TRINITY COLLEGE

DEPARTMENT OF COMPUTER SCIENCE

CPSC 215: Data Structures and Algorithms

Instructor: Dr. Chandranil Chakraborttii

Spring 2025

Bonus Assignment

Academic Honesty Policy

In working on programming assignments, you may discuss broad issues of interpretation and understanding and general approaches to a solution. However, conversion to a specific solution or to program code must be your own work. Programming assignments are expected to be the work of the individual student, designed and coded by him or her alone. Violations are easy to identify and will be dealt with promptly according to the Academic Integrity and Intellectual Dishonesty outlined in Student Handbook. Copying another person's programs or encouraging or assisting another person to commit plagiarism is cheating. In particular, the following activities are strictly prohibited:

- Giving and receiving help in the actual development of code or writing of an assignment
- Looking at another person's code
- Showing your code to another person
- Sharing a copy of all or part of your code regardless of whether that copy is on paper or in a computer file
- Turning in the work of any other person(s) (former students, friends, textbook authors, people on the Internet, etc.) and representing it as your own work
- Fabricating compilation or execution results
- Use of AI ChatBots for any help regarding this assignment is strictly prohibited.

The penalty for cheating is a failing grade (F) for the course, and the student is asked to appear at an Academic Dishonesty Hearing. In addition, the College also places a record of the incident in the student's permanent record. It is your responsibility to protect your work from unauthorized access. Do not discard copies of your programs in public places. Do not leave computers unattended and copies of output lying around.

Submission Guidelines:

- The output of the class **ProductInventory** (in a text file, or screenshots in a doc)
- If completed, the output from the final version of the class **ProductInventory** showing the extra credit work.
- The final version of **ProductInventory.java** (Need to create).
- The final version of **Product.java** (Need to create).
- Consolidate and upload the submission to the Moodle course website as **single Zip file.**

Part 1: Assignment Overview

In this bonus assignment, you will design and implement a simple product inventory system. Your program will allow to manage **products** by providing the following operations:

- **insert ():** Add a new product. Products should be identified by unique product numbers(integers). There should not be two products with the same product number.
- **remove ():** Remove a product by using its product number.
- **display ():** List all products by their product number. (You should maintain the list of products ordered by their product numbers.)
- **display Available():** List all available (in stock) products along with the total number of products and the total number of products available in stock.
- **find ():** Look up a product by its product number.
- **topSearched():** List top 5 recently searched products (not product numbers).

Please pay attention to how your program works. The user should be notified about error conditions such as trying to perform a non-existent operation, trying to remove a product that does not exist, etc. Keep it neat and user-friendly.

Extra Credit (20 points)

Once you implement the functionality described above, you can attempt the extra credit part. To earn extra credit, you should add to your program a simple text-based menu that will allow a user to continuously perform the above operations (by specifying the operation they want to perform, for example, 1 to add a product, 2 to remove a product, etc.) until the user specifically requests to end the program (for example, by entering 'quit' or 'q'). You should carefully design the menu such that your program neatly displays all available options, the user is always prompted to provide the necessary information to perform different operations, and the screen is cleared appropriately after each operation to ensure a user-friendly design.

Part 2: Programming Requirements

You should design and implement a class called **Product** to represent an individual product. You should:

- Design a Product class with instance variables, constructors, methods, and getters/setters. Define default and parametrized constructors. Your constructors should make use of this keyword.
- Include appropriate instance variables to keep track of the product's name, number, category, and availability as well as other instance variables (if needed) to support operations described in Part 1.
- Include appropriate methods, setters, and getters to support operations described in Part 1.
- Implement the Comparable interface so products can be compared using their product number. You need to include a method int compareTo(Object o). Note that it accepts an Object o, not a Product, so you will have to cast o to Product first.
- Provide toString() method to properly display a product by listing its product number, name, and category.
- Declare all helper methods as private.
- Include Javadoc and in-line comments.

You should design and implement a class called **ProductInventory** to represent a collection of individual products. You should:

- Use the **PositionalList** ADT.
- Include appropriate instance variables to support operations described in Part 1.
- Include a variable to store the name of the company using your inventory system.
- Define default and parametrized constructors.
- Include appropriate methods to provide operations described in Part 1.
- Include a main() method to test your program. This method should include your text-based menu that accepts input from the user.
- Handle error conditions, both for your program and the text-based menu.
- Make use of an Iterator to traverse your collection of products.
- Provide toString() method to display all products.
- Declare all helper methods as private.
- Include Javadoc and in-line comments.

