**FINAL PROJECT**

**CAFÉ MANAGEMENT SYSTEM**

**OOP-LAB**



**Submitted to:**

**SIR UMAR KHALIL**

**Submitted by:**

**REHAN ALI (692)**

**PROGRAM:**

**BSCS SECOND SEMESTER**

**SESSION:**

**2023-2027**

**DEPARTMENT OF COMPUTER SCIENCES**

**SUPERIOR UNIVERSITY (GOLD CAMPUS)**

**RAIWIND ROAD LAHORE**

**CODE:**

#include <iostream>

#include <iomanip>

#include <string>

#include <map>

using namespace std;

class MenuItem {

public:

MenuItem() : name(""), price(0.0), quantity(0) {}

MenuItem(const string& name, double price, int quantity) : name(name), price(price), quantity(quantity) {}

string getName() const { return name; }

double getPrice() const { return price; }

int getQuantity() const { return quantity; }

void useItem(int amount) { quantity -= amount; }

private:

string name;

double price;

int quantity;

};

class Category {

public:

Category() : name(""), size(0) {}

Category(const string& name) : name(name), size(0) {}

void addItem(const MenuItem& item) {

if (size < maxItems) {

items[size++] = item;

}

}

void displayCategory() const {

cout << "Category: " << name << endl;

cout << setw(20) << left << "Item" << setw(10) << "Price" << setw(10) << "Quantity" << endl;

cout << setfill('-') << setw(40) << "" << setfill(' ') << endl;

for (size\_t i = 0; i < size; ++i) {

cout << setw(20) << left << items[i].getName() << setw(10) << items[i].getPrice() << setw(10) << items[i].getQuantity() << endl;

}

}

MenuItem& getItem(size\_t index) {

return items[index];

}

size\_t getSize() const {

return size;

}

private:

static const size\_t maxItems = 10;

string name;

MenuItem items[maxItems];

size\_t size;

};

class Menu {

public:

Menu() {

Category beverages("Beverages");

beverages.addItem(MenuItem("Coffee", 2.5, 50));

beverages.addItem(MenuItem("Tea", 2.0, 50));

Category snacks("Snacks");

snacks.addItem(MenuItem("Sandwich", 5.0, 30));

snacks.addItem(MenuItem("Cake", 3.5, 20));

Category fastFood("Fast Food");

fastFood.addItem(MenuItem("Burger", 7.0, 25));

fastFood.addItem(MenuItem("Pizza", 8.0, 15));

categories["Beverages"] = beverages;

categories["Snacks"] = snacks;

categories["Fast Food"] = fastFood;

}

void displayMenu() const {

for (const auto& category : categories) {

category.second.displayCategory();

cout << endl;

}

}

MenuItem& getItem(const string& category, size\_t index) {

return categories.at(category).getItem(index);

}

bool isValidCategory(const string& category) const {

return categories.find(category) != categories.end();

}

private:

map<string, Category> categories;

};

class Order {

public:

Order() : size(0) {}=

void addItem(const MenuItem& item) {

if (size < maxItems) {

items[size++] = item;

}

}

double calculateTotal() const {

double total = 0.0;

for (size\_t i = 0; i < size; ++i) {

total += items[i].getPrice();

}

return total;

}

void displayOrder() const {

cout << "Your Order:" << endl;

for (size\_t i = 0; i < size; ++i) {

cout << items[i].getName() << " - $" << items[i].getPrice() << endl;

}

cout << "Total: $" << fixed << setprecision(2) << calculateTotal() << endl;

}

size\_t getSize() const {

return size;

}

void clear() {

size = 0;

}

bool isEmpty() const {

return size == 0;

}

private:

static const size\_t maxItems = 10;

MenuItem items[maxItems];

size\_t size;

};

class Cafe {

public:

Cafe() : orderCount(0) {}

void start() {

int choice;

do {

displayOptions();

cin >> choice;

switch (choice) {

case 1:

menu.displayMenu();

break;

case 2:

takeOrder();

break;

case 3:

if (!currentOrder.isEmpty()) {

processPayment();

}

else {

cout << "No items in the current order." << endl;

}

break;

case 4:

viewOrders();

break;

case 5:

cout << "Exiting the cafe management system. Goodbye!" << endl;

break;

default:

cout << "Invalid choice. Please try again." << endl;

break;

}

} while (choice != 5);

}

private:

Menu menu;

Order currentOrder;

Order orders[10];

size\_t orderCount;

void displayOptions() const {

cout << "\nCafe Management System" << endl;

cout << "1. Display Menu" << endl;

cout << "2. Take Order" << endl;

cout << "3. Process Payment" << endl;

cout << "4. View Orders" << endl;

cout << "5. Exit" << endl;

cout << "Enter your choice: ";

}

void takeOrder() {

string category;

int choice;

do {

menu.displayMenu();

cout << "Enter the category of the item you want to order: ";

cin.ignore();

getline(cin, category);

if (!menu.isValidCategory(category)) {

cout << "Invalid category. Please try again." << endl;

continue;

}

do {

cout << "Enter the number of the item you want to order (0 to finish): ";

cin >> choice;

if (choice >= 1 && static\_cast<size\_t>(choice) <= menu.getItem(category, choice - 1).getQuantity()) {

currentOrder.addItem(menu.getItem(category, choice - 1));

menu.getItem(category, choice - 1).useItem(1);

cout << "Added " << menu.getItem(category, choice - 1).getName() << " to your order." << endl;

