**Week-04**

Name: Shajih Baig

Course:

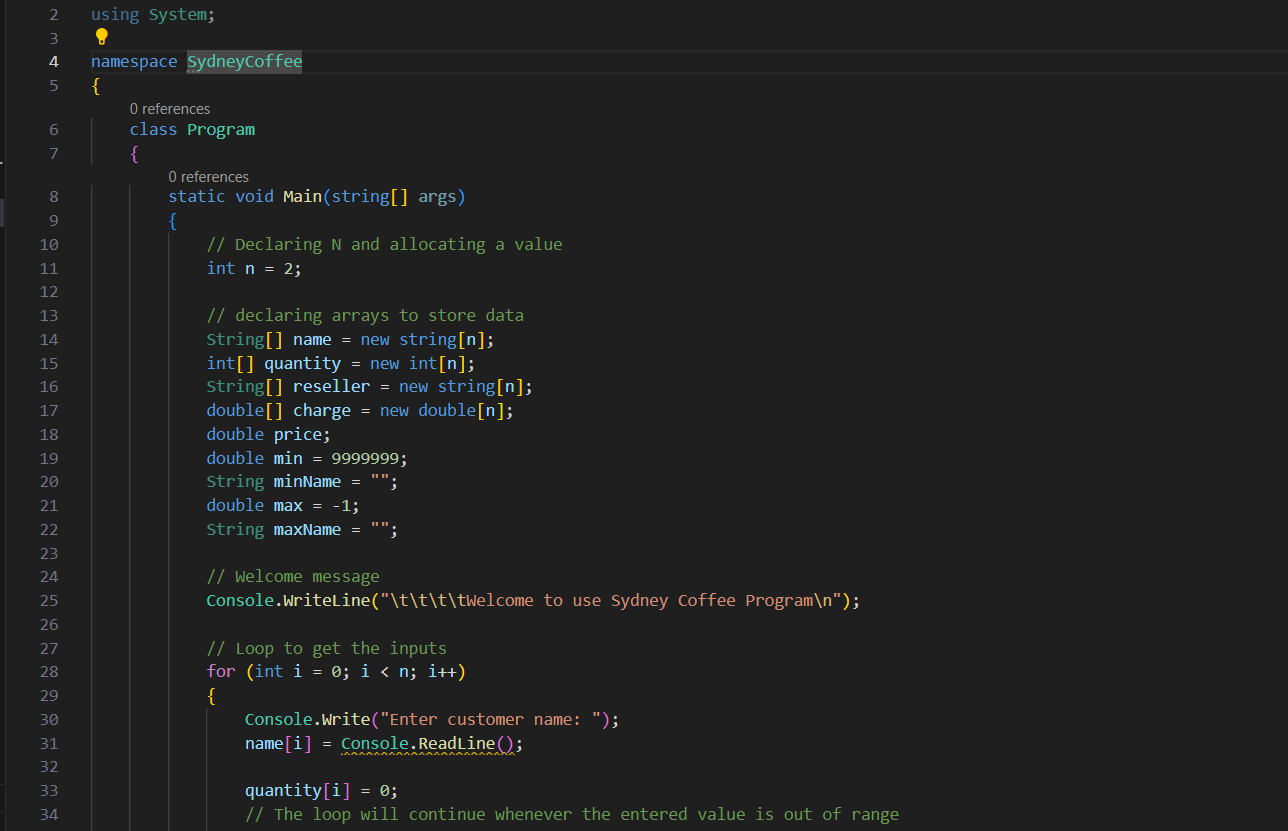
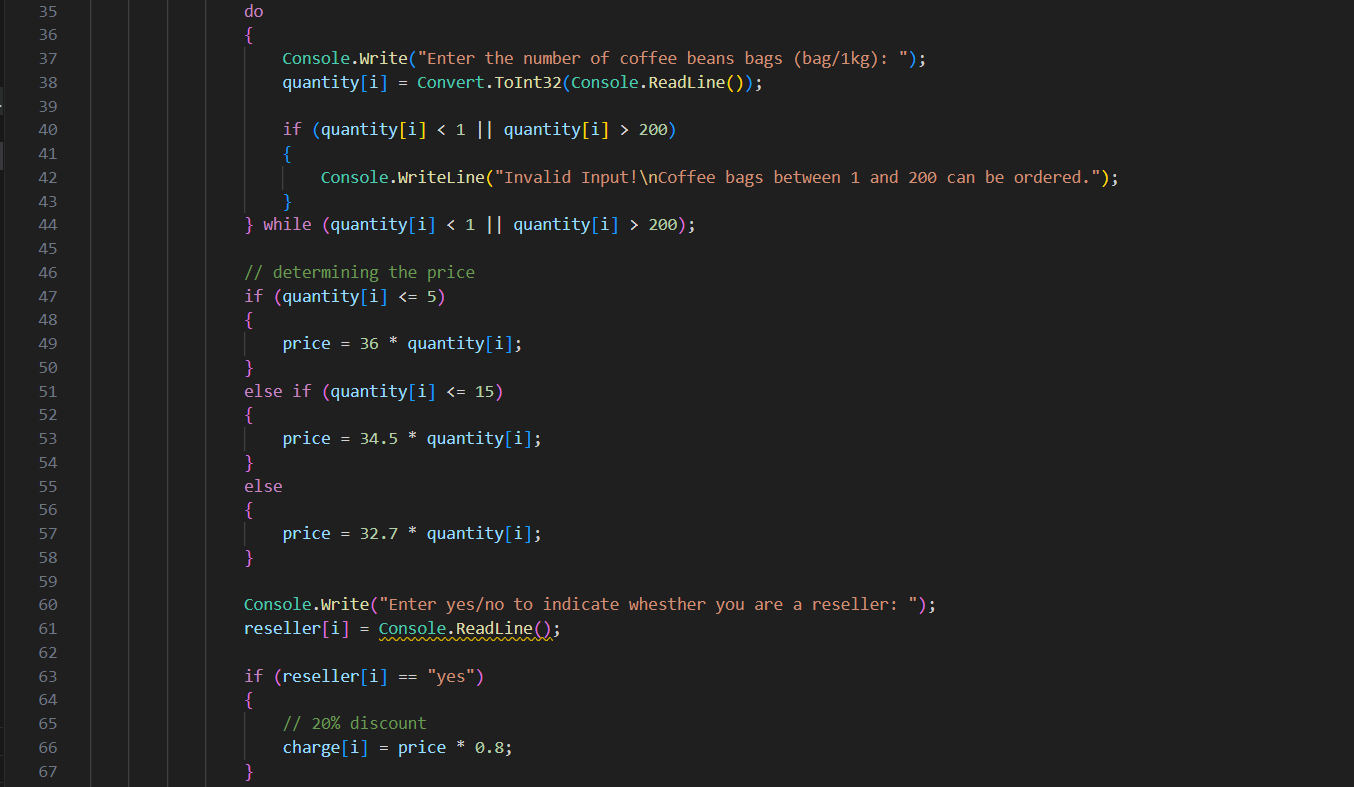
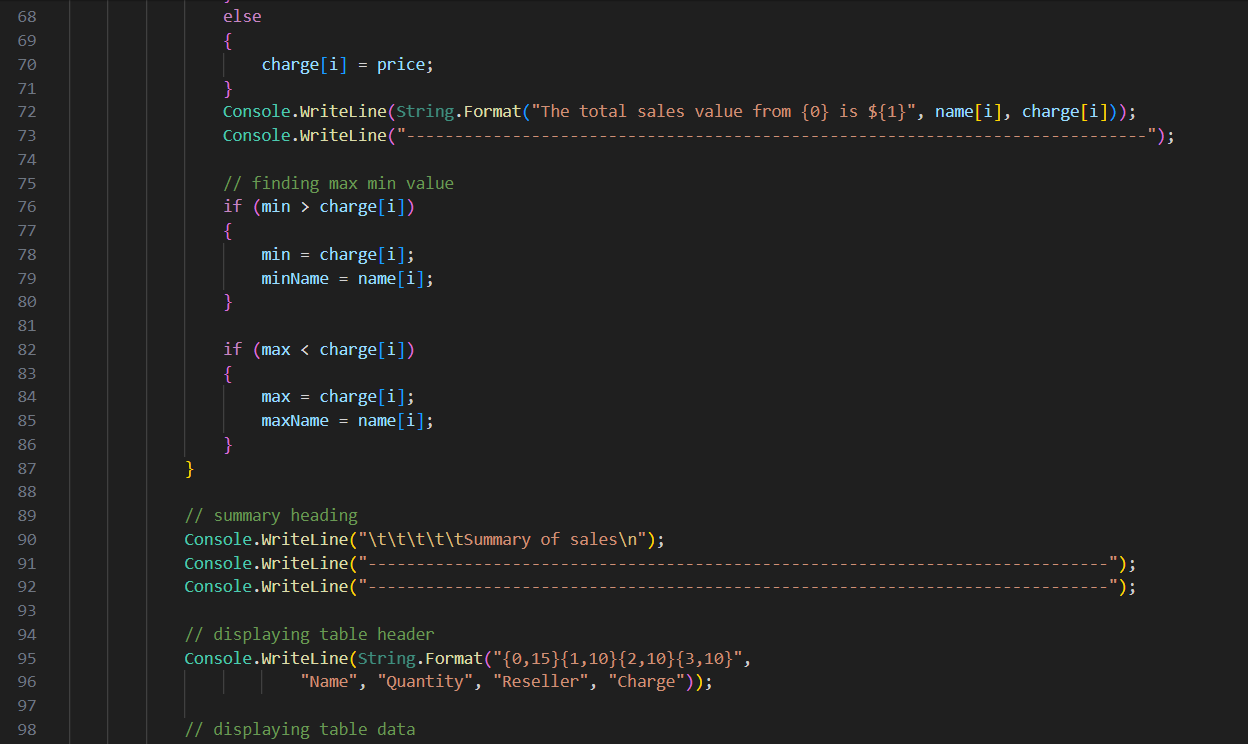
Instructor:

