COMSATS UNIVERSITY ISLAMABAD
ATTOCK CAMPUS

ASSIGNMENT # 01

Submitted By:

Marwa Sajjad.

Registration Number:

(Sp22-BSE-035).

Course:

Mobile Application Development.

Date:

September 26,2024.

Submitted To:

Mr. Muhammad Kamran.

Contents

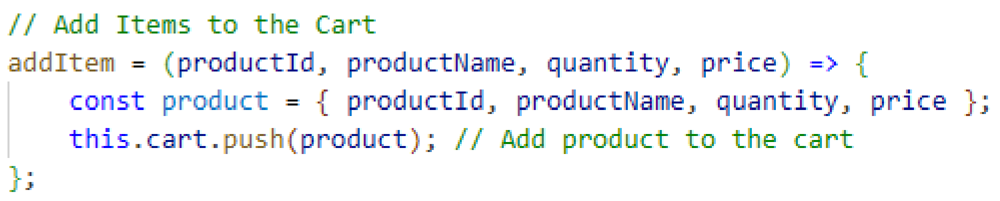
Introduction:

The provided code defines a ShoppingCart class that simulates the functionality of a simple shopping cart system. This class allows users to add, remove, and update items, calculate the total cost of the items in the cart, display a summary of the cart, and apply discount codes.

Code Explanation:

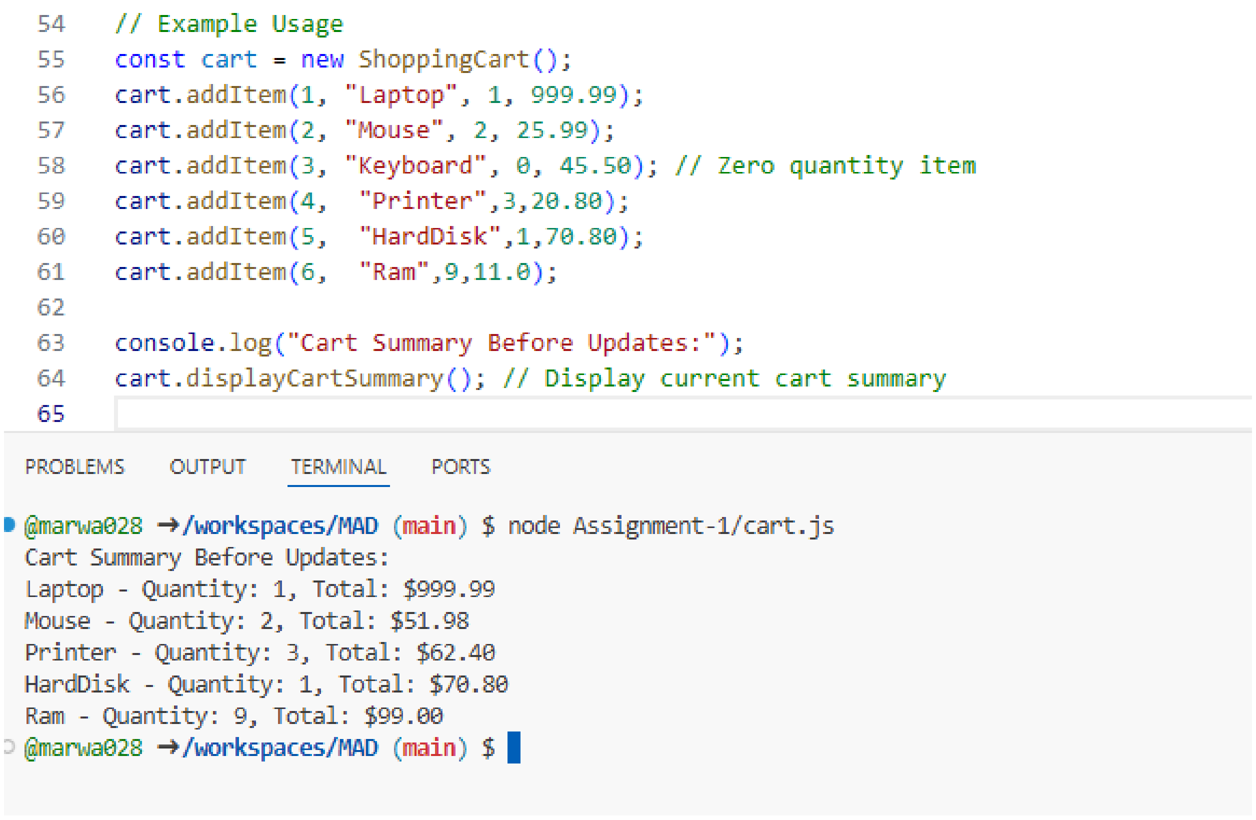
1. **Class Definition:** The ShoppingCart class is defined with a constructor that initializes an empty array named cart, which will hold the items.
2. **Add Items:** The addItem method takes parameters for product ID, name, quantity, and price. It creates a product object and pushes it to the cart array.
3. **Remove Items:** The removeItem method filters out an item from the cart based on its product ID. This is achieved by creating a new array that excludes the item with the specified ID.
4. **Update Item Quantity:** The updateQuantity method updates the quantity of a specific item in the cart. It uses the map method to return a new array where the specified product's quantity is updated.
5. **Calculate Total Cost:** The calculateTotalCost method computes the total cost of all items in the cart using the reduce method. It multiplies the price by the quantity for each item and sums them up.
6. **Display Cart Summary:** The displayCartSummary method generates a summary of the cart, filtering out items with zero quantity. It maps over the cart to create a simplified view and logs each item's name, quantity, and total cost to the console.
7. **Apply Discount Code:** The applyDiscount method takes a discount code, checks for its validity in a predefined set of codes, and calculates the total cost after applying the discount. If the code is not found, it defaults to zero discount.

