Café Cashier Stack System

PRESENTED BY:

MAXIM MAMDOUH
ABDALLAH SALAH
MARIN MEDHAT
YOUANNA WAGEH
SEIF EL DEN MAMDOUH
SHEHAB EL DEN AHMED
SHERIF BASEM

PRESENTED TO:

DR. TAMER ABDEL LATIF

Table of Contents

Table of Figures

Chapter 1: Phase One

1.1. Introduction

Cashier systems play a pivotal role in the smooth functioning of businesses across various industries, serving as the backbone of transactional operations. These systems encompass a diverse array of hardware and software [1].

Our system utilizes a stack data structure to manage orders, streamline transactions, and enhance customer satisfaction. Designed with the needs of modern cafes in mind, our solution optimizes workflow and minimizes errors [2].

Figure 1: Cashier system [3]

1.2. C++ code:

```
#include <iostream>
#include <string>
#include <limits>

using namespace std;

const int MAX_ORDERS = 15;

void clearConsole()
{
   system("cls");
}
void waitForEnter()
{
   cout << "\n\nPress Enter to continue...\n";
   cout.flush();
   cin.ignore(numeric_limits<streamsize>::max(), '\n');
   cin.get();
   clearConsole();
}
class Stack
{
private:
   string orders[MAX_ORDERS];
   float prices[MAX_ORDERS];
   int numOrders;

public:
   float totalPrice;
   Stack()
   {
      numOrders = 0;
      totalPrice = 0.0;

      prices[0] = 7.5;
      prices[1] = 15;
      prices[2] = 20;
      prices[3] = 20;
      prices[4] = 27;
      prices[5] = 15;
      prices[6] = 30;
      prices[7] = 20;
```

```
        prices[8] = 15;
        prices[9] = 15;
        prices[10] = 20;
        prices[11] = 10;
    }

    void push(string order, float price)
    {
        clearConsole();
        if (isFull())
        {
            cout << "Maximum orders reached. Cannot place more orders.\n";
            return;
        }
        orders[numOrders] = order;
        prices[numOrders] = price;
        numOrders++;
        totalPrice += price;
        cout << "Order placed successfully: " << order << " (Price: " << price << " EGP)\n";
    }

    string popLast()
    {
        string c;
        string order ;
        float price ;


        if (isEmpty())
        {
            cout << "No orders to process.\n";
            waitForEnter();
            return "";
        }
        order=orders[numOrders-1];
        cout << "Are you sure you want to delete the last order ? (last order is : " << order <<
" )\nEnter Y/y for confirming or anything else to get back: ";
        cin >> c;
        if (c == "Y" || c == "y")
        {
            numOrders--;
            order = orders[numOrders];
            price = prices[numOrders];
            totalPrice -= price;
            cout << "Deleted last order: " << order << " (Price: " << price << " EGP)\n";
            waitForEnter();
        }
        else
        {
            cout << "nothing effect the orders";
            waitForEnter();
        }
        clearConsole();
        return order;
    }

    void popAll()
    {
        if (isEmpty())
        {
            cout << "No orders to process.\n";
            waitForEnter();
            return;
        }
        string c;
        string order = orders[numOrders-1];
        float price = prices[numOrders-1];

        cout << "Are you sure you want to delete all the orders ? there is no return for this
choice\nEnter Y/y for confirming or anything else to get back: ";
        cin >> c;
        if (c == "Y" || c == "y")
        {
            numOrders = 0;
            order = orders[numOrders];
            price = prices[numOrders];
            totalPrice = 0;
            cout << "All the orders have been deleted successfully.\n";
            waitForEnter();
        }
        else
        {
            cout << "nothing effect the orders";
            waitForEnter();
        }
        return;
    }

    bool isEmpty()
    {
        return (numOrders == 0);
    }

    bool isFull()
    {
        return (numOrders == MAX_ORDERS);
    }

    void display()
    {
        if (isEmpty())
        {
            cout << "No orders to display.\n";

            return;
        }
        cout << "Orders in the stack:\n";
