#include <iostream>

#include <string>

using namespace std;

// Abstract base class

class CustomerData {

protected:

int customerId;

string name;

string email;

string address;

string phoneNumber;

public:

// Constructor

CustomerData() : customerId(0), name(""), email(""), address(""), phoneNumber("") {}

// Initialize data

void setData(int id, string n, string e, string a, string p) {

customerId = id;

name = n;

email = e;

address = a;

phoneNumber = p;

}

// Pure virtual functions

virtual void displayData() = 0;

virtual void updateData() = 0;

// Getters

int getId() { return customerId; }

string getName() { return name; }

};

// Customer class that inherits from CustomerData

class Customer : public CustomerData {

private:

double accountBalance;

bool premiumMember;

bool isActive; // To track if the customer is deleted or not

public:

// Constructor

Customer() : accountBalance(0.0), premiumMember(false), isActive(true) {}

// Initialize customer specific data

void initializeCustomer(double balance, bool premium) {

accountBalance = balance;

premiumMember = premium;

}

// getData function to input customer data from user

void getData() {

cout << "Enter Customer ID: ";

cin >> customerId;

cin.ignore(); // Clear the newline character

cout << "Enter Name: ";

getline(cin, name);

cout << "Enter Email: ";

getline(cin, email);

cout << "Enter Address: ";

getline(cin, address);

cout << "Enter Phone Number: ";

getline(cin, phoneNumber);

cout << "Enter Account Balance: ";

cin >> accountBalance;

cout << "Premium Member (1 for Yes, 0 for No): ";

int temp;

cin >> temp;

premiumMember = (temp == 1);

isActive = true;

}

// showData function to display customer data

void showData() {

if (!isActive) {

cout << "Customer record has been deleted." << endl;

return;

}

cout << "Customer ID: " << customerId << endl;

cout << "Name: " << name << endl;

cout << "Email: " << email << endl;

cout << "Phone: " << phoneNumber << endl;

cout << "Balance: $" << accountBalance << endl;

cout << "Premium Status: " << (premiumMember ? "Premium" : "Regular") << endl;

}

// Implementing abstract methods

void displayData() override {

showData(); // Reuse the showData function

}

void updateData() override {

edit(); // Reuse the edit function

}

// order function to place an order

void order() {

if (!isActive) {

cout << "Cannot place order: Customer record has been deleted." << endl;

return;

}

double orderAmount;

cout << "Enter order amount: $";

cin >> orderAmount;

if (accountBalance >= orderAmount) {

accountBalance -= orderAmount;

cout << "Order placed successfully for $" << orderAmount << endl;

cout << "New balance: $" << accountBalance << endl;

}

else {

cout << "Insufficient balance to place order" << endl;

}

}

// choice function to present menu options

int choice() {

if (!isActive) {

cout << "Customer record has been deleted." << endl;

return 0;

}

int option;

cout << "\nCustomer Menu for " << name << ":" << endl;

cout << "1. Show Customer Data" << endl;

cout << "2. Edit Customer Data" << endl;

cout << "3. Place an Order" << endl;

cout << "4. Make a Payment" << endl;

cout << "5. Upgrade Account" << endl;

cout << "6. Delete Customer" << endl;

cout << "0. Exit" << endl;

cout << "Enter your choice: ";

cin >> option;

return option;

}

// edit function to edit customer data

void edit() {

if (!isActive) {

cout << "Cannot edit: Customer record has been deleted." << endl;

return;

}

cout << "Editing Customer ID: " << customerId << endl;

cin.ignore(); // Clear the newline character

cout << "Enter new Name (current: " << name << "): ";

getline(cin, name);

cout << "Enter new Email (current: " << email << "): ";

getline(cin, email);

cout << "Enter new Address (current: " << address << "): ";

getline(cin, address);

cout << "Enter new Phone Number (current: " << phoneNumber << "): ";

getline(cin, phoneNumber);

cout << "Customer data updated successfully!" << endl;

}

// allData function to show complete customer data

void allData() {

if (!isActive) {

cout << "Customer record has been deleted." << endl;

return;