Positional List is another abstract data structure (ADT) which allows us to add, remove, insert, and sort elements within the list. The list is a generalization of both stacks and queues. But we are very restricted as to where these insertions and deletions may occur. Generally, two positions are recognized: front and rear. If a list has at least one node, the front is the position occupied by this node; if a list is empty, the front coincides with the rear. If our program requires frequent addition or deletion at random locations, and if the program requires that position information isn't affected by add/delete operations, a positional list is a better choice than an array or linked list. The following link contains additional information about Positional Lists and its implementation in Java:
https://cs226sp22.github.io/notes/14-list/index.html

You are provided with the implementations of the Stack, Queue, Deque, and PositionalList ADTs. You do not need to use all of these ADTs. Your program must be based on the **PositionalList ADT** but it can also use auxiliary ADTs. Your program cannot directly make use of any arrays or linked lists. For example, it is acceptable to use a linked-list implementation of the Stack ADT but not a linked list instead of a Stack. All files for the above ADTs are attached to the Moodle assignment. Note that the Input files are archived into a zip file.

Input Files:

The archive consists of the following files. You should not make any changes to these files.

- Deque.java Interface for a double-ended queue
- DoublyLinkedList.java A basic doubly linked list implementation
- LinkedPositionalList.java Implementation of a positional list stored as a doubly linked list.
- LinkedQueue.java FIFO queue implementation as an adaptation of a SinglyLinkedList
- Stack.java Interface for a stack
- LinkedStack.java Stack implementation as an adaptation of a SinglyLinkedList
- Position.java Interface representing position of a single element within container)
- PositionalList.java Interface for positional lists
- Queue.java Interface for a double-ended queue
- SinglyLinkedList.java A basic singly linked list implementation.

You will need to create the following classes: **Product.java** and **ProductInventory.java**

Part 3: Testing

You should design and implement a series of tests to show that the operations described in Part 1 work correctly. To make testing easier, pick some specific products you will use. Do not use “product1”, “product2”, etc. as it will make your output look confusing and difficult to read. Remember, your goal in designing those tests is to show that your program works correctly. It is your responsibility to ensure that your output does so. You will be graded on how effectively your tests demonstrate that your assignment meets all requirements outlined in this handout. At the minimum, you should:

- Add several products and then display all products.
- Remove a product and then display all product.
- List all available products.
- Look up several products and display the top 5 searched products.
- Demonstrate how your program handles error conditions.

Extra Credit: As above, you should design and implement a series of tests to show that your text-based menu works well. Your goal is to demonstrate how your menu works, not that the main operations work properly. You should test both parts of your program separately.

Part 4: Documentation

Source code documentation: Document the code in **Product.java** and **ProductInventory.java** using Javadoc tags as before. Specifically, make sure to add @author, @version, @param, @return and @throws tags as appropriate. Additionally, analyze and add a line to the header of each method stating its time complexity. For example:

*** Time Complexity: O(n)**

Add in-line comments to explain how your code works. If you are unable to complete the assignment or your program does not work properly, make sure to document every part of your code that causes issues.

External documentation: Document your code using the provided cover page. You should include the interface (all public methods that define behaviors) for Product and **ProductInventory**. Do not create an actual Java interface, simply list all behaviors for each class as we did in class. You should document all arbitrary design and implementation choices that you made that are relevant to grading your assignment. As before, you should document any issues with your code.

Part 5: Grading Criteria

This assignment gives you a lot of freedom to decide how to best design and implement your program. To receive full credit, it is not sufficient that your program works, however. You should take your time to design and implement it properly. Therefore, your grade will be based on the following items:

- **Correctness:** your program must properly perform the operations described in Part 1.
- **Design:** your program must be well designed, that is, it should properly utilize and implement object-oriented principles.
- Implementation: your program must adhere to the implementation requirements described in Part 2.
- **Testing:** your main() method should properly demonstrate the correctness of your program.
- **Documentation:** your code should be properly documented using Javadoc, in-line comments and the cover page. Each method must have its complexity stated.