        for (int i = numOrders - 1; i >= 0; --i)
        {
            cout << orders[i] << endl;
        }
        cout << "Total Price: " << totalPrice << " EGP";
    }
};

int main()
{
    Stack stack;
    int choice;

    do
    {
        cout << "\nCafe Cashier System Menu:                    "
             << "Total price for the current order is : " << stack.totalPrice << " EGP\n";
        cout << "1. Place Order\n";
        cout << "2. Display Orders\n";
        cout << "3. Reset Orders\n";
        cout << "4. Delete Last Order\n";
        cout << "5. Exit\n";
        cout << "Enter your choice: ";

        string input;
        cin >> input;

        try
        {
            choice = stoi(input);
            switch (choice)
            {
            case 1:
            {
                clearConsole();
                while (true)
                {
                    cout << endl
                         << "Menu:                                    Total price for the current order is
: " << stack.totalPrice << " EGP\n";
                    cout << "1. Tea (7.5 )\n";
                    cout << "2. Coffee (15 )\n";
                    cout << "3. Nescafe (20 )\n";
                    cout << "4. Sahlab (20 )\n";
                    cout << "5. Sahlab nuts (27 )\n";
                    cout << "6. Anise (15 )\n";
                    cout << "7. Mango Drink (30 )\n";
                    cout << "8. Grape Drink (20 )\n";
                    cout << "9. Tea and Milk (15 )\n";
                    cout << "10. Pepsi - Cola  Drink (15 )\n";
                    cout << "11. Shesha Fruits (20 )\n";
                    cout << "12. Shesha (10 )\n";
                    cout << "0. Done placing orders\n";
                    cout << "Enter product number to add to order (0 to finish): ";

                    string inputp;
                    cin >> inputp;
                    int productChoice = stoi(inputp);

                    if (productChoice == 0)
                    {
                        clearConsole();
                        break;
                    }
                    if (productChoice < 1 || productChoice > 12)
```

```
{
    clearConsole();
    cout << "Invalid product choice.\n";

    continue;
}

string product;
switch (productChoice)
{
case 1:
    product = "Tea";
    break;
case 2:
    product = "Coffee";
    break;
case 3:
    product = "Nescafe";
    break;
case 4:
    product = "Sahlab";
    break;
case 5:
    product = "Sahlab nuts";
    break;
case 6:
    product = "Anise";
    break;
case 7:
    product = "Mango Drink";
    break;
case 8:
    product = "Grape Drink";
    break;
case 9:
    product = "Tea and Milk";
    break;
case 10:
    product = "Pepsi - Cola  Drink";
    break;
case 11:
    product = "Shesha Fruits";
    break;
```

```
case 12:
    product = "Shesha";
    break;
}

float price;
switch (productChoice)
{
case 1:
    price = 7.5;
    break;
case 2:
    price = 15.0;
    break;
case 3:
    price = 20.0;
    break;
case 4:
    price = 20.0;
    break;
case 5:
    price = 27.0;
    break;
case 6:
    price = 15.0;
    break;
case 7:
    price = 30.0;
    break;
case 8:
    price = 20.0;
    break;
case 9:
    price = 15.0;
    break;
case 10:
    price = 15.0;
    break;
case 11:
    price = 20.0;
    break;
case 12:
    price = 10.0;
```

```
                break;
            }

            stack.push(product, price);
        }
        break;
    }
    case 2:
    {
        clearConsole();
        stack.display();
        waitForEnter();

        break;
    }
    case 3:
    {
        clearConsole();
        stack.popAll();

        break;
    }
    case 4:
    {
        clearConsole();
        stack.popLast();
        break;
    }
    case 5:
    {
        clearConsole();
        cout << "Exiting...\n";
        break;
    }
    default:
        cout << "Invalid choice. Please enter a number between 1 and 5.\n";
    }
}
catch (const invalid_argument &e)
{
    clearConsole();
    cout << "Invalid choice. Please enter an integer.\n";
    choice = -1;
```

```
        }
    } while (choice != 5);

    return 0;
}
```

