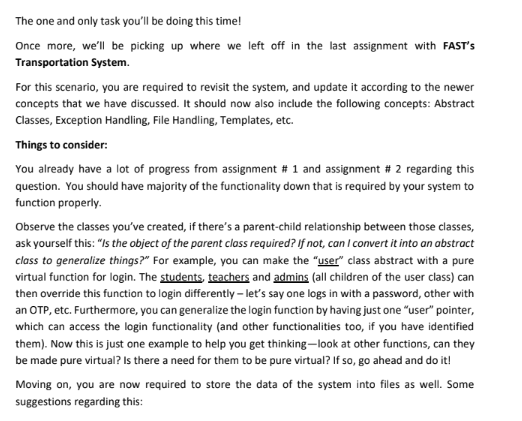
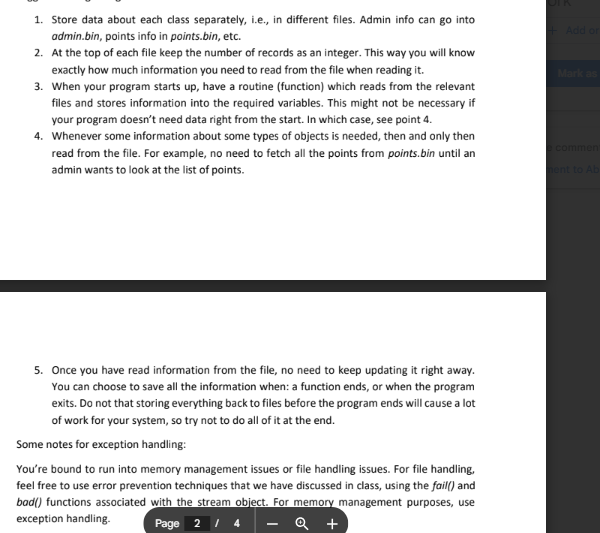
**OOP Assignment 3**

**Question 1:**





**Code:**

#include <iostream>

#include <fstream>

#include <cstring>

#include <exception>

using namespace std;

class AppException : public exception {

protected:

    string msg;

public:

    AppException(const string& m) : msg(m) {}

    const char\* what() const noexcept override {

        return msg.c\_str();

    }

};

class InvalidIDException : public AppException {

public:

    InvalidIDException() : AppException("Invalid ID entered. ID must be a positive number.") {}

};

class FileOpenException : public AppException {

public:

    FileOpenException(const string& filename) : AppException("Could not open file: " + filename) {}

};

class MaxLimitException : public AppException {

public:

    MaxLimitException() : AppException("Maximum user limit reached.") {}

};

class InvalidOptionException : public AppException {

public:

    InvalidOptionException() : AppException("Invalid menu option selected.") {}

};

class User {

protected:

    char name[50];

    int id;

public:

    User() { strcpy(name, ""); id = 0; }

    virtual void input() {

        cout << "Enter name: ";

        cin.ignore();

        cin.getline(name, 50);

        cout << "Enter ID (positive number): ";

        cin >> id;

        if(cin.fail() || id <= 0  ){

            throw InvalidIDException();

        }

    }

    virtual void display() const {

        cout << "Name: " << name << "\nID: " << id << endl;

    }

    virtual int getType() const = 0;

    virtual ~User() {}

};

// 🔹 Derived Classes

class Student : public User {

    char department[30];

public:

    void input() override {

        User::input(); cin.ignore();

        cout << "Enter department: ";

        cin.getline(department, 30);

    }

    void display() const override {

        cout << "[Student]\n"; User::display();

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

    }

    int getType() const override { return 1; }

};

class Teacher : public User {

    char subject[30];

public:

    void input() override {

        User::input(); cin.ignore();

        cout << "Enter subject: ";

        cin.getline(subject, 30);

    }

    void display() const override {

        cout << "[Teacher]\n"; User::display();

        cout << "Subject: " << subject << endl;

    }

    int getType() const override { return 2; }

};

class Staff : public User {

    char role[30];

public:

    void input() override {

        User::input(); cin.ignore();

        cout << "Enter role: ";

        cin.getline(role, 30);

    }

    void display() const override {

        cout << "[Staff]\n"; User::display();

        cout << "Role: " << role << endl;

    }

    int getType() const override { return 3; }

};

template <typename T>

class UserArray {

    T\* arr[10];

    int count;

public:

    UserArray() : count(0) {}

    void addUser() {

        if (count >= 10)

            throw MaxLimitException();

        arr[count] = new T();

        arr[count]->input();

        count++;

    }

    void displayAll() const {

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

            arr[i]->display();

            cout << "--------------------------\n";

        }

    }

    void saveAll(ofstream& fout) const {

        fout.write((char\*)&count, sizeof(count));

        for (int i = 0; i < count; ++i)

            saveUser(arr[i], fout);

