CISC 3130 **Ari Mermelstein**

Homework #4:

In this assignment you will be asked to implement a simulator for a hardware store. (This will contain a number of classes and a lot of moving parts, so please ask a lot of questions.) This assignment will require you to use the ArrayList or LinkedList (your choice), in addition to the Queue interface and PriorityQueue You will also need to use your object oriented programming skills to create clean, maintainable code.

(**N.B.** If your code is not documented, you will not receive credit. This means that if you write any line of code that isn't completely trivial, you must explain it to me.
You may not use any code or any function that I have not discussed in class. Any fancy algorithm you use must be written by you or discussed with me beforehand. I have been getting some interesting code for which ascertaining its correctness is almost impossible.)

You will create the following classes. I will provide prototypes below:

An Item class that models the name and price of something that is sold in a hardware store. For example, a hammer that costs $10.50.

This class must support the following operations:

Item(**String** name, **int** price);
A constructor that takes in the name and price as input.

String getName();
This returns the name of the item
int getPrice();
This returns the price of the item.

String toString();

Creates a String representation for an Item.

A ShoppingCart class that contains a List of Items.

The ShoppingCart class must implement the following methods.

ShoppingCart();
This creates an empty shopping cart.

void addItem(Item item);
This adds an item into the shopping cart.

int grandTotal();
This returns the total price of all of the items in the shopping cart in cents.

int numItems();
This returns the number of items in the shopping cart.

String toString(); This creates a String representation for a

ShoppingCart.

A Shopper class, which needs to implement Comparable<Shopper> which contains a name and a ShoppingCart. It must implement the following methods.

Shopper(**String** firstName, **String** lastName);
This creates a shopper with this name and with an empty ShoppingCart.

void addItemToCart(Item item, **int** numItems);
This adds *numItems* items of the same type to the shopping cart. If no second parameter is specified, only one item should be added.

int amountOwed();
This is the amount of money in cents that Shopper needs to pay the cashier in the hardware store. It is the total cost of the items in the cart plus an additional 8.875% tax. The price should be rounded to the nearest cent. (reminder: the ceil function in the Math class, rounds up a double to the nearest integer).

String toString(); Prints the representation for a shopper.

int compareTo(Shopper shopper); This will return a negative number if "this" has a bigger total to pay than "shopper," 0 if they owe the same amount, and a positive number if "shopper" has a bigger total than "this." This is how the priority queue will know which shopper to take next.

And finally, a HardwareStore class, that models the actual store. Each store has a List of Items that is its, an amount of money taken in by the store, a PriorityQueue and a List of Queues. (i.e. A bunch of checkout lines).

It must have the following methods:

HardwareStore(**int** numRegisters);
Creates a HardwareStore with the specified number of registers. (i.e. the number of queues).

void addShopperToLine(Shopper shopper);
Adds a shopper to the line that currently has the least number of people waiting on it.

void processShopper();
Processes the shopper at the front of the longest line.
void checkoutAllShoppers();
Processes all Shoppers still on line for all of the lines.

double totalRevenue();
Returns the amount of money the HardwareStore has taken in so far. (At the end of the program, it will represent the total revenue of the store.)

Your main program will read in 2 files: an inventory file and an event file.

The inventory file will have a number of lines of the form:

itemName itemCost

The event file will have a number of lines of either the form

P (which means process) followed by no additional information.
Or

S *n item1 numberOfPiecesOfItem1 item2 numberOfPiecesOfItem2 … itemN numberOfPiecesOfItemN*

When you see a P, you process the shopper in the front of the longest line.

When you see an S, you create a new shopper out of these information, add the shopper that the priority queue says should go next into line, and then add this new shopper to the priority queue. When the file is over, put the remaining Shoppers from the priority queue into the lines, and then process the rest of the shoppers.

You must write a main program that will read in the inventory file and store that information in the HardwareStore's inventory vector. Then, it will read in the event file and process the shoppers in it.

You must submit the following files:
Item.java
ShoppingCart.java
Shopper.java
HardwareStore.java
Main.java (a program that runs the simulation).