➢ Add Items:



```javascript
// Add Items to the Cart
addItem = (productId, productName, quantity, price) => {
    const product = { productId, productName, quantity, price };
    this.cart.push(product); // Add product to the cart
};
```

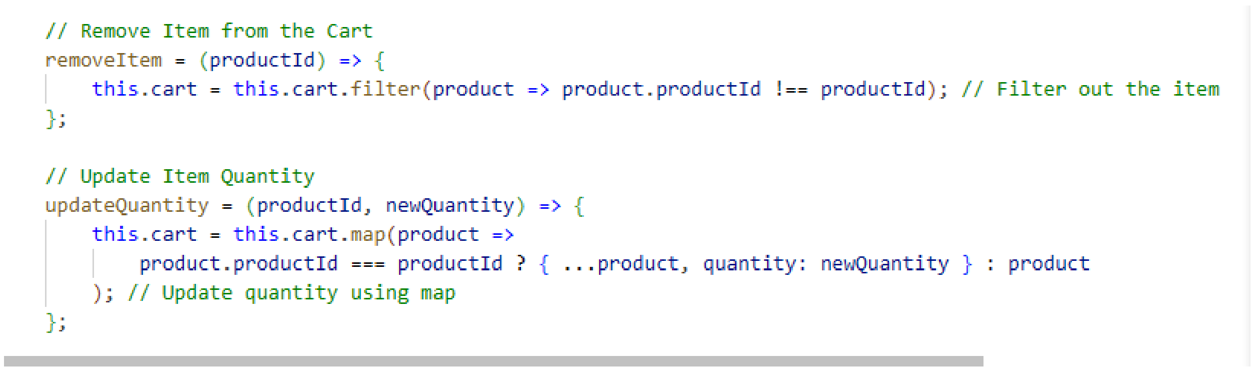
Output:



```javascript
// Example Usage
const cart = new ShoppingCart();
cart.addItem(1, "Laptop", 1, 999.99);
cart.addItem(2, "Mouse", 2, 25.99);
cart.addItem(3, "Keyboard", 0, 45.50); // Zero quantity item
cart.addItem(4,  "Printer",3,20.80);
cart.addItem(5,  "HardDisk",1,70.80);
cart.addItem(6,  "Ram",9,11.0);

console.log("Cart Summary Before Updates:");
cart.displayCartSummary(); // Display current cart summary