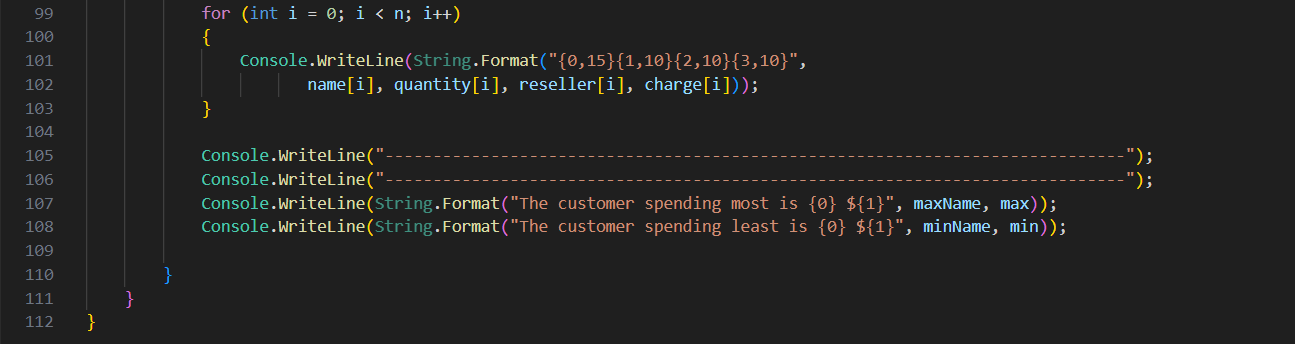
Section:

Semester:

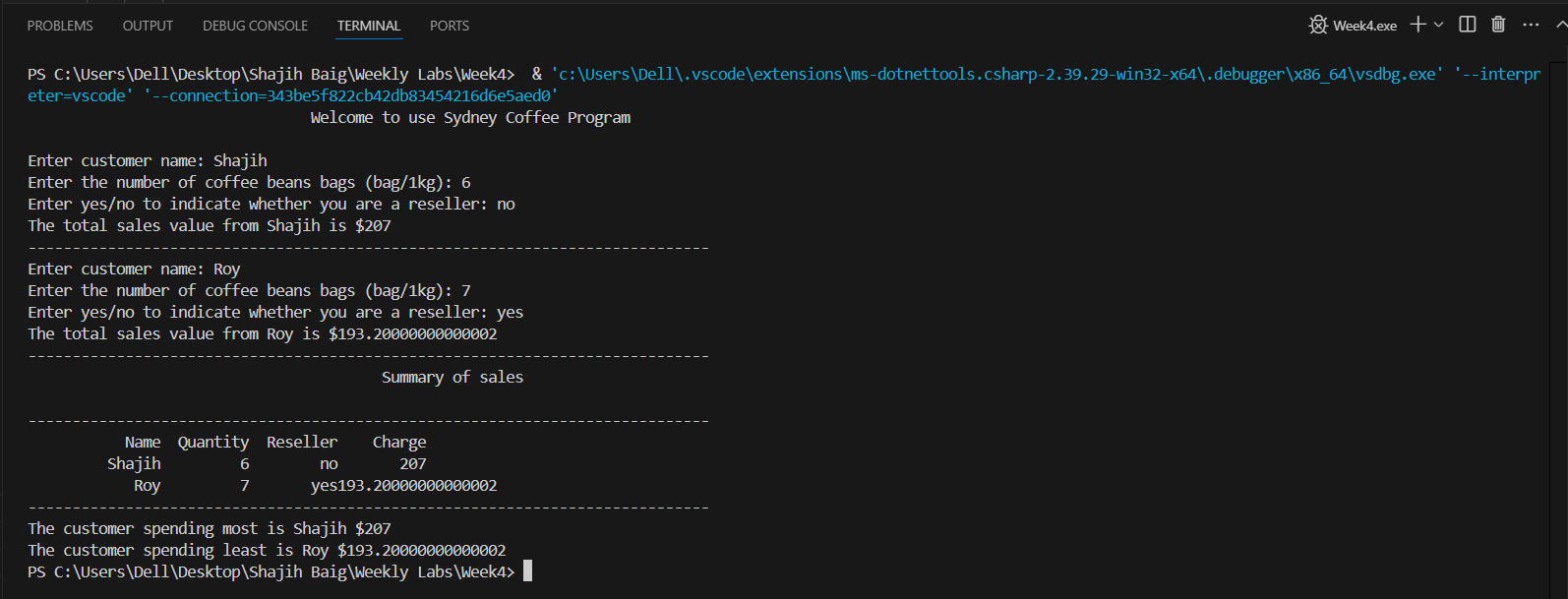
Date: 10-08-2024

**Task-01:** Code (provided) screenshot.

**Code:**  



**Output:**



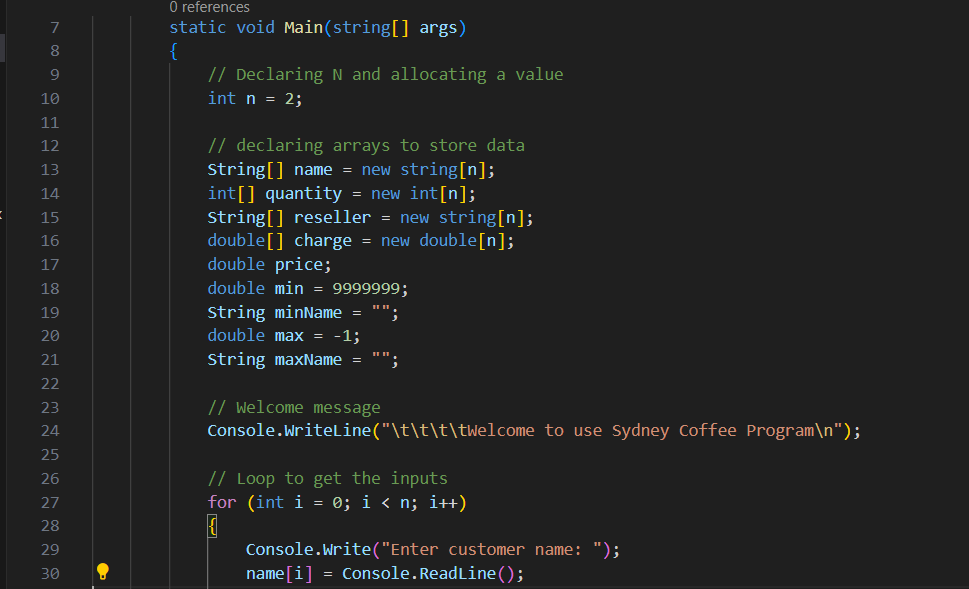
**Task-02:** Description ofcode (provided).

