Note that much has changed from hw-01.

* There are three packages:
  + shop.data holds the inventory
  + shop.command holds the command interface -- more here next week.
  + shop.main holds the integration test -- more here next week.
* There are interfaces for Video, Inventory and Record.
* The inventory object classes are all package-private.
* The static factory Data provides access to inventory objects.
* The inventory interface has changed substantially:
  + toCollection has been replaced with two iterator methods: one is an unsorted iterator, the other uses the strategy pattern to return a sorted iterator.
* The mutator methods of the inventory are all package-private.
* To modify the inventory from outside the package, one must invoke the run method, passing in a Command object. See the example in file:hw-02/source/shop/main/TEST1.java [source] [doc-public] [doc-private].
* RecordObj is a nested class, emphasizing its dependence on InventorySet.
* Since record objects are protected behind the public interface Record, it is no longer necessary to copy them -- they cannot be modified outside the shop.data package.
* Complete the incomplete classes (you may wish to paste in some of your hw-01 solution).
* Modify your unit tests so that they pass. (There is no longer any need to test Record).
* Write the integration test shop.main.TEST1.

Comment on integration tests [2/5]

* A nice way to generate an integration test (like shop.main.TEST1):
  + write a driver with some print statements, like this:
    - System.out.println(Data.inventory().get(v1));
  + which may output something like
    - Title1 (2000) : Director1 [10,0,0]
  + read the output to check it and make sure it's okay
  + then replace the System.output with an assert:
    - Assert.assertEquals("Title1 (2000) : Director1 [10,0,0]", Data.inventory().get(v1).toString());
* Now you can run the test over and over and it will make sure that the output is always what you got the time you checked it.

Comment on implementing iterators [3/5]

* For InventorySet.iterator(), it's easy just return

Collections.unmodifiableCollection(\_data.values()).iterator();

* For InventorySet.iterator(Comparator c), it is a bit more delicate.
* Here's one way:
  + Create a separate copy in a list, for example

List l = new ArrayList(\_data.values())

* + Sort the list using Collections.sort(List,Comparator)
  + Return an iterator over the sorted list
* How to test InventorySet.iterator(), where we don't know the order...
  + Create a local HashSet and an InventorySet.
  + Put some videos in the InventorySet.
  + Put corresponding Videos in the HashSet.
  + Use Video, not Record, since you cannot create records directly. You could also use strings such as "Title1 (2000) : Director1 [10,0,0]"
  + Now call InventorySet.iterator() and go through the elements.
  + For each record returned do the following:
    - Make sure an equivalent video is in the local HashSet
    - Remove the equivalent video from the HashSet
  + Once you're done with the iterator, make sure the local HashSet is empty.
* To test InventorySet.iterator(), where we do know the order ...
  + Create class that implements Comparator. When called, the. arguments will be Records.
  + For example, the following class will compare Records based on the number owned:
    - Class NumOwnedComparator implements java.util.Comparator {

public int compare (Object o1, Object o2) {

Record r1 = (Record)o1;

Record r2 = (Record)o2;

return r2.numOwned() - r1.numOwned();

}

}

* + Continue as before, but this time use a List to keep the videos in expected order.