Object Oriented Programming Lab report

Md Sahil
Roll – 001710501029
BCSE-II

C++ Assignment 4

Design a STUDENT class to store roll, name, course, admission date and marks in 5 subjects. Provide methods corresponding to admission date and receiving marks, preparing mark sheet. Support must be there to show the number of students who have taken admission, sort the student by their marks (CGPA/DGPA) dept-wise or all over the dept.

Design

Class ***student*** represents each student in the university. It stores roll, name, course, admission date and marks of five subjects It provides necessary accessor and mutator functions. The ***genRoll()*** function generates the roll number of the student according to his/her department and the number of students enrolled in that particular department. There are functions to set data, display data, update data, calculate grade and display marksheet of a student.

Class ***student_file*** is a container for the ***student*** class. It stores the objects (records) in a file. Provides methods for sorting, searching, adding students, removing students and displaying students. It also provides a support for defragmenting the file.

Class ***userInerface*** provides the necessary UI for the menu driven program.

The code

```cpp
#include<iostream>
#include<fstream>
#include<utility>
#include<string.h>
#include<time.h>
using namespace std;
#define Max 100


// class to hold student records
class student{
    int roll;
    char name[31];
    char course[21];
    char admission_date[31];
    pair<char[31],int> marks[5];

    //generates a roll number according to the department
    // and the number of students already admitted
    int genRoll(int courseId){
        char course[31];
        int r=0;
        switch (courseId)
        {
            case 1:
                strcpy(course,"Cse");
                break;
            case 2:
                strcpy(course,"Etce");
                break;

            case 3:
                strcpy(course,"Electrical");
                break;

            default:
                break;
        }
        int maxr=-1;
        ifstream f;
        student s;
        f.open("student.txt", ios::in | ios::binary);
        while(f.read((char*)&s, sizeof(s))){
            if(s.getRoll()>maxr && strcmp(s.getCourse(),course)==0)
                maxr=(s.getRoll())%100;
        }
        f.close();
        if(maxr==-1) maxr=0;
        roll = courseId*100+(maxr+1);
    }

    public:

    student(int t=-1){
        roll=t;
        strncpy(marks[0].first,"Physics",31);
        strncpy(marks[1].first,"Maths",31);
        strncpy(marks[2].first,"Chemestry",31);
        strncpy(marks[3].first,"English",31);
        strncpy(marks[4].first,"P.Ed",31);
    }

    void operator = (const student s){
        roll = s.roll;
        strcpy(name,s.name);
        strcpy(course,s.course);
        strcpy(admission_date,s.admission_date);
        for(int i=0;i<5;i++){
            marks[i].second=s.marks[i].second;
        }

    }

    void setData(){
        cout<<"Enter the name: ";
        cin.getline(name,31);
        cout<<"Course options: "<<endl;
        cout<<"1.Cse\n2.Etce\n3.Electrical\nEnter choice: ";
        int courseChoice;
        cin>>courseChoice;
        cin.ignore(100,'\n');
        switch (courseChoice)
        {
            case 1:
                strcpy(course,"Cse");
                genRoll(1);
                break;
            case 2:
                strcpy(course,"Etce");
                genRoll(2);
                break;

            case 3:
                strcpy(course,"Electrical");
                genRoll(3);
                break;

            default:
                break;
        }
        for(int i=0;i<5;i++){
            cout<<"Enter marks in "<<marks[i].first<<": ";
            cin>>marks[i].second;
        }
        cin.ignore(100,'\n');

        time_t now = time(0);
        char dt[100];
        strcpy(admission_date,ctime(&now));
        return;
    }

    void updateData(){
        cout<<"1. Update name"<<endl;
        cout<<"2. Update Course"<<endl;
        cout<<"3. Update admission date"<<endl;
        cout<<"4. Update marks in "<<marks[0].first<<endl;
        cout<<"5. Update marks in "<<marks[1].first<<endl;
        cout<<"6. Update marks in "<<marks[2].first<<endl;
        cout<<"7. Update marks in "<<marks[3].first<<endl;
        cout<<"8. Update marks in "<<marks[4].first<<endl;
        cout<<"Enter choice: ";
        int choice;
        cin>>choice;
        cin.ignore(100,'\n');
        switch(choice){
            case 1:{
                cout<<"Enter new name: ";
                char n_name[31];
                cin.getline(n_name,31);
                strcpy(name,n_name);