}

cout << "===== COMPLETE CUSTOMER DETAILS =====" << endl;

cout << "Customer ID: " << customerId << endl;

cout << "Name: " << name << endl;

cout << "Email: " << email << endl;

cout << "Address: " << address << endl;

cout << "Phone: " << phoneNumber << endl;

cout << "Account Balance: $" << accountBalance << endl;

cout << "Premium Member: " << (premiumMember ? "Yes" : "No") << endl;

cout << "Account Status: " << (isActive ? "Active" : "Deleted") << endl;

cout << "====================================" << endl;

}

// delete function to mark customer as deleted

void deleteCustomer() {

if (!isActive) {

cout << "Customer record already deleted." << endl;

return;

}

char confirm;

cout << "Are you sure you want to delete customer " << name << "? (Y/N): ";

cin >> confirm;

if (confirm == 'Y' || confirm == 'y') {

isActive = false;

cout << "Customer record deleted successfully." << endl;

}

else {

cout << "Deletion cancelled." << endl;

}

}

// Additional customer functions

void makePayment(double amount) {

if (!isActive) {

cout << "Cannot process payment: Customer record has been deleted." << endl;

return;

}

accountBalance += amount;

cout << "Payment of $" << amount << " received" << endl;

cout << "New balance: $" << accountBalance << endl;

}

void upgradeAccount() {

if (!isActive) {

cout << "Cannot upgrade: Customer record has been deleted." << endl;

return;

}

if (!premiumMember) {

char confirm;

cout << "Upgrade to Premium account for $100? (Y/N): ";

cin >> confirm;

if (confirm == 'Y' || confirm == 'y') {

if (accountBalance >= 100) {

accountBalance -= 100;

premiumMember = true;

cout << "Account upgraded to Premium!" << endl;

cout << "New balance: " << accountBalance << endl;

}

else {

cout << "Insufficient balance to upgrade. You need " << (100 - accountBalance) << " more." << endl;

}

}

else {

cout << "Upgrade cancelled." << endl;

}

}

else {

cout << "Already a Premium member" << endl;

}

}

// Check if customer is active

bool isActiveCustomer() {

return isActive;

}

};

// Admin class with similar structure to Customer class

class Admin {

private:

int adminId;

string adminName;

string email;

string department;

int accessLevel;

string password;

bool isActive;

public:

// Constructor

Admin() : adminId(0), adminName(""), email(""), department(""), accessLevel(0), password(""), isActive(true) {}

// getData function to input admin data

void getData() {

cout << "Enter Admin ID: ";

cin >> adminId;

cin.ignore(); // Clear the newline character

cout << "Enter Admin Name: ";

getline(cin, adminName);

cout << "Enter Email: ";

getline(cin, email);

cout << "Enter Department: ";

getline(cin, department);

cout << "Enter Access Level (1-5): ";

cin >> accessLevel;

cin.ignore(); // Clear the newline character

cout << "Create Password: ";

getline(cin, password);

isActive = true;

}

// showData function to display admin data

void showData() {

if (!isActive) {

cout << "Admin record has been deleted." << endl;

return;

}

cout << "Admin ID: " << adminId << endl;

cout << "Name: " << adminName << endl;

cout << "Email: " << email << endl;

cout << "Department: " << department << endl;

cout << "Access Level: " << accessLevel << endl;

}

// choice function to present menu options

int choice() {

if (!isActive) {

cout << "Admin record has been deleted." << endl;

return 0;

}

int option;

cout << "\nAdmin Menu for " << adminName << ":" << endl;

cout << "1. Show Admin Profile" << endl;

cout << "2. Edit Admin Profile" << endl;

cout << "3. Manage Customers" << endl;

cout << "4. Generate Reports" << endl;

cout << "5. System Settings" << endl;

cout << "6. Delete Admin Account" << endl;

cout << "0. Exit" << endl;

cout << "Enter your choice: ";

cin >> option;

return option;

}

// edit function to edit admin data

void edit() {

if (!isActive) {

cout << "Cannot edit: Admin record has been deleted." << endl;

return;

