, the equals() method will be dispatched to the List equals(), and it will find that the InstrumentedCollection is not a List and return false. (Help also provided by the Piazza Post by Kathleen Lazo)**
   * Demonstrate the problem with a suitable JUnit test.
     1. **Please see the symmetric, reflexive, and transitive tests in code.**
2. Now consider the example:
   * With lists.
     1. **There isn’t a problem. As stated in the List Javadoc under the equals method “Compares the specified object with this list for equality. Returns true if and only if the specified object is also a list, both lists have the same size, and all corresponding pairs of elements in the two lists are equal. (Two elements e1 and e2 are equal if (e1==null ? e2==null : e1.equals(e2)).) In other words, two lists are defined to be equal if they contain the same elements in the same order. This definition ensures that the equals method works properly across different implementations of the List interface.” The equals method is implemented in a way that it will work properly across different implementations which is unlike collection (since the equals method is just the Object.equals() reference test).**
   * With maps.
     1. **There isn’t a problem. As stated in the Map javadocs for equals “Compares the specified object with this map for equality. Returns true if the given object is also a map and the two maps represent the same mappings. More formally, two maps m1 and m2 represent the same mappings if m1.entrySet().equals(m2.entrySet()). This ensures that the equals method works properly across different implementations of the Map interface.” They implemented equals in such a way that it will work properly across different implementations which is unlike (since the equals method is just the Object.equals() reference test).**
     2. In either case, is there a problem with equals()? If so, give a JUnit test. If not, why not? If you're not sure, simply run the code.