
# Milestone 3

CST-135

Gary Davis
Nathaniel Kumar
Jesse Gardner
Alondra Dominguez Ramirez

# Summary

Update the Snack and Drink classes

- Implement a Comparable interface.

Comparison

- Alphabetical Order by name (ignoring case)
- Items with the same name will be ordered by price in ascending order

Test code in Main() method

Create ReadMe file

# Drink Class Output

## Random Name Order

Randomly generated drink vending machine:

| Product Name | Price | Quantity | Type | Location |
|---|---|---|---|---|
| Tea | 2.75 | 10 | Drink | A1 |
| Ice Coffee | 2.0 | 10 | Drink | A2 |
| Coca-Cola | 1.5 | 10 | Drink | A3 |
| Ice Coffee | 2.0 | 10 | Drink | A4 |
| Crush | 1.5 | 10 | Drink | B1 |
| Dr Pepper | 1.3 | 10 | Drink | B2 |
| Ice Coffee | 1.5 | 10 | Drink | B3 |
| Mtn Dew | 1.5 | 10 | Drink | B4 |
| Sprite | 1.75 | 10 | Drink | C1 |
| Coca-Cola | 1.0 | 10 | Drink | C2 |
| Coca-Cola | 1.0 | 10 | Drink | C3 |
| Tea | 2.75 | 10 | Drink | C4 |
| Pepsi | 1.5 | 10 | Drink | D1 |
| Crush | 1.5 | 10 | Drink | D2 |
| Dr Pepper | 1.5 | 10 | Drink | D3 |
| Sprite | 1.75 | 10 | Drink | D4 |

## Alphabetical Order

Same Name so ascending order by price

Alphabetical ordering of drink vending machine

| Product Name | Price | Quantity | Type | Location |
|---|---|---|---|---|
| Coca-Cola | 1.0 | 10 | Drink | C2 |
| Coca-Cola | 1.0 | 10 | Drink | C3 |
| Coca-Cola | 1.5 | 10 | Drink | A3 |
| Crush | 1.5 | 10 | Drink | B1 |
| Crush | 1.5 | 10 | Drink | D2 |
| Dr Pepper | 1.3 | 10 | Drink | B2 |
| Dr Pepper | 1.5 | 10 | Drink | D3 |
| Ice Coffee | 1.5 | 10 | Drink | B3 |
| Ice Coffee | 2.0 | 10 | Drink | A2 |
| Ice Coffee | 2.0 | 10 | Drink | A4 |
| Mtn Dew | 1.5 | 10 | Drink | B4 |
| Pepsi | 1.5 | 10 | Drink | D1 |
| Sprite | 1.75 | 10 | Drink | C1 |
| Sprite | 1.75 | 10 | Drink | D4 |
| Tea | 2.75 | 10 | Drink | A1 |
| Tea | 2.75 | 10 | Drink | C4 |

# Snack Class Output

## Random Name Order

Randomly generated snack vending machine:

| Product Name | Price | Quantity | Type | Location |
|---|---|---|---|---|
| Hersheys | 1.0 | 15 | Candy | A1 |
| Ruffles | 1.5 | 10 | Chips | A2 |
| Pringles | 1.5 | 10 | Chips | A3 |
| Snickers | 1.0 | 15 | Candy | A4 |
| Trident | 0.5 | 20 | Gum | B1 |
| Cheetos | 1.5 | 10 | Chips | B2 |
| Snickers | 1.0 | 15 | Candy | B3 |
| Kit-Kat | 1.0 | 15 | Candy | B4 |
| M&Ms | 1.0 | 15 | Candy | C1 |
| Tostitos | 1.5 | 10 | Chips | C2 |
| Trident | 0.5 | 20 | Gum | C3 |
| Hersheys | 1.0 | 15 | Candy | C4 |
| Snickers | 1.0 | 15 | Candy | D1 |
| Cheetos | 1.5 | 10 | Chips | D2 |
| Cheetos | 1.5 | 10 | Chips | D3 |
| Juicy Fruit | 0.5 | 20 | Gum | D4 |

## Alphabetical Order

Alphabetical ordering of snack vending machine

| Product Name | Price | Quantity | Type | Location |
|---|---|---|---|---|
| Cheetos | 1.5 | 10 | Chips | B2 |
| Cheetos | 1.5 | 10 | Chips | D2 |
| Cheetos | 1.5 | 10 | Chips | D3 |
| Hersheys | 1.0 | 15 | Candy | A1 |
| Hersheys | 1.0 | 15 | Candy | C4 |
| Juicy Fruit | 0.5 | 20 | Gum | D4 |
| Kit-Kat | 1.0 | 15 | Candy | B4 |
| M&Ms | 1.0 | 15 | Candy | C1 |
| Pringles | 1.5 | 10 | Chips | A3 |
| Ruffles | 1.5 | 10 | Chips | A2 |
| Snickers | 1.0 | 15 | Candy | A4 |
| Snickers | 1.0 | 15 | Candy | B3 |
| Snickers | 1.0 | 15 | Candy | D1 |
| Tostitos | 1.5 | 10 | Chips | C2 |
| Trident | 0.5 | 20 | Gum | B1 |
| Trident | 0.5 | 20 | Gum | C3 |

# ReadMe

- This application was developed for a vending machine interface. Multiple classes were developed based on the product type (Candy, Chips, Gum and Drink). These classes will have getters, setters and toString methods. The dispenser class will auto populate an object array to store and track the created objects. The dispenser class will also have a displayProducts() method that will report all the names, prices, quantities, types (depends on the class that creates it) and stored locations.

- Now we have an interface that will compare the products. The program will first compare by the name of the product, and if the name are the same, then the program will compare the products by price. This helps for the programmers and the manager to ensure that the products are not duplicated or that the price is set up properly.