```

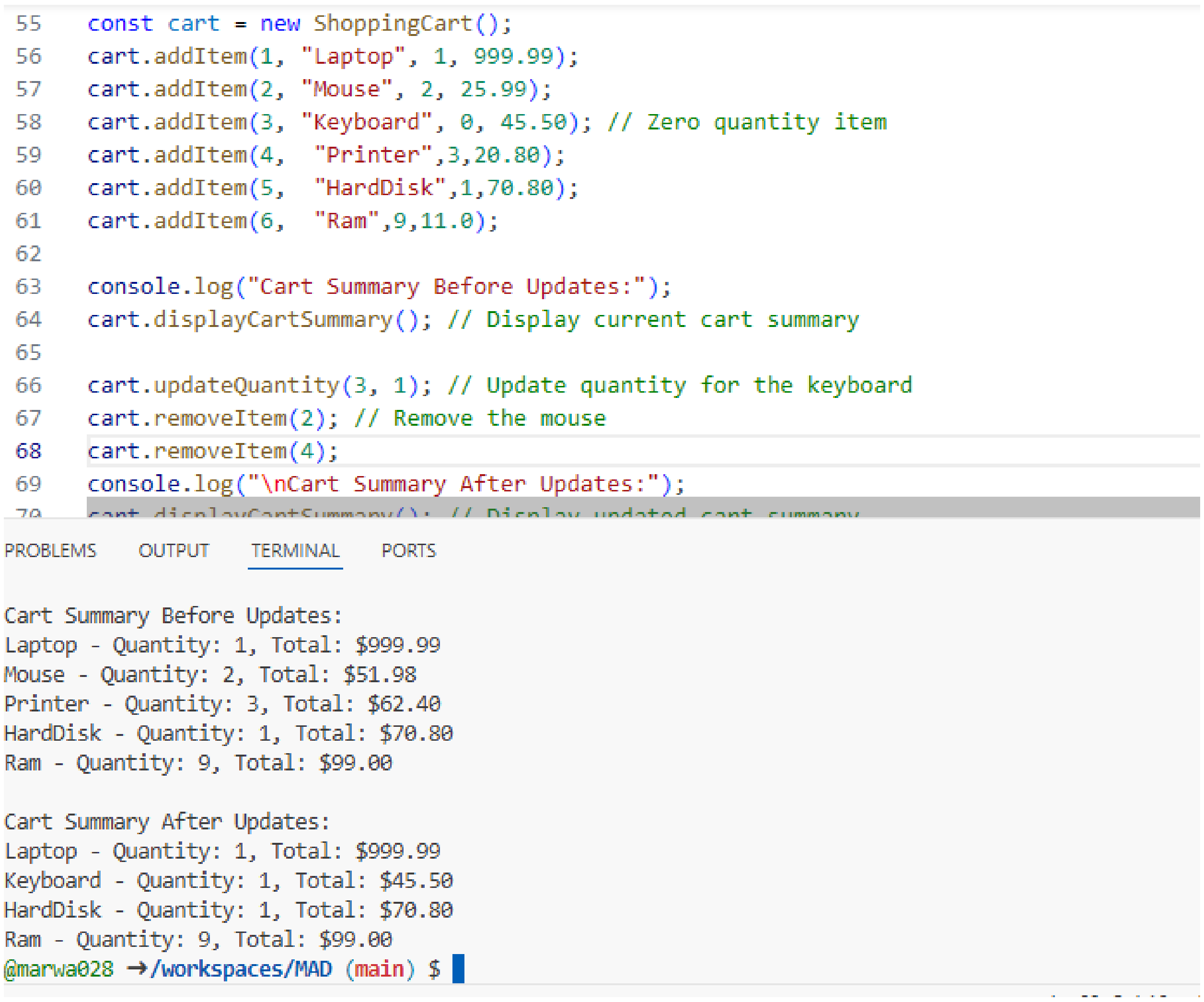
```
@marwa028 ➜ /workspaces/MAD (main) $ node Assignment-1/cart.js
Cart Summary Before Updates:
Laptop - Quantity: 1, Total: $999.99
Mouse - Quantity: 2, Total: $51.98
Printer - Quantity: 3, Total: $62.40
HardDisk - Quantity: 1, Total: $70.80
Ram - Quantity: 9, Total: $99.00
@marwa028 ➜ /workspaces/MAD (main) $
```