                break;
            }

            case 2:{
                cout<<"Course options: "<<endl;
                cout<<"1.Cse\n2.Etce\n3.Electrical\nEnter new choice: ";
                int courseChoice;
                cin>>courseChoice;
                cin.ignore(100,'\n');
                switch (courseChoice)
                {
                    case 1:
                        strcpy(course,"Cse");
                        genRoll(1);
                        break;
                    case 2:
                        strcpy(course,"Etce");
                        genRoll(2);
                        break;

                    case 3:
                        strcpy(course,"Electrical");
                        genRoll(3);
                        break;

                    default:
                        cout<<"Invalid input!"<<endl;
                        break;
                }
                break;
            }

            case 3:{
                cout<<"Enter new admission date: ";
                char n_admission_date[11];
                cin.getline(n_admission_date,11);
```

```cpp
                strcpy(admission_date,n_admission_date);
                break;
            }

            case 4:
                cout<<"Enter new marks in "<<marks[0].first<<": ";
                cin>>marks[0].second;
                break;

            case 5:
                cout<<"Enter new marks in "<<marks[1].first<<": ";
                cin>>marks[1].second;
                break;

            case 6:
                cout<<"Enter new marks in "<<marks[2].first<<": ";
                cin>>marks[2].second;
                break;

            case 7:
                cout<<"Enter new marks in "<<marks[3].first<<": ";
                cin>>marks[3].second;
                break;

            case 8:
                cout<<"Enter new marks in "<<marks[4].first<<": ";
                cin>>marks[4].second;
                break;

            default :
                cout<<"Invalid choice!"<<endl;
                break;
        }
    }

    void showData(){
        cout<<"Roll no.: "<<roll<<endl;
        cout<<"Name: "<<name<<endl;
        cout<<"Course: "<<course<<endl;
        cout<<"Admission date: "<<admission_date;
        for(int i=0;i<5;i++)
            cout<<"Marks in "<<marks[i].first<<": "<<marks[i].second<<endl;
        cout<<"Grade: "<<getGrade()<<endl;
        return;
    }

    void displayMarksheet(){
        cout<<endl<<endl;;
        cout<<"\tName\t\t\t: "<<getName()<<endl;
        cout<<"\tRoll\t\t\t: "<<getRoll()<<endl;
        cout<<"\tCourse\t\t\t: "<<getCourse()<<endl;
        cout<<"\tAdmission date\t\t: "<<getAdmissionDate()<<endl;
        cout<<"\t_________________________________________"<<endl;
        cout<<"\t|\t\tMARKSHEET\t\t|"<<endl;
        cout<<"\t-----------------------------------------"<<endl;
        cout<<"\t| "<<marks[0].first<<"\t\t\t: "<<marks[0].second<<"\t|"<<endl;
        cout<<"\t| "<<marks[1].first<<"\t\t\t\t: "<<marks[1].second<<"\t|"<<endl;
        cout<<"\t| "<<marks[2].first<<"\t\t\t: "<<marks[2].second<<"\t|"<<endl;
        cout<<"\t| "<<marks[3].first<<"\t\t\t: "<<marks[3].second<<"\t|"<<endl;
        cout<<"\t| "<<marks[4].first<<"\t\t\t\t: "<<marks[4].second<<"\t|"<<endl;
        cout<<"\t-----------------------------------------"<<endl;
        cout<<"\t| CGPA\t\t\t\t: "<<getGrade()<<"\t|"<<endl;
        cout<<"\t~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"<<endl;
        cout<<endl;
        return;
    }

    int getRoll(){
        return roll;
    }

    char* getName(){
        return name;
    }

    char* getCourse(){
        return course;
    }

    char* getAdmissionDate(){
        return admission_date;
    }

    pair<char[31],int>* getMarks(){
        return marks;
```

```cpp
    }

    //calculates the grade on the marks are provided
    float getGrade(){
        float grade=0;
        for(int i=0;i<5;i++){
            grade+=marks[i].second;
        }
        grade=grade/50;
        return grade;
    }
};


//stores the records in a file
class student_file{

    //string that stores the path to the file
    char file[51];

    void sortDeptWise(student s[Max],int n){
        for(int i=0;i<n;i++){
            for(int j=0;j<n;j++){
                if(strcmp(s[i].getCourse(),s[j].getCourse())>1){
                    student dummy=s[i];
                    s[i]=s[j];
                    s[j]=dummy;
                }
                else if(s[i].getCourse()==s[j].getCourse() && s[i].getRoll()>s[i].getRoll()){
                    student dummy=s[i];
                    s[i]=s[j];
                    s[j]=dummy;
                }
            }
        }
    }