The provided code is designed to manage coffee sales for a small shop. It starts by welcoming the user and then prompts them to input details for two customers, including their names, the number of coffee bags they want to buy, and whether they are resellers. The program calculates the total charge based on the quantity of coffee bags, applying different price rates depending on the number of bags. If the customer is a reseller, a 20% discount is applied. The program keeps track of which customer spends the most and the least, and finally, it displays a summary of all the sales, showing each customer’s name, quantity purchased, reseller status, and the final charge. The code uses arrays to store the data for each customer and loops to handle multiple entries. Simple conditional statements are used to calculate prices and apply discounts, making the program functional but with room for improvement in terms of efficiency and structure.

**Task-03:** Redesigning and recoding.

1. **Use of Functions.**

**Old:**



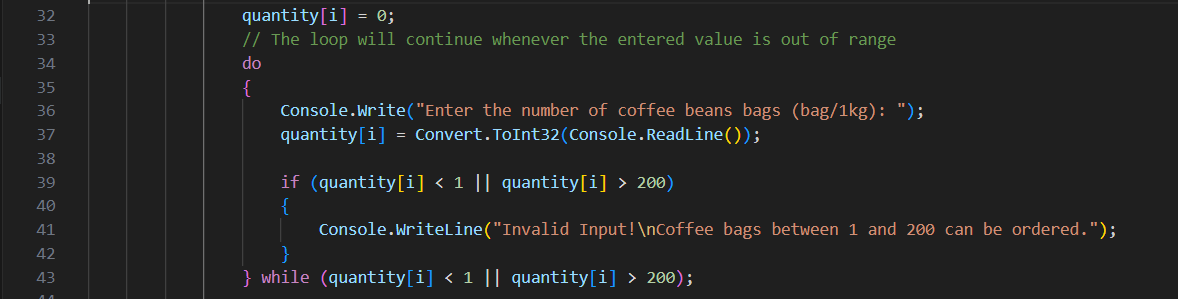
**New:**



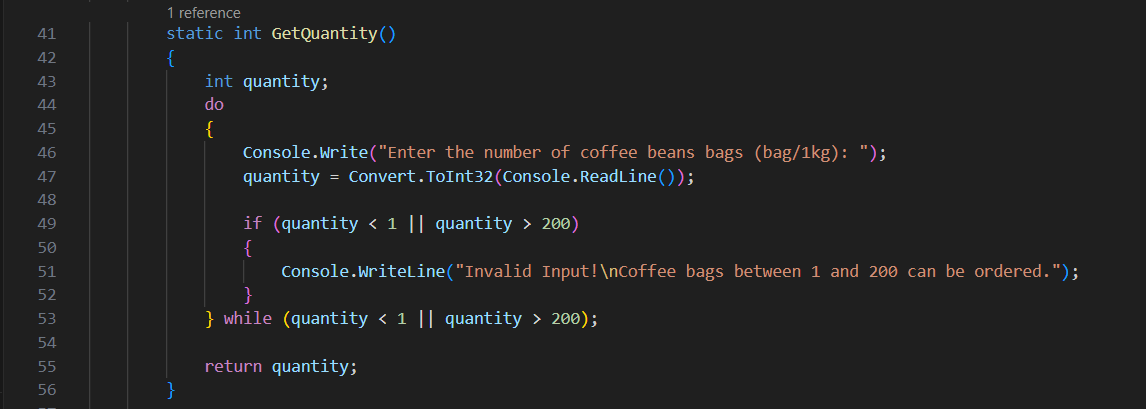
**Reason:** Breaking the code into functions to improve code readability and reusability, making it easier to debug and understand each part's role.