    }

    ~UserArray() {

        for (int i = 0; i < count; ++i)

            delete arr[i];

    }

};

void saveUser(User\* user, ofstream& fout) {

    int type = user->getType();

    fout.write((char\*)&type, sizeof(type));

    if (type == 1)

        fout.write((char\*)user, sizeof(Student));

    else if (type == 2)

        fout.write((char\*)user, sizeof(Teacher));

    else if (type == 3)

        fout.write((char\*)user, sizeof(Staff));

}

void loadUsers(const string& filename) {

    ifstream fin(filename, ios::binary);

    if (!fin)

        throw FileOpenException(filename);

    int count;

    fin.read((char\*)&count, sizeof(count));

    cout << "Loading " << count << " users from " << filename << "...\n";

    while (fin.peek() != EOF) {

        int type;

        fin.read((char\*)&type, sizeof(type));

        if (type == 1) {

            Student s;

            fin.read((char\*)&s, sizeof(s));

            s.display();

        } else if (type == 2) {

            Teacher t;

            fin.read((char\*)&t, sizeof(t));

            t.display();

        } else if (type == 3) {

            Staff st;

            fin.read((char\*)&st, sizeof(st));

            st.display();

        } else {

            throw AppException("Unknown user type in file.");

        }

        cout << "--------------------------\n";

    }

    fin.close();

}

int main() {

    UserArray<Student> students;

    UserArray<Teacher> teachers;

    UserArray<Staff> staff;

    try {

        loadUsers("students.dat");

        loadUsers("teachers.dat");

        loadUsers("staff.dat");

    } catch (const AppException& e) {

        cout << "Warning: " << e.what() << endl;

    }

    int choice;

    do {

        cout << "\n--- Transport System ---\n";

        cout << "1. Add Student\n2. Add Teacher\n3. Add Staff\n4. Save to Files\n5. Load from Files\n6. Exit\nChoice: ";

        cin >> choice;

        try {

            if (cin.fail()) throw InvalidOptionException();

            switch (choice) {

            case 1:

                students.addUser(); break;

            case 2:

                teachers.addUser(); break;

            case 3:

                staff.addUser(); break;

            case 4: {

                ofstream studentFile("students.dat", ios::binary);

                if (!studentFile) throw FileOpenException("students.dat");

                students.saveAll(studentFile);

                studentFile.close();

                cout << "Data saved to students.dat\n";

                ofstream teacherFile("teachers.dat", ios::binary);

                if (!teacherFile) throw FileOpenException("teachers.dat");

                teachers.saveAll(teacherFile);

                teacherFile.close();

                cout << "Data saved to teachers.dat\n";

                ofstream staffFile("staff.dat", ios::binary);

                if (!staffFile) throw FileOpenException("staff.dat");

                staff.saveAll(staffFile);

                staffFile.close();

                cout << "Data saved to staff.dat\n";

                break;

            }

            case 5:

                loadUsers("students.dat"); break;

                loadUsers("teachers.dat"); break;

                loadUsers("staff.dat"); break;

            case 6:

                cout << "Exiting...\n"; break;

            default:

                throw InvalidOptionException();

            }

        } catch (const AppException& e) {

            cin.clear(); cin.ignore(1000, '\n');

            cout << e.what() << endl;

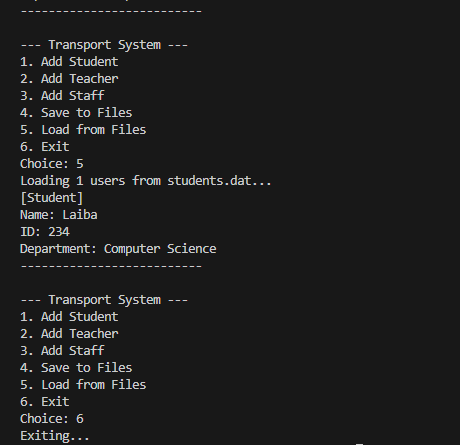
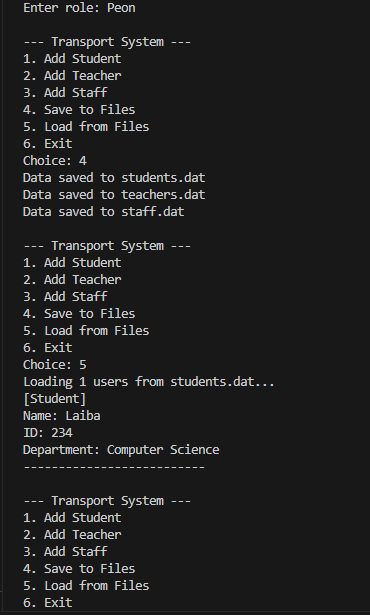
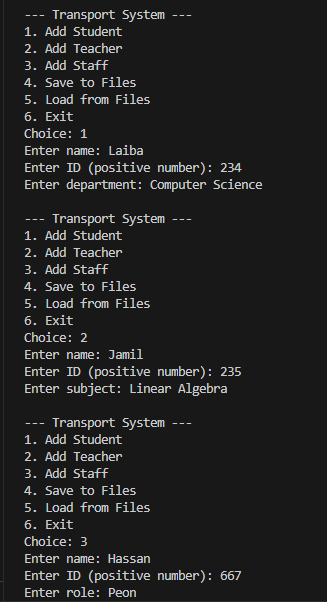
        }

