# ᐈ Shopping trolley stock pictures, Royalty Free shopping cart full images | download on Depositphotos®CSS 143 Final Assignment: Grocery Manager

Instructor: Lesley Kalmin

Updated 11/21

You will write a system to manage a grocery store. The initial store inventory is read from a file. Orders are read from file. The system stores the inventory, processes the orders, tracks which items are out of stock, and sorts the inventory list using two different algorithms.

Classes:

* GroceryManager
* GroceryOrder
* GroceryItem
* Dairy – subclass of GroceryItem
* Produce – subclass of GroceryItem
* Meat – subclass of GroceryItem
* GroceryException
* GroceryDriver – provided to you. You do not need to modify it.

The driver runs through the following scenario:

1. Stock the store by reading inventory from a file
2. Print inventory
3. Read the orders
4. Process orders
5. Sort by name, print inventory
6. Sort by price, print inventory
7. Print restocking list

Concepts used:

1. Inheritance
2. Generics
3. Exception Handling
4. HashSet
5. Insertion Sort
6. Bubble Sort

## Classes in Detail

**GroceryItem** – an abstract class that implements Comparable. GroceryItems are things like milk, lettuce, and bread. Each has a price. If they are stocked in the store, they also have a quantity.

Public methods:

* **GroceryItem** (String name, int quantity, double price);
* **GroceryItem();**
* void setName(String name);
* String getName();
* void setPrice(double price);
* double getPrice();
* void setQuantity(int n);
* int getQuantity();
* String toString();
* int compareTo(Object o); // Use name as criterion

**Dairy, Produce and Meat**

Each is a subclass of **GroceryItem**. GroceryItems have more specific information associated with them, depending upon their type (subclass). Note that the constructor actually takes an input line that has been read from a file. The line will have data specific to its type, which is why it is sent to the subclass. Remember that the constructor which read the line can parse it and then call the other constructor in the subclass.

Below are the public interfaces

Dairy:

* + - Dairy(String name, , int quantity, double price, int refrigerationTemperature); // use super
    - Dairy(String inputLine); // takes file input line, parses and sets data
    - void setRefrigerationTemperature(int temp);
    - int getRefrigerationTemperature();

Produce

* + - Produce(String name, , int quantity, double price, boolean isOrganic); // use super
    - Produce (String inputLine); // takes file input line, parses and sets data
    - void setIsOrganic(boolean organic);
    - boolean getIsOrganic();

Meat

* + - Meat(String name, int quantity, double price, boolean isGround); // use super
    - Meat(String inputLine); // takes file input line, parses and sets data
    - void setIsGround(boolean ground);
    - boolean getIsGround();

GroceryOrder – extends ArrayList<T>, limiting T to type GroceryItem

No further methods. Use Java ArrayList

GroceryManager – pulls the whole thing together

Contains two private collections:

* + - ArrayList< GroceryItem > inventory – the store’s current inventory
    - HashSet<String> restockingList – names of items that need to be reordered/restocked. Entries are unique (HashList does this for you).

Public methods:

* Void loadInventory(String filename) // reads input and populates inventory
  + Calls appropriate constructor with input line.
  + The input file starts with 3 integers, representing the number of Dairy, Produce, and Meat items to follow. You may assume the data is properly formatted.
* void processOrder(GroceryOrder<GroceryItem> order) // Subtracts the items and quantities in the order from the inventory. If the order has more than is in stock, fill it with whatever quantity is available. Be careful not to go below 0. If inventory goes to 0, add to restocking list. If the quantity ordered is greater than the quantity in inventory, throw a GroceryException with message “out of xxxx”, but keep on going. If an item in the order is not in inventory, also throw a GroceryException but keep going.
* GroceryItem findItemByName(String name); // Implement any way you want
  + Return null if not found
* void sortInventoryByName(); // Use BubbleSort, and use compareTo()
* void sortInventoryByPrice() // Use InsertionSort. Will compare without compareTo()
* void displayRestockingList() // print the list of items that need to be restocked.
* void displayInventory() // list inventory with all their class-specific data. Line up columns for readability. Use overloads of toString() from each class. Example:

Name: yogurt Quantity: 0 Price: $7.46 temperature: 34

Name: apples Quantity: 48 Price: $0.82 organic: true

Name: chicken Quantity: 9 Price: $5.29 isGround: false

See separate file: Grocery Manager Sample Output.docx for longer example.

## Data Files

groceryInventory.txt

* The input file starts with 3 integers, representing the number of Dairy, Produce, and Meat items to follow.
* Each line has the
  + type, name, quantity, price and one type-specific datum
* You may assume the data is properly formatted.

groceryOrders.txt

* Each order starts with the word Order on a line
* Other lines have the type, name, and quantity of an item in the order
  + You may assume the data is properly formatted.