WAHAB RAZA AKRAM

CHECKOUT SYSTEM REPORT

2023

TABLE OF CONTENTS

INTRODUCTION

In this report, the implementation of the checkout system will be discussed using the programming language Java, this checkout system will be capable of handling pricing schemes for a supermarket. The goal of the system is to accurately calculate the total price of the Items A, B, C and D, taking into account the special offers as well.

The checkout system is designed using Eclipse and JCreator, using object oriented programming in Java. It is designed to be flexible, allowing items to be scanned in any order. This means that if the user scans a B, followed by an A, and then another B the output will be £0.95 meeting client requirements.

The primary objectives for this implementation are:

1. Handle pricing schemes for individual items and special offers.
2. Items can be scanned in any order and the total price will be calculated accurately.
3. Create a user-friendly command-line interface for item input and to display in real time the running total.

In this report I will be going through some details such as the code structure, how is the pricing scheme handled, and how this implementation is user-friendly and easy to use.

OVERVIEW OF THE CLASSES

Three different classes have been created for the Checkout System, Checkout, SpecialPrice, and Calculator.

The Checkout class is the main component for the checkout system, this manages the quantity of the items, the different unit prices, and the special prices.

- The Checkout class interacts with the user through the Scanner class, this allows the program to scan items, remove items and calculate the final total.
- Stores the unit prices in the unit prices array and the quantity of the scanned objects in the items array.
- By using the specialPrices array, the SpecialPrice class was used to save and access the special pricing guidelines for the items.
- The Calculator class is created with a reference to the Checkout class; this allows the total price to be calculated using the scanned products and any special pricing saved in the Checkout arrays.

The SpecialPrice class represents a special pricing rule for an item. Is defining the total quantity of items required to use the special price, for example, if user inputs 3 A's then the SpecialPrice is £0.30 for the third and not £0.50 which is the unit price:

- This class is used by the Checkout class to store and get specific pricing data for each item.
- The Checkout class uses an array of SpecialPrice objects to associate the special pricing rules with specific items

The Calculator class is the one responsible for calculating the total price of the scanned items taking into account the pricing rules and the special prices:

- References to the Checkout class which provides the necessary information about the items, unit prices and special prices.
- The Calculator class uses the methods getSpecialQuantity() and getSpecialPrice() from the Checkout class to get the values of the items.
- It uses the calculateTotal() method to go through all the scanned items applying the special pricing rules and then calculate the total price.

PRICING LOGIC

The pricing logic in the code allows the Unit prices represent the normal price for an item, while special prices provide the discounts required for purchasing multiple quantities of an item.

The Calculator class calculates the total price by going over each item that has been scanned and applying the pricing rules, It considers the Unit price and if possible the applicable special prices which have been set for some items.

The methods getSpecialQuantity and getSpecialPrice are expalined below:

getSpecialQuantity:

- The purpose of this method is to get the quantity required to activate the special price for a specific item.
- It accesses the specialPrices array in the Checkout class, which has the SpecialPrice objects linked with each item.
- The getSpecialQuantity method is called in the calculateTotal to check how many times the special price should be used.

getSpecialPrice:

- The purpose of this method is to get the special price for a specific item A or B in our case.
- It accesses the specialPrices array in the Checkout class which stores the SpecialPrice objects associated with each item.
- This method is called within the calculateTotal method of the Calculator class to get the special price for an item and if there is no special price for the item (C and D have no special price set) it uses the unit price.

USER INTERACTION

The user interaction utilizing the Scanner class.

Scanning items:

- Using the System.out.println("1. To scan an item, press 1") statement the code tells the user to enter an item that they want to scan.
- After the user has scanned an item using the 'Scanner' class this calls the 'next()' method which reads the item entered by the user.
- And then the item is passed to the 'checkout.scan(item)' method, which updates the quantity of the scanned item in the 'items' array of the 'checkout' class.

