

# Computer Science 3310

## Program 1

Your assignment is to write, run, and test a program to simulate the operation of a grocery store checkout system. The program should operate as follows:

1. Read from an input data file the inventory information. The data file should be named **inventory.dat**. There is one input line per product, and these lines have the following format:

<product number> <description> <price> <tax>

where <product number> is a five-digit positive integer (you may assume the leftmost digit is not a zero), <description> is a string of at most 12 characters with no embedded blanks, <price> is a real number, and <tax> is a character ('F' if the product is a food item; 'N' if it is a non-food item). The tax rate is 7% for non-food items and 2% for food items. You may assume the inventory has no more than 100 items (a small store!).

2. Read from standard input the customer purchases and output the cash register receipt for the customer to standard output. Each input line should have the following format.

<product number> <times>

where <product number> is as described above, and <times> is an integer in the range 1 to 100 indicating the quantity purchased. A zero input for <product number> and <times> indicates the end of the customer's purchases. There may be multiple customers in one file. Use the end-of-file marker to detect the end of the data. For output the receipt should be nicely formatted with the product description (not the product number), number of items, item price, and total price for each product printed on a single line. At the end of each customer's purchases, the subtotal for all items, amount of tax, and total bill should be printed, clearly labeled. Produce this output as you are reading the purchases. A sample is shown at the end of the specification.

3. Error Checking: The following input errors are possible and should be handled by your program.
  - <product number> not in the inventory file. Write an error message on the receipt, ignore that product number, and continue with the next item.
  - Duplicate <product number> in the inventory file. Write an error message to the screen and skip the second entry. Note that the duplicate may not come immediately after the original.
  - <times> not in the specified range. Write an error message to the receipt, ignore that line, and continue with the next item.

## PROGRAM DETAILS

- **Program Structure:**

Your source code should be developed in two files: *GroceryItem.java* and *GrocerySim.java*. *GroceryItem.java* will contain the class definition for a grocery item according to the requirements specified below. *GrocerySim.java* will be the class definition that includes the actual main program for running the simulation.
- **Data Structure:**

Each inventory item should be an object of the *GroceryItem* class. Define a separate instance variable of the appropriate type for the four items of information about each inventory item. Instance variables should be maintained as *private* data. Required methods for the class are the following.

  - ❖ An accessor method for each instance variable.
  - ❖ A *readItem* method that will take as an input parameter a *Scanner* object that represents the input file being read. This method should then read in all the information for one inventory item and leave the position of the *Scanner* object at the beginning of the next input line.
  - ❖ A static method *itemSearch* that takes three input parameters: (1) an array of *GroceryItem* objects that represent the entire inventory; (2) an integer specifying how many elements of the array from (1) are actually being used; and (3) an integer representing a product item. The method should search the array of items looking for the

item whose product item is given by the third parameter. The method should return the index within the array where the information for the item is stored. If it cannot find the item, the method should return  $-1$ .

- ❖ A static method *printInventory* that can print the information on an array of inventory items, one per line, on standard output. The parameters required should be the same as the first two of *itemSearch*.
- File I/O:
  - Sections 1.6 through 1.8 provide a nice description of what is needed to handle File I/O within Java. Page 47 describes how to format output nicely. A couple other technical items not necessarily mentioned are given below.
  - ❖ You will need to import both `java.io.*` and `java.util.*` in *GrocerySim.java*.
  - ❖ The *Scanner* class has a *hasNext()* method that returns false when end-of-file is reached. This can be used to control your loops for reading the inventory and later the purchases.
  - ❖ You can use I/O redirection to specify the name of the purchases file. An example command line that would enable this is `java GrocerySim < purchases.dat`

Execution of this command causes Linux to associate the file *purchases.dat* with standard input (System.in).

## PROGRAM DEADLINES.

February 3, 5:00 P.M.

Deadline for submitting completed program, including all documentation. Submission should conform to the requirements discussed in the **Programming Standards** handout. The program ID should be *1*.

## PROGRAM GRADING

This program is worth **100** points. Your grade on this program will be weighted as follows:

Test Plan	5%
Correctness	75%
Coding Style	10%
Documentation	10%

## SAMPLE INTERACTION

### Sample Inventory Input File

11012	gallon-milk	1.99	F
11014	butter	2.59	F
11110	pie-shells	0.99	F
20115	laundry-soap	3.60	N
30005	homestyle-br	0.99	F

### Sample Customer Purchase Input File

20115	1
40012	3
11110	2
0	0
30005	2
11012	1
0	0

### Sample Output File

#### Customer 1

laundry-soap	1 @ 3.60	3.60 N
*** item 40012 not in inventory		***
pie-shells	2 @ 0.99	1.98 F

Subtotal 5.58

Tax 0.29

Total 5.87

#### Customer 2

homestyle-br	2 @ 0.99	1.98 F
gallon-milk	1 @ 1.99	1.99 F

Subtotal 3.97

Tax 0.08

Total 4.05