1 Description

We will be designing, implementing and testing a bookstore application in the next several homework assignments. Each assignment will build on the previous. It will be to your advantage to complete each assignment and do so on-time.

2 Implementation Instructions

This assignment adds three things to the Bookstore Application: support for more than a single Merchandise, a new Magazine item, and a slightly re-organized class hierarchy.

In the previous assignment, the Bookstore held only a single Merchandise object. Now that we have some exposure to collections in Java, we are going to add support for a more complete inventory. The function and result of any method not mentioned below should remain the same as the previous assignment.

Change the following methods:

- void addMerchandise(Merchandise merch)
  - If the Merchandise being added does not already exist in the inventory, add it to the Inventory and update the count for that Merchandise to 1
  - If the Merchandise does exist (there is a Merchandise with the same SKU) then simply update the count for that Merchandise
- boolean buy(int SKU)
  - Note: the return type of this method has changed.
  - If a Merchandise with the proper SKU is available, decrement the count for that Merchandise and return true.
  - Otherwise, return false.
- boolean returnMerchandise(int SKU)
  - Note: the return type of this method has changed
  - If a Merchandise with the proper SKU exists in the Inventory, increment the count for that Merchandise and return true.
  - Otherwise, return false.
Also you will be adding a Magazine object. A Magazine is like a Book, but also has the following two interface methods:

- `Date getIssueDate()`
- `void setIssue(Date date)`

Since Book and Magazine are very similar, it makes sense to have them share a common base class. Also, the owners of the bookstore are thinking about expanding. They plan to sell snacks and coffee, and are also considering getting into the e-book business. The owners would like their inventory system to support all of these things.

To be more flexible we’ll add the following methods to Merchandise:

- `String getName()`
- `void setName(String name)`

Like we noticed before, a Book and Magazine have a lot in common. So, we are going to implement a ReadableMerchandise class. ReadableMerchandise should extend Merchandise and implement all the common interface components between Book and Magazine. Also, since title is more appropriate for ReadableMerchandise than just general Merchandise, we will move the getTitle and setTitle methods to ReadableMerchandise. The getTitle and setTitle methods are actually just convenience methods, and should call getName and setName respectively.

Also, you should prevent instantiation of both the Merchandise and ReadableMerchandise classes, since they are abstract entities.

3 Next Steps

Finally, as mentioned above, the owners are considering selling e-books. We will be implementing this functionality in the next assignment. Begin to think about how e-books should be implemented.

4 Suggestions & Submission Details

You may wish to implement a testing procedure to evaluate the correctness of your implementation. Your code will be tested using a test harness to ensure its correctness. In addition to submitting the .java files for all of the object implementations, please submit a brief write-up (pdf or plain-text files only) describing any difficulties you have encountered or any concerns you may have.