Selecting choice from the Menu:

- The menu shows different options when starting the program to the user such as Scan an item, remove an item or get the total final price.
- The user choice is obtained using the scanner class by calling the 'nextInt()' method, this method reads the next integer entered by the user (the menu only accepts integers as an option to input).
- Created conditional statements such as 'if', 'else if', 'else' to achieve the required actions based on the user input.

When displaying the running total and the final total:
Running total:

- After the user scans an item or removes an item, the code calls the finalCalculation.calculateTotal() method to calculate the running total which is later displayed for the user to know how much are they spending using the 'System.out.println("Running total: £" + total)'

Final total:

- The menu displays 3 options, as mentioned before 2 are when the user wants to scan items or remove items and the third one is to finish the transaction which if the user inputs 3 the loop breaks and calculates the final total.
- The final total price is calculated again calling the finalCalculation.calculateTotal() method again.
- And to finish the program the final total is displayed to the user using the System.out.println("Thanks for your purchase, your final total is: £" + finalTotal) and the scanner gets closed by using scanner.close().

TESTING

Test number	Description	Expected result	Actual result	Pass/Fail	Comments
1	Menu User input: 1	To give option to scan the different items	Scan item A, B, C or D:	Pass	Works, lets the user select what item they want to scan.
2	Menu User input: 2	To give option to remove an item.	Enter the item to remove:	Pass	Works, lets the user remove item.
3	Menu User input: 3	To break the loop and give final total price.	Thanks for your purchase, your final total is: £0.00	Pass	Works, program outputs the final total.
4	Scanning items, If scanned B, A and another B	To give running total: £0.95	Output: Running total: £0.95	Pass	Calculates the unit prices and special prices effectively
5	Scanning items, if scanned A, A and another A	To give running total: £1.30	Output: Running total: £1.30	Pass	Calculates the unit prices and special prices effectively
6	Scanning item C	Running total should be £0.20	Output: Running total: £0.20	Pass	This item has no special price assigned, price will always stay at £0.20 each
7	Scanning item D	Running total should be £0.15	Output: Running total: £0.15	Pass	This item has no special price assigned, price will always stay at £0.15 each
8	Delete item after adding, example: Adding A and deleting A	Should output running total: £0.50 and then input 2 in menu to remove item and remove A, running total will be £0.00	Output: Running total is £0.50 when A item is added and when user wants to remove A the running total is £0.00	Pass	Works, after adding different items to the checkout shows the running total and if removed works perfect.

IMPROVEMENTS FOR THE FUTURE

While I was doing the program, I found some improvements that could be made to make it easier and user-friendly. Due to limited time, the improvements can be made in future interactions.

- Instead of using array lists, using the HashMap, to improve this project I will work towards getting the 'HashMap' data structure to store the quantities as this gives more flexibility and efficient item management.

- Instead of a command line program, create a graphical user interface (GUI) using a Java GUI library like JavaFX, this can be more appealing and organized way for users to interact with the checkout system

- Make infinite items, make a loop that goes through A to Z and then goes from A1 to Z1, A2 to Z2, A3 to Z3 etc..., the type was chosen as a string instead of a char to give flexibility for this into future versions of the program.

- To implement unit tests using a testing framework like JUnit to ensure the correctness/effectiveness of the code functionality.

- Change the menu interface in the command line so it does not loop every time an item is scanned, and it remains inside option n1 until selected to show final total.

CONCLUSION

In conclusion, I have enjoyed working in this project a lot, developing this checkout system in Java has helped me to learn a bit more and improve my programming skills, I have gained practical knowledge in implementing pricing logic and the user interaction.

While I was coding this project, I encountered various challenges, such as designing an efficient data structure to store the item information and handling the user input validation, I have added as an extra a Menu where the users can choose what they want to do (Scan, remove or finish), This challenges have provided me valuable learning opportunities and understand more java concepts.

I will continue to work on the checkout system and try to implement the changes suggested in this report at my best capacity, and to see a newer and fresher version 2.0.