➢ Remove Items:



```
// Remove Item from the Cart
removeItem = (productId) => {
    this.cart = this.cart.filter(product => product.productId !== productId); // Filter out the item
};

// Update Item Quantity
updateQuantity = (productId, newQuantity) => {
    this.cart = this.cart.map(product =>
        product.productId === productId ? { ...product, quantity: newQuantity } : product
    ); // Update quantity using map
};
```

Output:

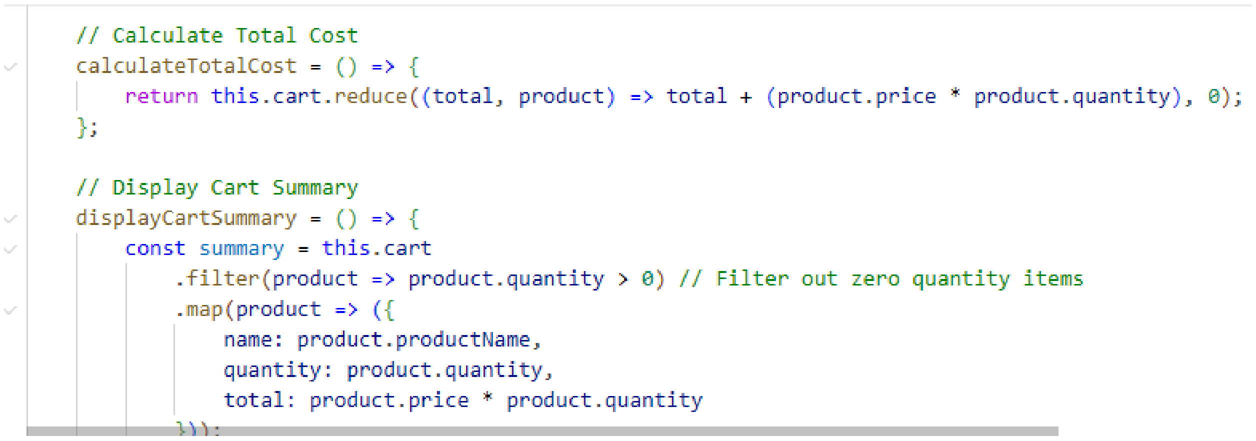


```
const cart = new ShoppingCart();
cart.addItem(1, "Laptop", 1, 999.99);
cart.addItem(2, "Mouse", 2, 25.99);
cart.addItem(3, "Keyboard", 0, 45.50); // Zero quantity item
cart.addItem(4,  "Printer",3,20.80);
cart.addItem(5,  "HardDisk",1,70.80);
cart.addItem(6,  "Ram",9,11.0);

console.log("Cart Summary Before Updates:");
cart.displayCartSummary(); // Display current cart summary

cart.updateQuantity(3, 1); // Update quantity for the keyboard
cart.removeItem(2); // Remove the mouse
cart.removeItem(4);
console.log("\nCart Summary After Updates:");
cart.displayCartSummary(); // Display updated cart summary
```