1. **Loop Structure & Input handling.**

**Old:**

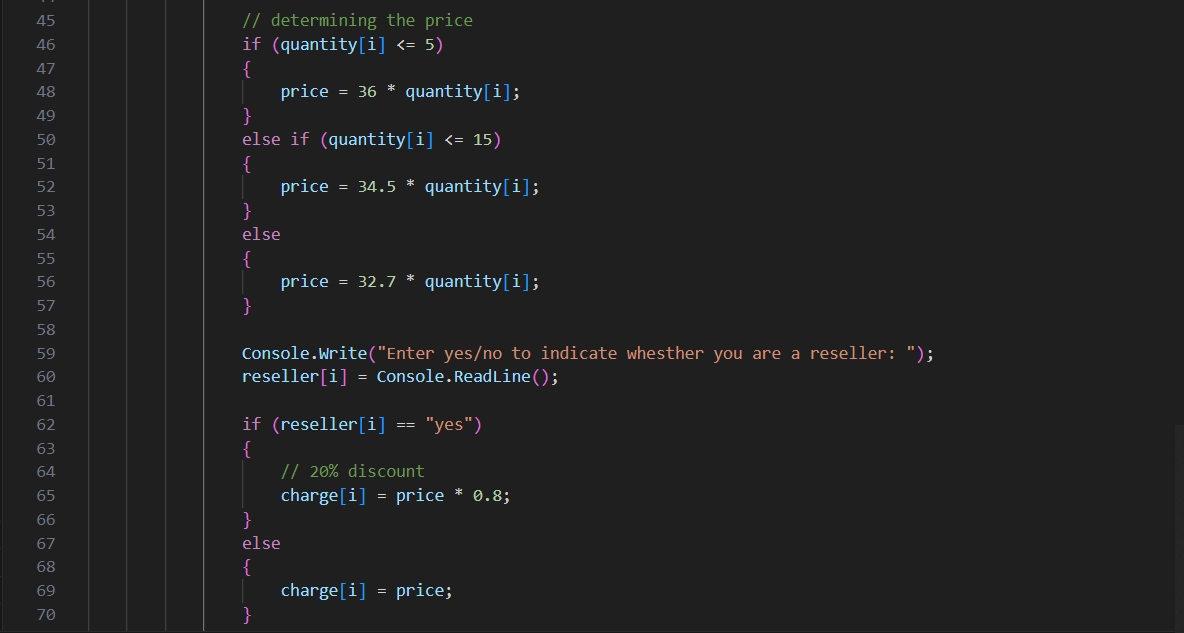


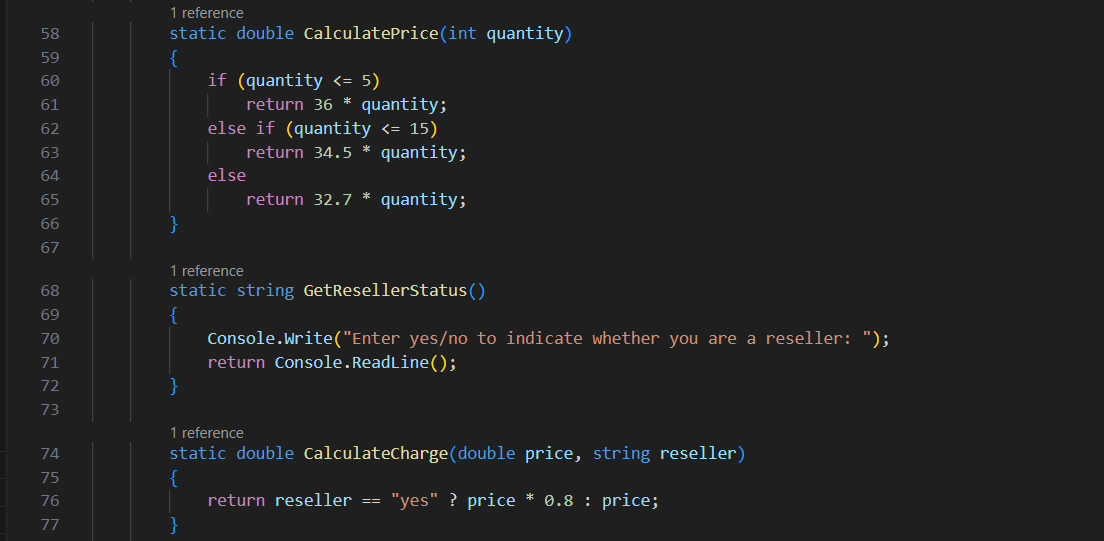
**New:**



**Reason:** The updated loop and input handling reduce repetitive code, enhance error checking and easier to extend, ensuring more reliable user input.