1.3. Pseudocode:

this Program is Built for Running a Coffee-Shop Cashier System.

```
Define constant MAX_ORDERS = 15

Define function clearConsole():
    Clear the console screen using system command 'cls'

Define function waitForEnter():
    Display message "Press Enter to continue..."
    Flush the output buffer
    Ignore remaining characters in the input buffer until newline character
    Wait for user to press Enter
    Call clearConsole()

Define class Stack:
    Define private attributes:
        - orders: array of strings to store orders
        - prices: array of floats to store prices
        - numOrders: integer to track the number of orders
    Define public attribute:
        - totalPrice: float to store total price

    Define constructor Stack():
        Initialize numOrders to 0
        Initialize totalPrice to 0.0
        Initialize prices array with predefined prices
```

```
Define method push(order: string, price: float):
    Call clearConsole()
    If numOrders is equal to MAX_ORDERS:
        Display message "Maximum orders reached. Cannot place more orders."
        Return
    Store order and price in respective arrays at index numOrders
    Increment numOrders by 1
    Add price to totalPrice
    Display message " indicating successful order placement ."

Define method popLast():
    Declare variables c, order, and price
    Set order to the last order in orders array
    Set price to the price corresponding to the last order
    If numOrders is 0:
        Display message "No orders to process."
        Call waitForEnter()
        Return an empty string
    Prompt user if they want to delete the last order
    If user confirms deletion:
        Decrement numOrders by 1
        Update order and price to the new last order and its price
        Subtract price from totalPrice
        Display message indicating successful deletion of last order
        Call waitForEnter()
    Else:
        Display message "Nothing affects the orders"
        Call waitForEnter()
    Call clearConsole()
    Return the deleted order

Define method popAll():
    If numOrders is 0:
        Display message "No orders to process."
        Call waitForEnter()
        Return
    Declare variables c, order, and price
    Prompt user if they want to delete all orders
    If user confirms deletion:
        Set numOrders to 0
        Reset order and price
        Set totalPrice to 0
        Display message "All the orders have been deleted successfully."
        Call waitForEnter()
    Else:
        Display message "Nothing affects the orders"
        Call waitForEnter()

Define method isEmpty():
    Return true if numOrders is 0, else false

Define method isFull():
    Return true if numOrders is equal to MAX_ORDERS, else false

Define method display():
        If numOrders is 0:
            Display message "No orders to display."
            Return
        Display orders in reverse order along with totalPrice

Define function main():
    Create an instance of Stack called stack
    Declare integer choice
    Start a do-while loop:
        Display menu options
        Get user input for choice
        Try to convert input to integer
        Switch on choice:
            CASE 1:
                    Clear console
                    WHILE true
                        Display menu of products and prices
                        Prompt user to select product or finish order
                        Read user input for product choice
                        Convert product choice to integer

                        IF product choice is 0 THEN
                            Clear console
                            BREAK loop
                        END IF

                        IF product choice is not between 1 and 12 THEN
                            Clear console
            Display "Invalid product choice" message
            CONTINUE loop
         END IF

         Get product name and price based on product choice
         Add product to order stack
      END WHILE
      BREAK
   CASE 2:
      Clear console
      Display current orders and total price
      Wait for user to press Enter
      BREAK
   CASE 3:
      Clear console
      Remove all orders from stack
      BREAK
   CASE 4:
      Clear console
      Remove last order from stack
      BREAK
   CASE 5:
      Clear console
      Display "Exiting..." message
      BREAK
   DEFAULT:
      Display "Invalid choice. Please enter a number between 1 and 5." message
END SWITCH
```

```
        CATCH invalid_argument exception
            Clear console
            Display "Invalid choice. Please enter an integer." message
            Set choice to -1
        END TRY

    WHILE choice is not equal to 5

END
```