PROBLEMS   OUTPUT   TERMINAL   PORTS

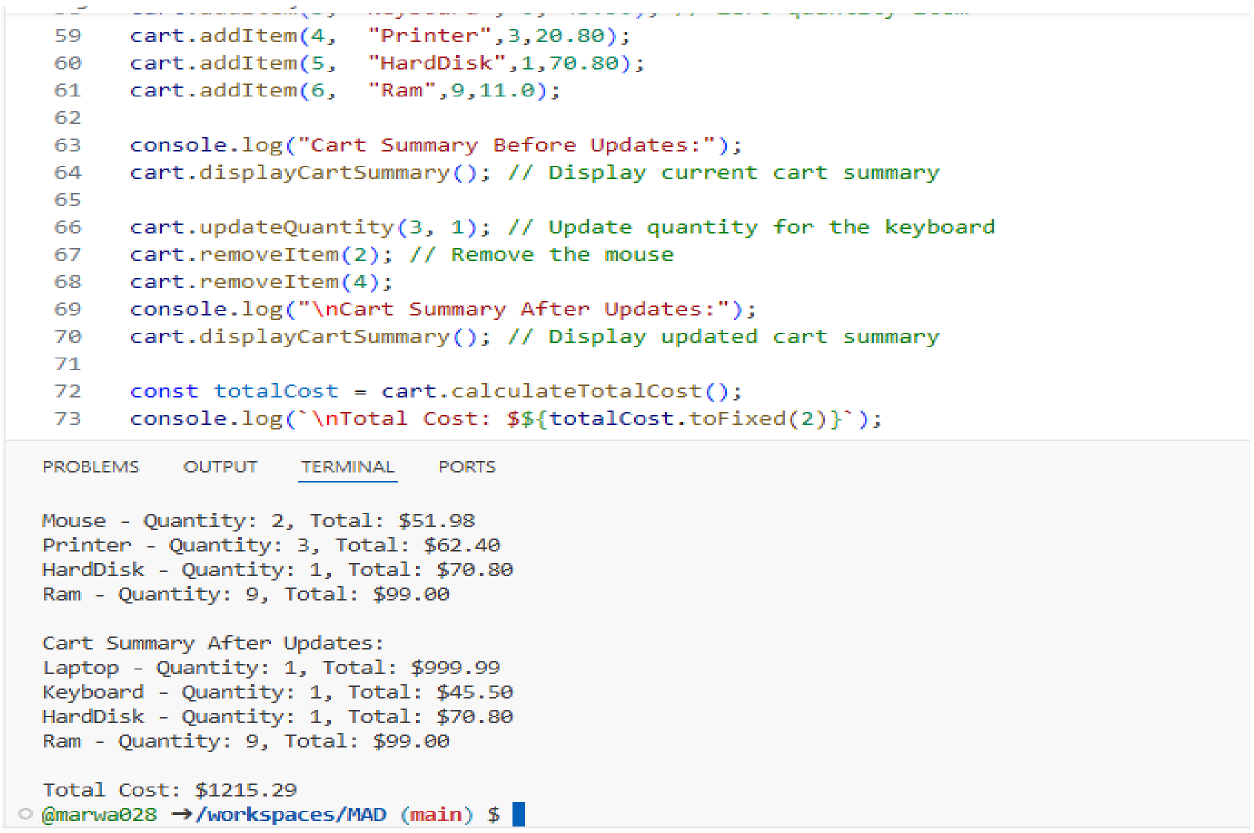
```
Cart Summary Before Updates:
Laptop - Quantity: 1, Total: $999.99
Mouse - Quantity: 2, Total: $51.98
Printer - Quantity: 3, Total: $62.40
HardDisk - Quantity: 1, Total: $70.80
Ram - Quantity: 9, Total: $99.00

Cart Summary After Updates:
Laptop - Quantity: 1, Total: $999.99
Keyboard - Quantity: 1, Total: $45.50
HardDisk - Quantity: 1, Total: $70.80
Ram - Quantity: 9, Total: $99.00
@marwa028 ➜/workspaces/MAD (main) $
```

➢ **Calculate total cost and cart summary:**



```
// Calculate Total Cost
calculateTotalCost = () => {
    return this.cart.reduce((total, product) => total + (product.price * product.quantity), 0);
};

// Display Cart Summary
displayCartSummary = () => {
    const summary = this.cart
        .filter(product => product.quantity > 0) // Filter out zero quantity items
        .map(product => ({
            name: product.productName,
            quantity: product.quantity,
            total: product.price * product.quantity
```

Output:

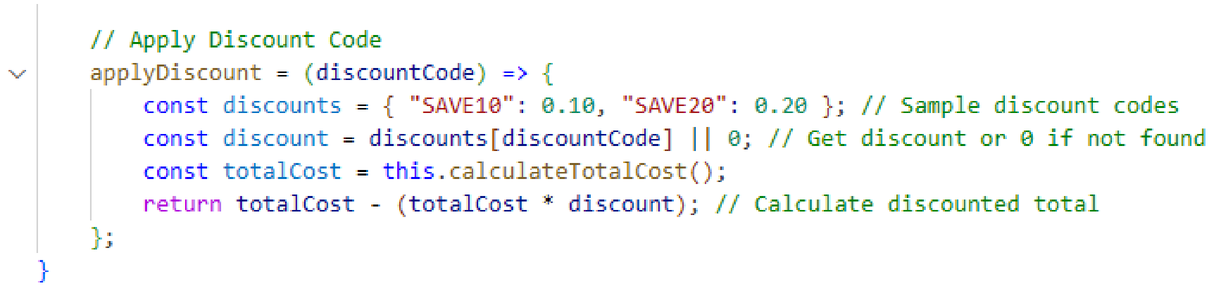


```
cart.addItem(4,  "Printer",3,20.80);
cart.addItem(5,  "HardDisk",1,70.80);
cart.addItem(6,  "Ram",9,11.0);

console.log("Cart Summary Before Updates:");
cart.displayCartSummary(); // Display current cart summary

cart.updateQuantity(3, 1); // Update quantity for the keyboard
cart.removeItem(2); // Remove the mouse
cart.removeItem(4);
console.log("\nCart Summary After Updates:");
cart.displayCartSummary(); // Display updated cart summary

const totalCost = cart.calculateTotalCost();
console.log(`\nTotal Cost: $${totalCost.toFixed(2)}`);
```

PROBLEMS OUTPUT TERMINAL PORTS

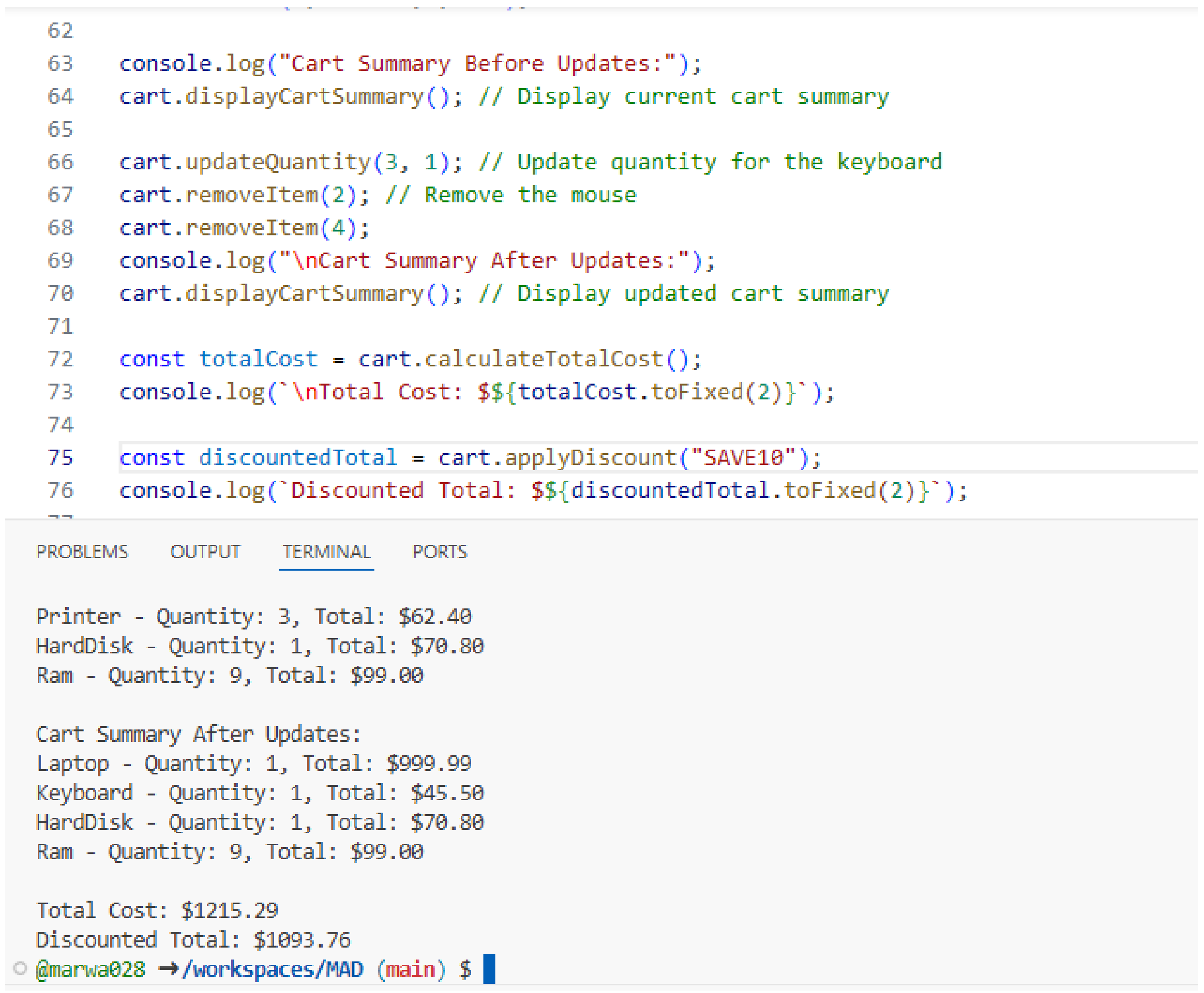
```
Mouse - Quantity: 2, Total: $51.98
Printer - Quantity: 3, Total: $62.40
HardDisk - Quantity: 1, Total: $70.80
Ram - Quantity: 9, Total: $99.00

Cart Summary After Updates:
Laptop - Quantity: 1, Total: $999.99
Keyboard - Quantity: 1, Total: $45.50
HardDisk - Quantity: 1, Total: $70.80
Ram - Quantity: 9, Total: $99.00

Total Cost: $1215.29
@marwa028 ➜/workspaces/MAD (main) $
```