1. **Calculation & Display logic.**

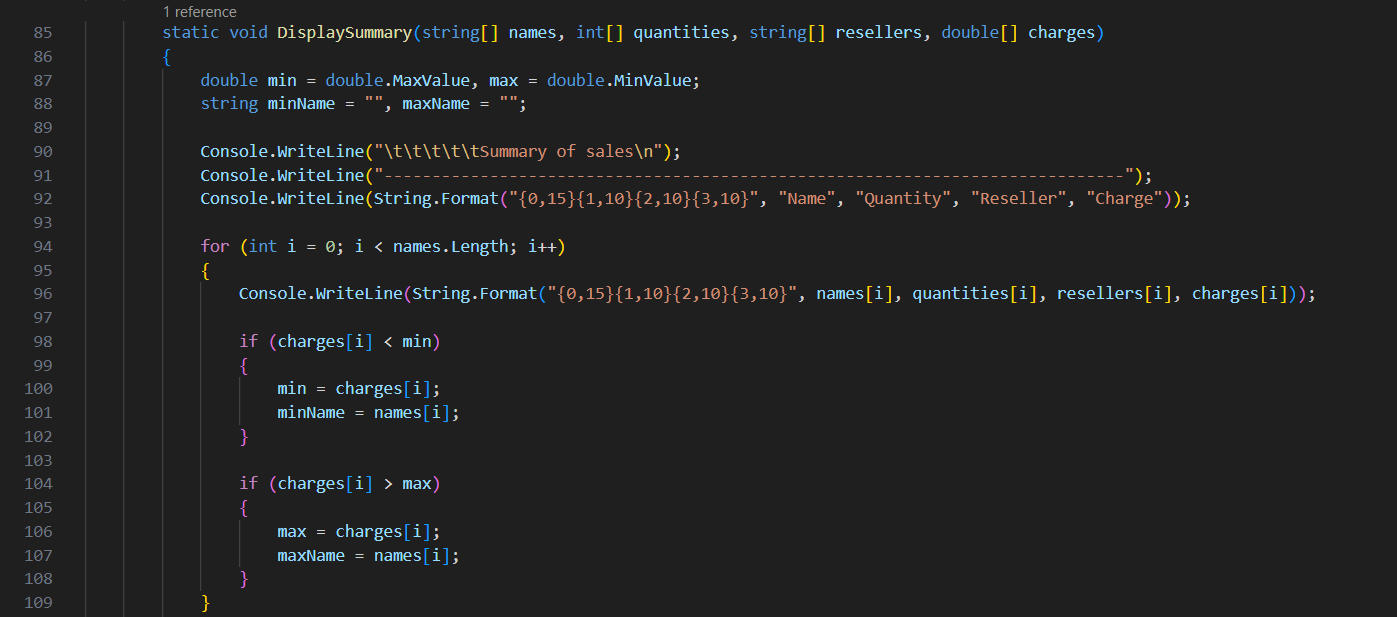
**Old:** 

**New:** 

**Reason:** Separating calculation and display logic into functions simplifies the main program flow, making the code easier to maintain and modify.

1. **Summary display.**

**Old:** 

**New:** 

**Reason:** The old code had sales value display embedded in the loop, making it repetitive. The new code encapsulates this in a function, reducing redundancy and enhancing clarity.