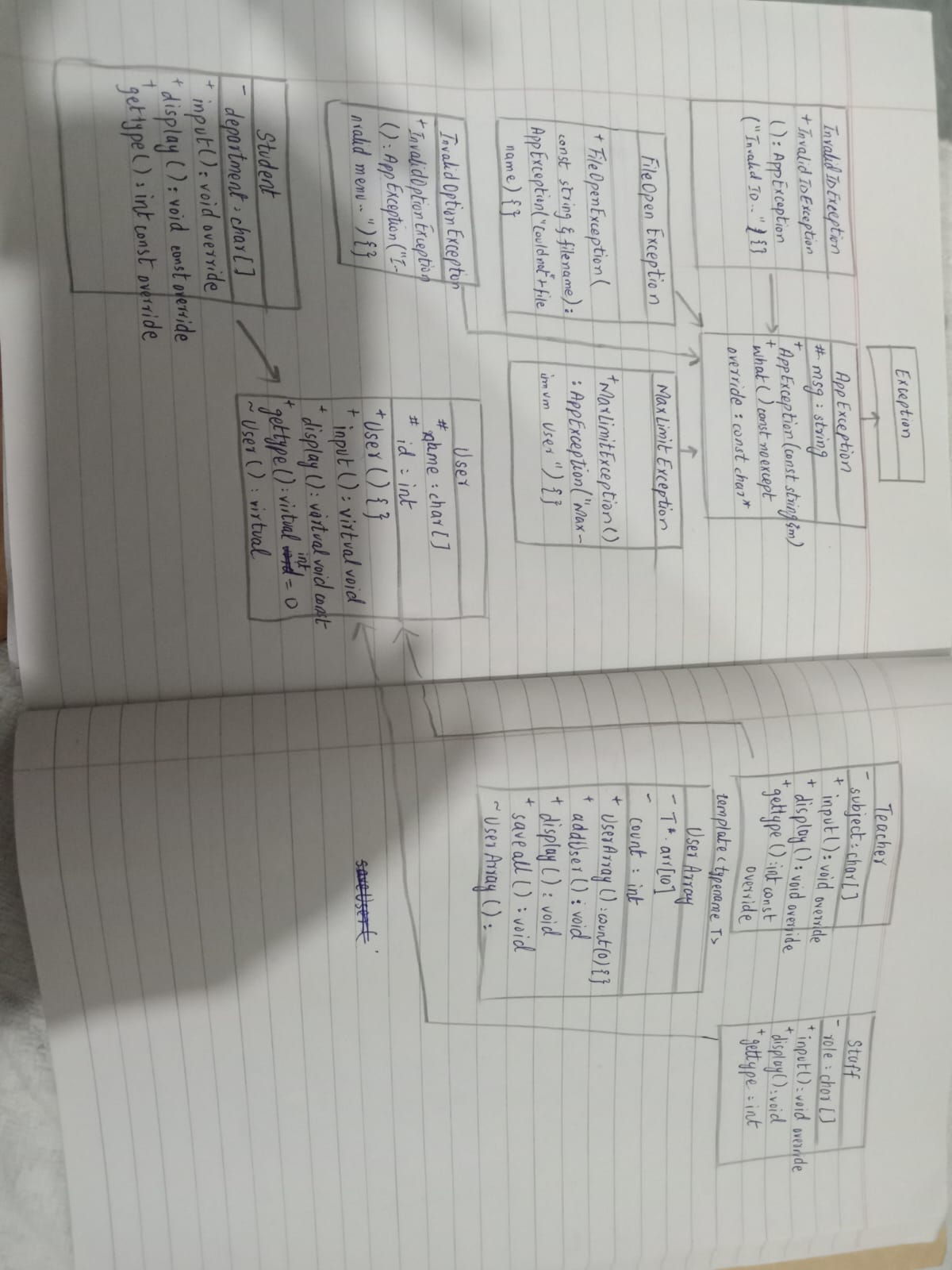
    } while (choice != 6);

    return 0;

}

**Output:**

****

**Class Diagram:  
**