Extra Credit:

You should only attempt the extra credit part once you are done with the mandatory parts of the assignment. The extra credit part will not be graded otherwise. Your text-based menu will be graded based on:

- **Correctness:** the menu allows user to properly perform the operations described in Part 1.
- **Usability:** your program should be easy to use from the user's perspective.

Sample Output:

```
.....,
TESTING ALL METHODS-

Displaying the names and product numbers of products in inventory-
The inventory is currently empty.

Displaying available products-
The inventory is currently empty.

Now adding Cheese to inventory...

Now adding Eggs to inventory...

Now adding Chicken to inventory...

Now adding Bread to inventory...

Now adding Dr Pepper to inventory...

Now adding Coffee to inventory...

That item number is already in use. Please try a different number.

Now adding Marlboro Reds to inventory...

Displaying the names and product numbers of products in inventory-
Cheese, Product Number 1
Chicken, Product Number 99
Dr Pepper, Product Number 297
Marlboro Reds, Product Number 368
Eggs, Product Number 498
Bread, Product Number 3213

Removing item with number 498...
Item 'Eggs' removed!

Removing item with number 2...
No item with number 2 found in inventory.

Displaying the names and product numbers of products in inventory-
Cheese, Product Number 1
Chicken, Product Number 99
Dr Pepper, Product Number 297
Marlboro Reds, Product Number 368
Bread, Product Number 3213

Displaying available products-
Cheese, 90 products in stock.
Chicken, 38 products in stock.
Marlboro Reds, 100 products in stock.
Bread, 819 products in stock.

The total number of available products is: 4
The inventory currently has 1047 items.

Looking for item with number 1...
```

```
Product Name: Cheese
Product Number: 1
Product Category: Dairy
Product Avaiability: 90

Looking for item with number 368...

Product Name: Marlboro Reds
Product Number: 368
Product Category: Cigarettes
Product Avaiability: 100

Looking for item with number 498...
No item with number 498 found in inventory.

Looking for item with number 297...

Product Name: Dr Pepper
Product Number: 297
Product Category: Soda
Product Avaiability: 0

Looking for item with number 99...

Product Name: Chicken
Product Number: 99
Product Category: Protein
Product Avaiability: 38

The top 4 recently searched products were-
1. Chicken
2. Dr Pepper
3. Marlboro Reds
4. Cheese

The top 4 recently searched products were-
1. Chicken
2. Dr Pepper
3. Marlboro Reds
4. Cheese

Looking for item with number 3213...

Product Name: Bread
Product Number: 3213
Product Category: Staples
Product Avaiability: 819

The top 5 recently searched products were-
1. Bread
2. Chicken
3. Dr Pepper
4. Marlboro Reds
5. Cheese

EXTRA CREDIT PART (press enter to continue)-
```

Extra Credit:

```
Please enter the company's name- Apple
```

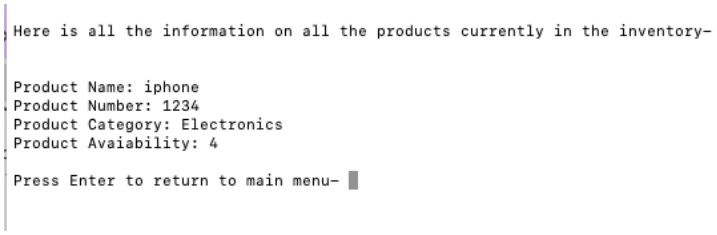
Adding company name

```
Welcome to Apple's Inventory management system!

Main Menu-

Please pick one of the options below to proceed.
1. Add a product to the inventory
2. Remove a product from the inventory
3. Display all products by product number
4. Display detailed information of all products
5. Display available products
6. Search for a product using it's product number
7. Display a list of the most recently searched products (5 maximum)
8. Exit the program

Please enter the number of the option you would like to choose- 1
```

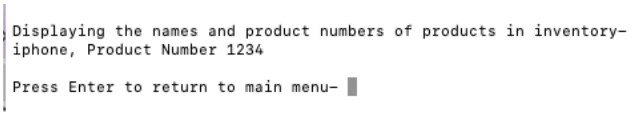
View all options

```
Enter the name of the product- Iphone
```

Adding products



View all products



View Product by product number

Submission Guidelines:

- Upon the completion of your lab, arrange all files required to run the assignment into a single folder, then compress into a single Zip file. Upload the consolidated submission to the Moodle course website by the deadline.