- **Discount on Items:**



```
    // Apply Discount Code
    applyDiscount = (discountCode) => {
        const discounts = { "SAVE10": 0.10, "SAVE20": 0.20 }; // Sample discount codes
        const discount = discounts[discountCode] || 0; // Get discount or 0 if not found
        const totalCost = this.calculateTotalCost();
        return totalCost - (totalCost * discount); // Calculate discounted total
    };
}
```

Output:



```

console.log("Cart Summary Before Updates:");
cart.displayCartSummary(); // Display current cart summary

cart.updateQuantity(3, 1); // Update quantity for the keyboard
cart.removeItem(2); // Remove the mouse
cart.removeItem(4);
console.log("\nCart Summary After Updates:");
cart.displayCartSummary(); // Display updated cart summary

const totalCost = cart.calculateTotalCost();
console.log(`\nTotal Cost: $${totalCost.toFixed(2)}`);

const discountedTotal = cart.applyDiscount("SAVE10");
console.log(`Discounted Total: $${discountedTotal.toFixed(2)}`);
```

```
Printer - Quantity: 3, Total: $62.40
HardDisk - Quantity: 1, Total: $70.80
Ram - Quantity: 9, Total: $99.00

Cart Summary After Updates:
Laptop - Quantity: 1, Total: $999.99
Keyboard - Quantity: 1, Total: $45.50
HardDisk - Quantity: 1, Total: $70.80
Ram - Quantity: 9, Total: $99.00

Total Cost: $1215.29
Discounted Total: $1093.76
@marwa028 ➜/workspaces/MAD (main) $
```

Learning Outcomes:

From this code, I learned several important concepts:

- Encapsulation: The ShoppingCart class encapsulates all cart-related functionalities, making it easier to manage the shopping process within a single unit.
- Array Manipulation: The use of methods like map, filter, and reduce demonstrates effective ways to manipulate arrays in JavaScript, highlighting the language's functional programming capabilities.
- Object Construction: Creating product objects for items in the cart illustrates the use of object literals to store related data, making it easier to manage product attributes.
- Discount Application: Implementing a simple discount system provided insight into conditional logic and basic data structures for storing values (e.g., discount codes).
- Console Logging for Output: The use of console.log for displaying cart summaries and totals reinforces the importance of outputting results for user feedback in any application.

Overall, this code serves as a practical example of how to implement a shopping cart system in JavaScript, combining essential programming techniques and concepts.

----------------------------------------*----------------------*