Shop Tracker System

Shop Tracker Program Documentation

Purpose of the Application

The Shop Tracker Program is designed to assist shop owners in managing and tracking customer purchases. The application allows users to display available products, add items to a shopping cart, and calculate the total cost of purchased items. It also provides a detailed summary of the cart contents, including the number of items bought and the total amount spent.

Features of the Application

- **Operation:**
    - Allow users to interact with the program as long as they wish. Users should be able to kill the application whenever they feel like.
    - A condition that will allow the program to kill based on a certain condition.
- **Well Print Message:**
    - Users are informed with a brief and precise message.
    - Users are prompted upon inputting and whenever they input wrong values.
- **Display Product List**:
    - Displays a list of products along with their prices in a tabular format.
    - Users can choose a product with ease and comfort.
- **Add Items to Cart**:
    - The program enables users to add items to their shopping cart by entering a correct product key.
    - Updates the cart with the selected product and its corresponding price.
- **View Cart Details**:

        - Provides a detailed summary of the items in the cart, including the product names, prices, total number of items, and the total cost.
- **Error Handling**:
        - Handles invalid product keys and input mismatches gracefully, providing user-friendly error messages.

Code Documentation

```
package projects;

import java.util.ArrayList;
import java.util.InputMismatchException;
import java.util.Scanner;

public class ShopTrackerProgram {

    /**
     * Prints the list of products along with their prices.
     *
     * This method takes two arrays: one containing product names
and the other containing
     * corresponding product prices. It prints out a tabular
representation of the product
     * key, product name, and product price.
     *
     * @param product An array of product names.
     * @param price   An array of product prices.
     */
    static void printProduct(String[] product, double[] price) {
        // Print table header
        System.out.println("Product Key \t Product Name \t Product
Price");

        // Loop through each product and its corresponding price
        for (int i = 0; i < product.length; i++) {
            // Print product key, product name, and product price
            System.out.println(i + "\t\t\t\t" + product[i] +
"\t\t\t\t" + price[i]);
        }
    }
```

```
    /**
     * Prints the details of the shopping cart.
     *
     * This method takes two ArrayLists: one containing product
names and the other containing
     * corresponding product prices, along with the total amount of
the cart. It prints out a
     * tabular representation of the product name and price, as
well as the total number of items
     * bought and the total item cost.
     *
     * @param productName  An ArrayList of product names.
     * @param productPrice An ArrayList of product prices.
     * @param totalAmount  The total cost of the items in the cart.
     */
    static void printCartDetail(ArrayList<String> productName,
ArrayList<Double> productPrice, double totalAmount) {
        System.out.println(); // create line
        System.out.println("Product Name \t Product Price");

        // Loop through each product and its corresponding price
        for (int i = 0; i < productName.size(); i++) {
            // Print product name and product price
            System.out.println(productName.get(i) + "\t\t\t\t" +
productPrice.get(i));
        }

        System.out.println(); // create line
        System.out.println("---------------------------------");
        System.out.println("Number of items bought: " +
productName.size());
        System.out.println("Total item cost: GMD" + totalAmount);
    }

      /**
     * Formats a given currency value into a string with a
specified format.
     *
     * This method takes a double value representing a currency
amount and formats it according
     * to the US locale currency format. The resulting string
representation of the currency
```

```
     * amount will have a "GMD" prefix instead of the default "$"
symbol used in the US locale.
     *
     * @param currency The currency amount to be formatted.
     * @return A string representing the formatted currency amount
with "GMD" as the prefix.
     */
    static String currencyFormatter(double currency){
        NumberFormat currencyFormatter =
NumberFormat.getCurrencyInstance(Locale.US);
        String formattedPrice = currencyFormatter.format((int)
currency);
        // Replace "$" with "GMD"
        formattedPrice = formattedPrice.replace("$", "GMD");
        return formattedPrice;
    }

    public static void main(String[] args) {
        System.out.println("************* Shop Tracker Program
*************");

        // Scanner object
        Scanner in = new Scanner(System.in);

        // Global variables
        String[] productList = {"banana", "bread", "milk", "sugar",
"apple", "egg"};
        double[] productPrice = {50, 10, 75, 17, 15, 15};
        ArrayList<String> cartProductName = new ArrayList<>();
        ArrayList<Double> cartProductPrice = new ArrayList<>();
        int totalNumberOfItemSold = 0;
        double totalAmount = 0;

        // While loop for continuous operation
        while (true) {
            printProduct(productList, productPrice); // Print
product list to the user

            System.out.println(); // Create empty line

            try {
```

```
                System.out.print("Enter product key: "); // Enter
input - product key
                int index = in.nextInt(); // Store product key into
index
                // Print product name and price
                System.out.println("You are buying '" +
productList[index] + "' it costs: 'GMD" + productPrice[index] +
"'");
                totalAmount += productPrice[index];

                cartProductName.add(productList[index]);
                cartProductPrice.add(productPrice[index]);
                System.out.println("Items in the cart: " +
cartProductName.size());
                System.out.println("Total price: GMD" +
totalAmount);
                System.out.println();

                System.out.print("Type Q to quit or any other key
to continue: ");
                String exit = in.next();

                if (exit.equalsIgnoreCase("Q")) // Break the
program when user types Q or q
                    break;

            } catch (ArrayIndexOutOfBoundsException e) {
                System.out.println("Product not found! Check the
key you typed.");
            } catch (InputMismatchException e) {
                System.out.println("Wrong input");
                in.nextLine(); // Clear the invalid input
            }

            System.out.println("The number of items sold: " +
totalNumberOfItemSold);
        } // End while

        System.out.println(); // Create a line
        // Print cart details
        printCartDetail(cartProductName, cartProductPrice,
totalAmount);
```

```
    }
}
```

Methods:

1. **printProduct:** This method takes two arrays: one containing product names and the other containing corresponding product prices. It prints out a tabular representation of the product key, product name, and product price.
2. **printCartDetail:** This method takes two ArrayLists: one containing product names and the other containing corresponding product prices, along with the total amount of the cart. It prints out a tabular representation of the product name and price, as well as the total number of items bought and the total item cost.
3. **currencyFormatter:** This method takes a double value representing a currency amount and formats it according to the US locale currency format. The resulting string representation of the currency amount will have a "GMD" prefix instead of the default "$" symbol used in the US locale.