1.4. Flowchart:

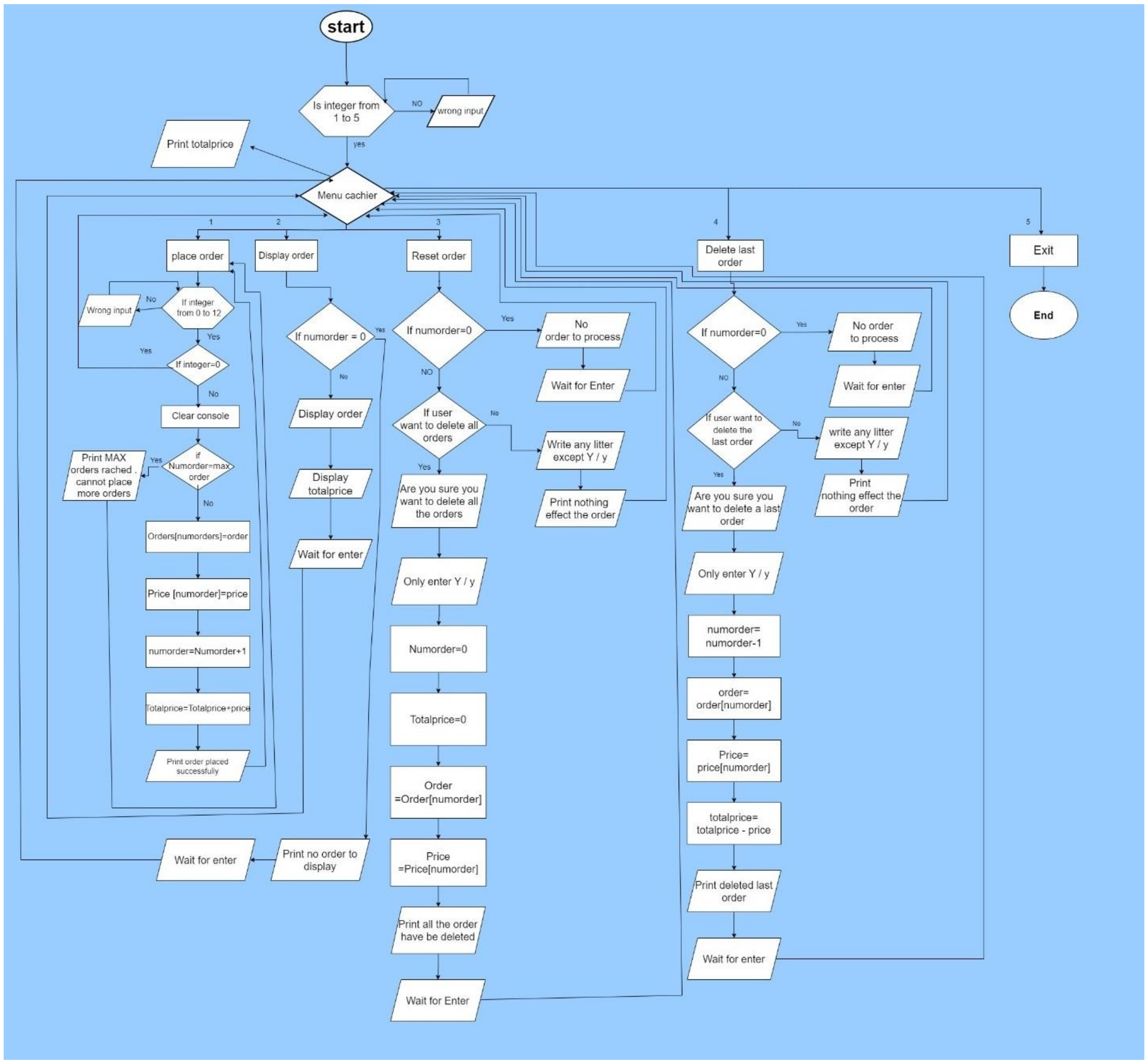

Figure 2: Flowchart [4]

References

1. https://www.investopedia.com/terms/p/point-of-sale.asp, March 16, 2023, 7:52PM
2. Self-written, March 16, 2023, 8:09PM
3. https://www.istockphoto.com/photos/pos-system, March 16, 2023, 8:21PM
4. Self-made, March 17, 7:12PM