}

else if (choice != 0) {

cout << "Invalid choice. Please try again." << endl;

}

} while (choice != 0);

} while (!menu.isValidCategory(category));

}

void processPayment() {

double total = currentOrder.calculateTotal();

cout << "Total amount to pay: $" << fixed << setprecision(2) << total << endl;

currentOrder.displayOrder();

if (orderCount < 10) {

orders[orderCount++] = currentOrder;

}

else {

cout << "Maximum order limit reached." << endl;

}

currentOrder.clear();

cout << "Payment processed. Order cleared." << endl;

}

void viewOrders() const {

cout << "Previous Orders:" << endl;

for (size\_t i = 0; i < orderCount; ++i) {

cout << "Order " << i + 1 << ":" << endl;

orders[i].displayOrder();

}

}

};

int main() {

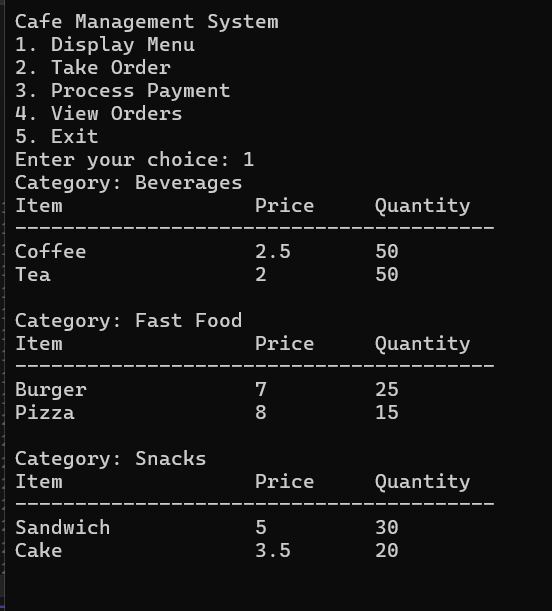
Cafe cafe;

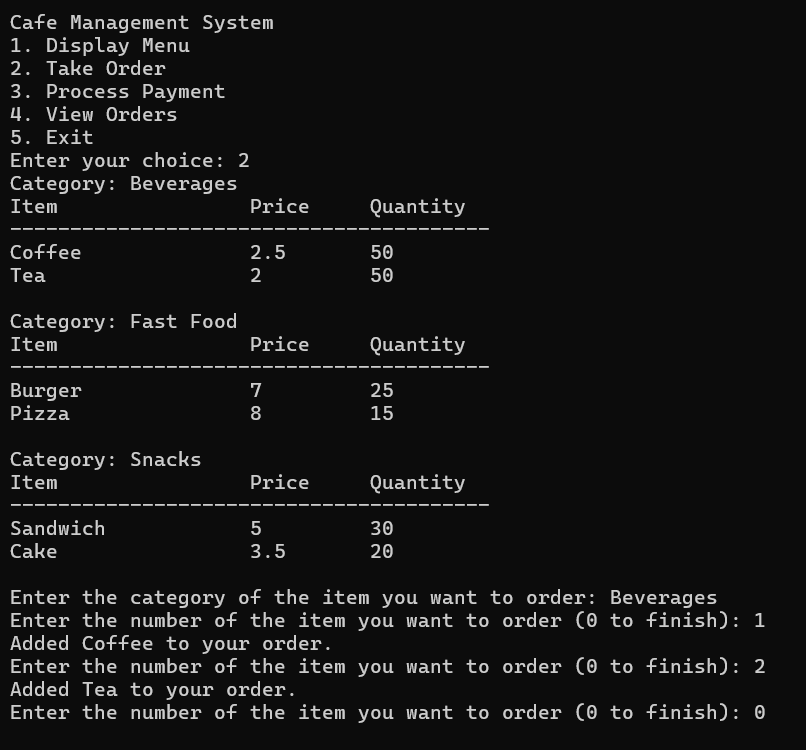
cafe.start();

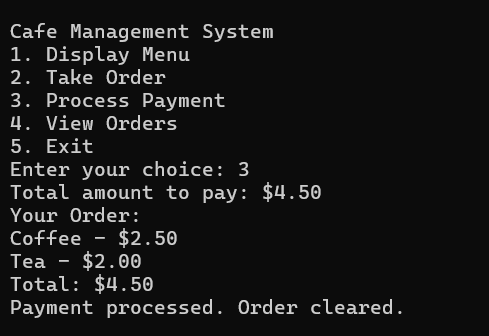
return 0;

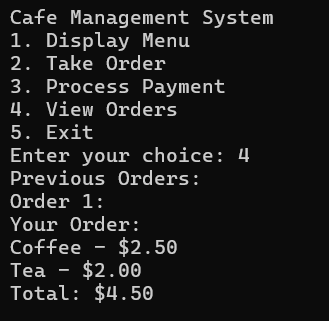
}

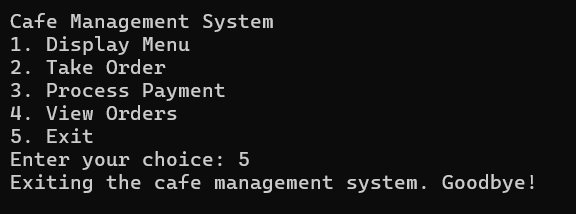
**OUTPUT:**

****

****

****

****

****