}

cout << "Editing Admin ID: " << adminId << endl;

cin.ignore(); // Clear the newline character

cout << "Enter new Name (current: " << adminName << "): ";

getline(cin, adminName);

cout << "Enter new Email (current: " << email << "): ";

getline(cin, email);

cout << "Enter new Department (current: " << department << "): ";

getline(cin, department);

cout << "Enter new Access Level (current: " << accessLevel << "): ";

cin >> accessLevel;

char changePass;

cout << "Change password? (Y/N): ";

cin >> changePass;

if (changePass == 'Y' || changePass == 'y') {

cin.ignore(); // Clear the newline character

cout << "Enter new Password: ";

getline(cin, password);

}

cout << "Admin data updated successfully!" << endl;

}

// allData function to show complete admin data

void allData() {

if (!isActive) {

cout << "Admin record has been deleted." << endl;

return;

}

cout << "===== COMPLETE ADMIN DETAILS =====" << endl;

cout << "Admin ID: " << adminId << endl;

cout << "Name: " << adminName << endl;

cout << "Email: " << email << endl;

cout << "Department: " << department << endl;

cout << "Access Level: " << accessLevel << endl;

cout << "Account Status: " << (isActive ? "Active" : "Deleted") << endl;

cout << "====================================" << endl;

}

// delete function to mark admin as deleted

void deleteAdmin() {

if (!isActive) {

cout << "Admin record already deleted." << endl;

return;

}

string confirmPassword;

cout << "Enter password to confirm deletion: ";

cin.ignore(); // Clear the newline character

getline(cin, confirmPassword);

if (confirmPassword == password) {

char confirm;

cout << "Are you sure you want to delete admin account " << adminName << "? (Y/N): ";

cin >> confirm;

if (confirm == 'Y' || confirm == 'y') {

isActive = false;

cout << "Admin record deleted successfully." << endl;

}

else {

cout << "Deletion cancelled." << endl;

}

}

else {

cout << "Incorrect password. Deletion cancelled." << endl;

}

}

// order function equivalent for admin - process any order

void order() {

if (!isActive) {

cout << "Cannot process orders: Admin record has been deleted." << endl;

return;

}

int customerId;

cout << "Enter Customer ID to process order: ";

cin >> customerId;

cout << "Admin " << adminName << " is processing an order for customer ID " << customerId << endl;

// In a real application, this would look up the customer

// and process their order

double orderAmount;

cout << "Enter order amount: $";

cin >> orderAmount;

cout << "Order processed successfully for $" << orderAmount << endl;

}

// manageCustomer function to handle customer data

void manageCustomer() {

if (!isActive) {

cout << "Cannot manage customer: Admin record has been deleted." << endl;

return;

}

int customerId;

cout << "Enter Customer ID to manage: ";

cin >> customerId;

cout << "Admin " << adminName << " managing customer ID: " << customerId << endl;

int option;

do {

cout << "\nCustomer Management Menu:" << endl;

cout << "1. View Customer Data" << endl;

cout << "2. Edit Customer Data" << endl;

cout << "3. Process Customer Order" << endl;

cout << "4. Delete Customer" << endl;

cout << "0. Exit" << endl;

cout << "Enter your choice: ";

cin >> option;

switch (option) {

case 1:

cout << "Displaying data for customer ID " << customerId << endl;

// In a real application, this would look up and display the customer

break;

case 2:

cout << "Editing data for customer ID " << customerId << endl;

// In a real application, this would look up and edit the customer

break;

case 3:

cout << "Processing order for customer ID " << customerId << endl;

order();

break;

case 4:

cout << "Deleting customer ID " << customerId << endl;

// In a real application, this would look up and delete the customer

break;

case 0:

cout << "Exiting customer management..." << endl;

break;

default:

cout << "Invalid option." << endl;

}

} while (option != 0);

}

// generateReport function

void generateReport() {

if (!isActive) {

cout << "Cannot generate report: Admin record has been deleted." << endl;

return;