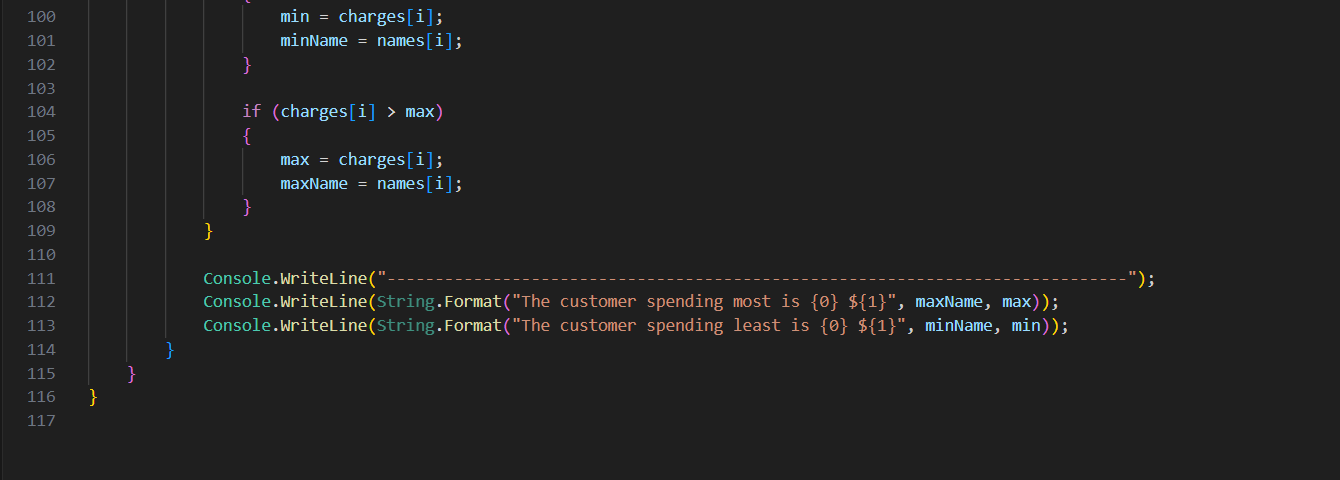
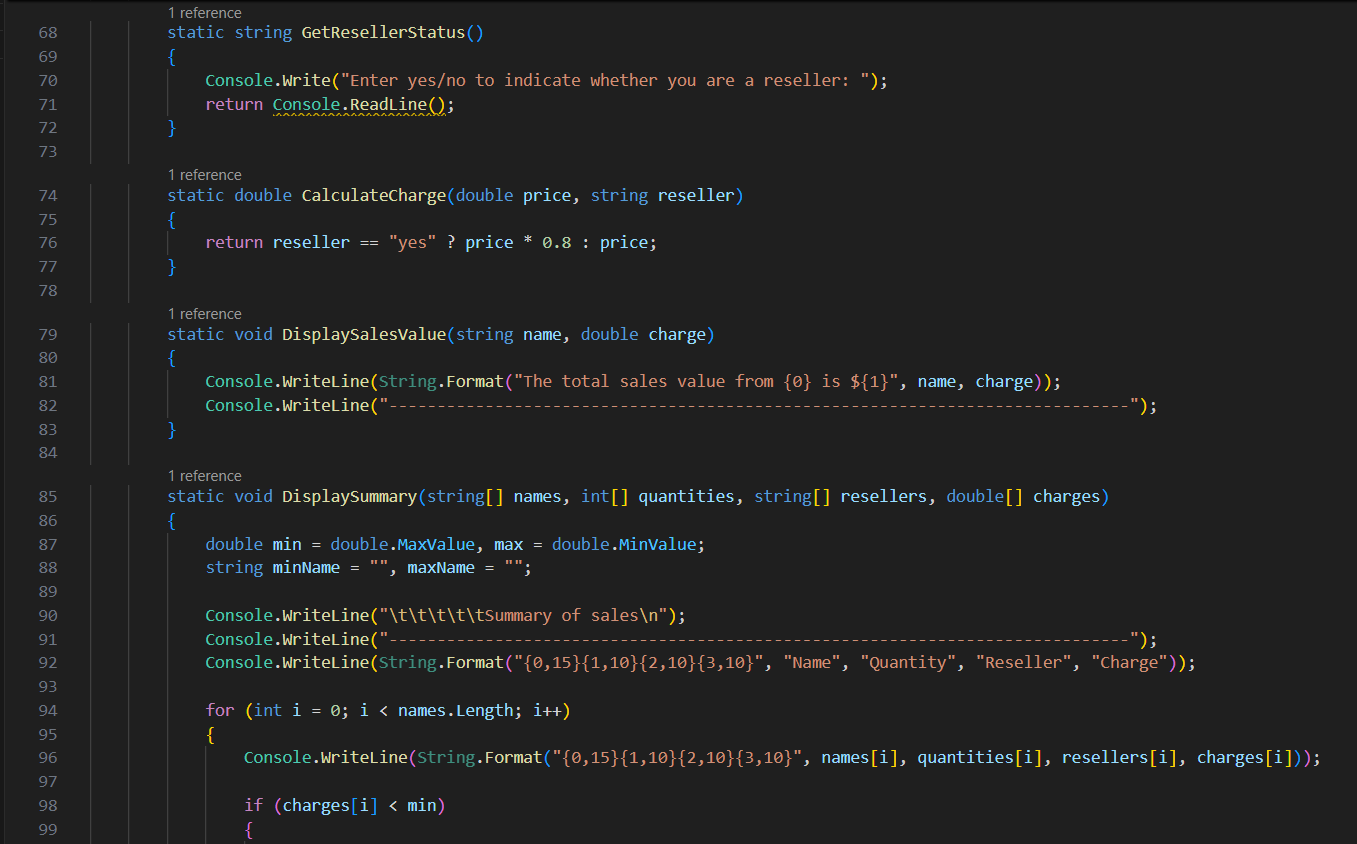
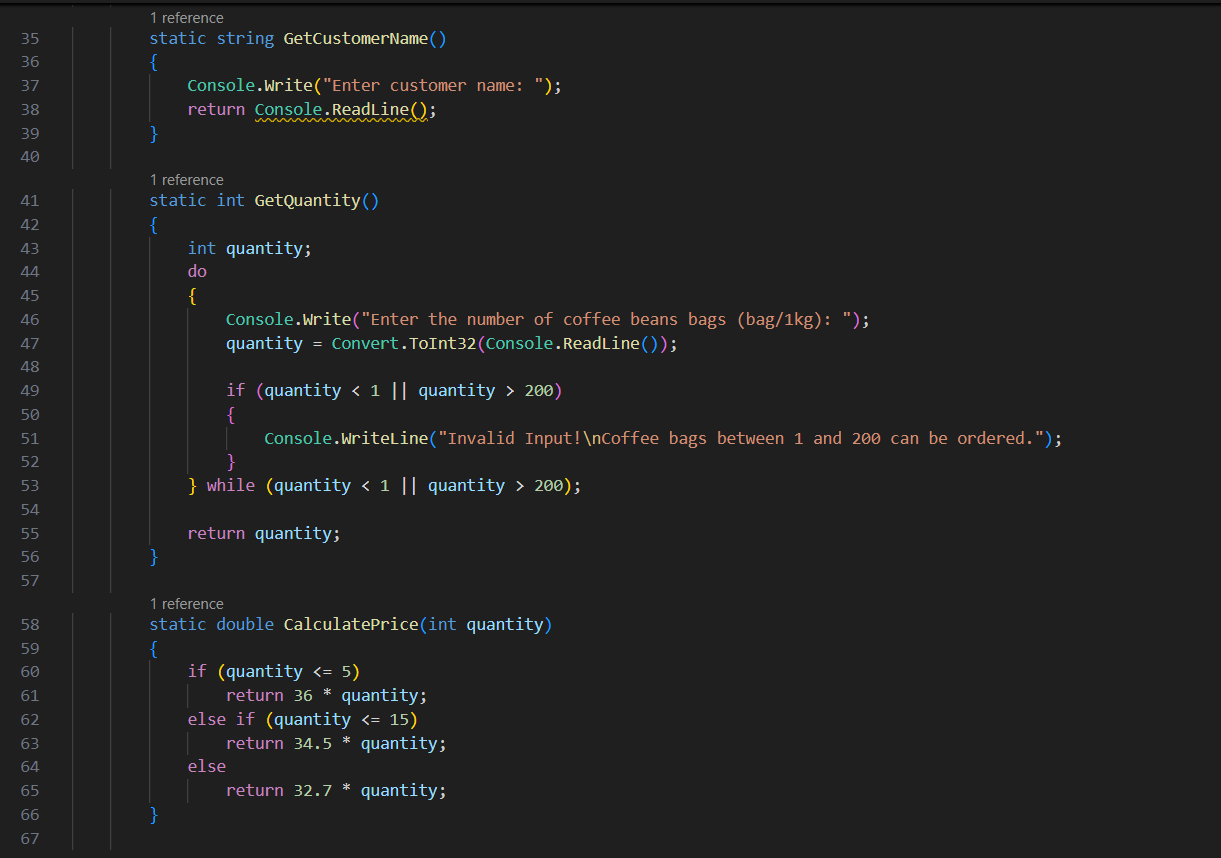
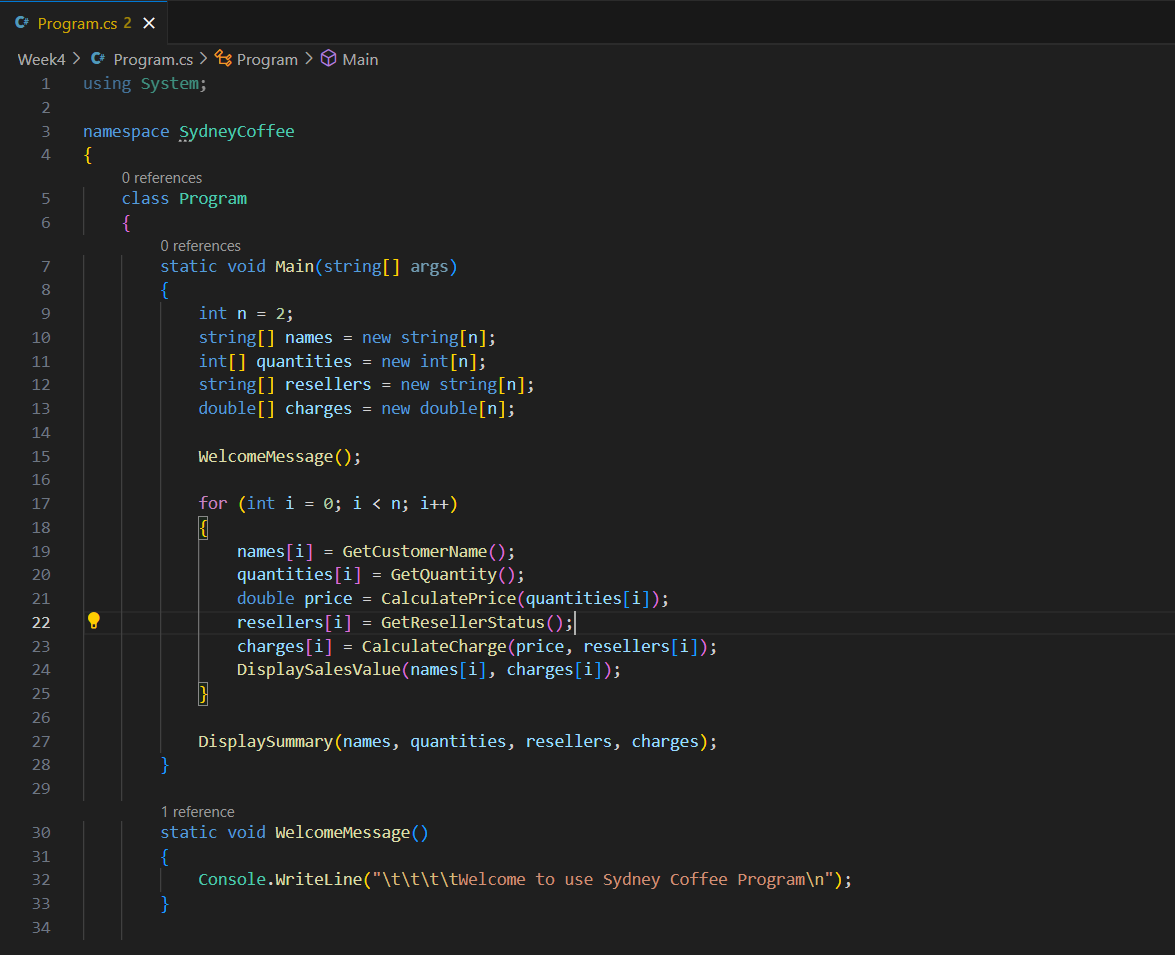
**Report**

**Introduction:** This program is designed to manage customer orders for small cafe by reading customer details, calculating total charges based on order size and reseller status, and summarizing the sales. The goal is to streamline order processing and provide clear summaries, making the program efficient and user-friendly.

**Summary (Original):** The original program used a single, large main function to handle all tasks—data input, calculations, and display. It relied on arrays to store customer information and used conditional statements within the loop to calculate charges based on order quantity and reseller status. The program also included logic for finding the highest and lowest spenders within the same loop, making the structure less modular and harder to manage.

**Summary (Updated):** The updated program breaks down the tasks into separate, well-defined functions. It still uses arrays for storing data but organizes the code into distinct parts: collecting input, calculating charges, and displaying results. This modular approach improves readability, facilitates debugging, and makes future modifications easier.

**Conclusion:** The program’s core logic remains the same, but the new version greatly improves structure and clarity. With the use of functions and optimized loop handling, the code is more efficient and easier to understand. This approach ensures the program is more maintainable and ready for future enhancements while preserving its original functionality.

**Updated Code:**

**Output:**

