****

**PROJECT**

**Submitted By:**

**Uqba Gulzar**

**Zunaira Khatoon**

**Registration No:**

**2023-BSE-067**

**2023-BSE-074**

**Submitted to:**

**Dr. Sidra Ejaz**

**Course Title:**

**Object Oriented Programming**

#include <iostream>

#include <string>

using namespace std;

// Base Cake class

class Cake {

public:

virtual void processing() const = 0;

virtual void track\_order() const = 0;

void thanks() const {

cout << "Thank you for ordering" << endl;

}

virtual ~Cake() {}

};

class ChocolateCake : public Cake {

public:

void processing() const override {

cout << "Processing your order for Chocolate Cake.\n";

}

void track\_order() const override {

cout << "Your order for Chocolate Cake is tracked and will be delivered within an hour\n";

}

};

class VanillaCake : public Cake {

public:

void processing() const override {

cout << "Processing your order for Vanilla Cake.\n";

}

void track\_order() const override {

cout << "Your order for Vanilla Cake is tracked and will be delivered within an hour\n";

}

};

class FruitCake : public Cake {

public:

void processing() const override {

cout << "Processing your order for Fruit Cake.\n";

}

void track\_order() const override {

cout << "Your order for Fruit Cake is tracked and will be delivered within an hour\n";

}

};

class Customer {

private:

string name;

string email;

string address;

string phone;

public:

Customer(string name, string email, string address, string phone)

: name(name), email(email), address(address), phone(phone) {}

string getName() const {

return name;

}

string getEmail() const {

return email;

}

string getAddress() const {

return address;

}

string getPhone() const {

return phone;

}

};

class Order {

private:

Customer customer;

Cake\* cake;

public:

Order(Customer customer, Cake\* cake)

: customer(customer), cake(cake) {}

void processOrder(){

cout << "Processing order for " << customer.getName() << " (" << customer.getPhone() << ") at " << customer.getAddress() << ".\n";

cake->processing();

cake->track\_order();

cake->thanks();

}

};

class Bakery {

private:

Order\* orders[10];

int orderCount = 0;

public:

~Bakery() {

for (int i = 0; i < orderCount; ++i) {

delete orders[i];

}

}

void addOrder(Order order) {

if (orderCount < 10) {

orders[orderCount] = new Order(order);

orderCount++;

}

else {

cout << "Cannot add more orders, bakery is full.\n";

}

}

void processOrders() const {

for (int i = 0; i < orderCount; ++i) {

orders[i]->processOrder();

}

}

};

int main() {

Bakery bakery;

ChocolateCake chocoCake;

VanillaCake vanillaCake;

FruitCake fruitCake;

string name, email, address, phone;

cout << "Enter customer name: ";

getline(cin, name);

cout << "Enter customer email: ";

getline(cin, email);

cout << "Enter customer address: ";

getline(cin, address);

cout << "Enter customer phone: ";

getline(cin, phone);

Customer customer(name, email, address, phone);

while (true) {

Cake\* selectedCake = nullptr;

while (true) {

string cakeType;

cout << "Enter cake type (chocolate, vanilla, fruit): ";

getline(cin, cakeType);

if (cakeType == "chocolate") {

selectedCake = &chocoCake;

break;

}

else if (cakeType == "vanilla") {

selectedCake = &vanillaCake;

break;

}

else if (cakeType == "fruit") {

selectedCake = &fruitCake;

break;

}

else {

cout << "Invalid cake type entered. Please re-enter.\n";

}

}

Order order(customer, selectedCake);

bakery.addOrder(order);

char anotherOrder;

cout << "Do you want to add another order? (y/n): ";

cin >> anotherOrder;

cin.ignore();

if (anotherOrder == 'n' || anotherOrder == 'N') {

break;

}

}

bakery.processOrders();

return 0;

}