}

cout << "===== REPORT GENERATION MENU =====" << endl;

cout << "1. Sales Report" << endl;

cout << "2. Customer Activity Report" << endl;

cout << "3. Inventory Report" << endl;

cout << "4. Financial Summary" << endl;

cout << "0. Exit" << endl;

int option;

cout << "Select report type: ";

cin >> option;

switch (option) {

case 1:

cout << "Generating Sales Report..." << endl;

cout << "Sales Report generated by admin " << adminName << endl;

break;

case 2:

cout << "Generating Customer Activity Report..." << endl;

cout << "Customer Activity Report generated by admin " << adminName << endl;

break;

case 3:

cout << "Generating Inventory Report..." << endl;

cout << "Inventory Report generated by admin " << adminName << endl;

break;

case 4:

cout << "Generating Financial Summary..." << endl;

cout << "Financial Summary generated by admin " << adminName << endl;

break;

case 0:

cout << "Exiting report generation." << endl;

break;

default:

cout << "Invalid report type." << endl;

}

}

// updateSystemSettings function

void updateSystemSettings() {

if (!isActive) {

cout << "Cannot update settings: Admin record has been deleted." << endl;

return;

}

if (accessLevel < 3) {

cout << "Access denied: Insufficient privileges. Level 3+ required." << endl;

return;

}

cout << "===== SYSTEM SETTINGS MENU =====" << endl;

cout << "1. User Access Controls" << endl;

cout << "2. Database Management" << endl;

cout << "3. Security Settings" << endl;

cout << "4. Backup and Recovery" << endl;

cout << "0. Exit" << endl;

int option;

cout << "Select setting to update: ";

cin >> option;

switch (option) {

case 1:

cout << "Updating User Access Controls..." << endl;

cout << "User Access Controls updated by admin " << adminName << endl;

break;

case 2:

cout << "Updating Database Management..." << endl;

cout << "Database Management updated by admin " << adminName << endl;

break;

case 3:

cout << "Updating Security Settings..." << endl;

cout << "Security Settings updated by admin " << adminName << endl;

break;

case 4:

cout << "Updating Backup and Recovery..." << endl;

cout << "Backup and Recovery updated by admin " << adminName << endl;

break;

case 0:

cout << "Exiting system settings." << endl;

break;

default:

cout << "Invalid setting option." << endl;

}

}

// Check if admin is active

bool isActiveAdmin() {

return isActive;

}

};

// Main function to demonstrate the system

int main() {

// Create a customer

Customer customer1;

cout << "=== Creating a new customer ===" << endl;

customer1.getData();

// Create an admin

Admin admin1;

cout << "\n=== Creating a new admin ===" << endl;

admin1.getData();

int userType;

do {

cout << "\n===== MAIN MENU =====" << endl;

cout << "1. Customer Interface" << endl;

cout << "2. Admin Interface" << endl;

cout << "0. Exit System" << endl;

cout << "Enter your choice: ";

cin >> userType;

switch (userType) {

case 1: {

// Customer menu

int option;

do {

option = customer1.choice();

switch (option) {

case 1:

customer1.showData();

break;

case 2:

customer1.edit();

break;

case 3:

customer1.order();

break;

case 4:

double amount;

cout << "Enter payment amount: $";

cin >> amount;

customer1.makePayment(amount);

break;

case 5:

customer1.upgradeAccount();

break;

case 6:

customer1.deleteCustomer();

break;

case 0:

cout << "Exiting customer menu..." << endl;

break;

default:

cout << "Invalid option." << endl;

}

} while (option != 0);

break;

}

case 2: {

// Admin menu

int option;

do {

option = admin1.choice();

switch (option) {

case 1:

admin1.showData();

break;

case 2:

admin1.edit();

break;

case 3:

admin1.manageCustomer();

break;

case 4:

admin1.generateReport();

break;

case 5:

admin1.updateSystemSettings();

break;

case 6:

admin1.deleteAdmin();

break;

case 0:

cout << "Exiting admin menu..." << endl;

break;

default:

cout << "Invalid option." << endl;

}

} while (option != 0);

break;

}

case 0:

cout << "Exiting system. Goodbye!" << endl;

break;

default:

cout << "Invalid option." << endl;

}

} while (userType != 0);

return 0;

}