    void sortGradeWise(student s[Max],int n){
        for(int i=0;i<n;i++){
            for(int j=0;j<n;j++){
                if(s[i].getGrade()>s[j].getGrade()){
                    student dummy=s[i];
                    s[i]=s[j];
                    s[j]=dummy;
                }
                else if(s[i].getGrade()==s[j].getGrade() && s[i].getRoll()>s[i].getRoll()){
                    student dummy=s[i];
                    s[i]=s[j];
                    s[j]=dummy;
                }
            }
        }
    }

    public:

    //returns a student according to the location in the file
    student retStudent(int k){
        ifstream f;
        student s;
        f.open(file, ios::in | ios::binary);
        f.seekg((k)*sizeof(student),ios::beg);
        f.read((char*)&s,sizeof(s));
        f.close();
        return s;
    }

    //returns the location of a student in the file
    //searches according to the roll no
    int searchRoll(int r){
        ifstream f;
        student s;
        int k=0;
        f.open(file, ios::in | ios::binary);
        while(f.read((char*)&s, sizeof(s))){
            if(s.getRoll()==r)
                return k;
            k++;
        }
        f.close();
        return -1;
    }

    void open(char* s){
```

```
    strcpy(file,s);
}

student_file(){
}

student_file(char *f){
    strcpy(file,f);
}

void addStudent(student s){
    if(searchRoll(s.getRoll())!=-1){
        cout<<"Roll no already exists!!"<<endl;
        getchar();
        return;
    }
    ofstream f;
    f.open(file, ios::out | ios::binary | ios::app);
    f.write((char*)&s,sizeof(s));
    f.close();
    return;
}

//removing involes replacing the entry with a black student object
//later removed by the defrag fucntion
student removeStudent(int r){
    fstream f("student.txt", ios::binary | ios::in | ios::out);
    student s;
    while(f.read((char*)&s,sizeof(s))){
        if(s.getRoll()==r){
            f.seekp(-1*sizeof(s),ios::cur);
            student dummy;
            f.write((char*)&dummy,sizeof(dummy));
            return s;
        }
    }
    f.close();
    cout<<"Corresponding entry does not exist!"<<endl;
    getchar();
    student dummy;
    return dummy;
}

void displayAll(){
    ifstream f("student.txt", ios::binary | ios::in);
    student st;
    student s[Max];
    int n =0;
    while(f.read((char*)&st,sizeof(st))){
        if(st.getRoll()>=0){
            s[n]=st;
            n++;
        }
    }
    f.close();

    cout<<"1. Dept wise"<<endl;
    cout<<"2. Grade wise sorted"<<endl;
    cout<<"Enter choice: ";
    int choice;
    cin>>choice;
    cin.ignore(100,'\n');
    switch (choice)
    {
        case 1:
            sortDeptWise(s,n);
            break;

        case 2:
            sortGradeWise(s,n);
            break;
        default:
            cout<<"Invalid input!"<<endl;
            getchar();
            return;
            break;
    }

    for(int i=0;i<n;i++){
        cout<<endl;
        s[i].showData();
    }
}

//returns the number of students admitted
```

```cpp
    int getCount(){
        ifstream f("student.txt", ios::binary | ios::in);
        student s;
        int count=0;
        while(f.read((char*)&s,sizeof(s))){
            if(s.getRoll()>=0)
                count++;
        }
        return count;
    }

    void updateInfo(int r){
        int k = searchRoll(r);
        if(k==-1){
            cout<<"Entry doesn't exist!"<<endl;
            getchar();
            return;
        }
        student s = retStudent(k);
        cout<<endl;
        cout<<"Current record:----------"<<endl;
        s.showData();
        cout<<endl;
        s.updateData();
        cout<<endl;
        cout<<"Updated record:----------"<<endl;
        s.showData();
        cout<<endl;
        cout<<"Confirm Update?(y/n): ";
        char choice;
        cin>>choice;
        cin.ignore(100,'\n');
        if(choice=='y'){
            removeStudent(r);
            addStudent(s);
        }
        else{
            cout<<"Update aborted!"<<endl;

        }
    }

    //removes the empty students records from the file
    void defrag(){
        char file_new[31];
        strcpy(file_new,"student_new.txt");
        ifstream f_old(file, ios::in | ios::binary);
        ofstream f_new(file_new, ios::out | ios::binary);
        student s;
        while(f_old.read((char*)&s,sizeof(s))){
            if(s.getRoll()!=-1)
                f_new.write((char*)&s,sizeof(s));
        }

        f_new.close();
        f_old.close();
        remove(file);
        rename(file_new,file);
        getchar();
        return;
    }
};


//UI of the program
class userInterface{
    student_file sfile;
    public:

    userInterface(char* s){
        sfile.open(s);
    }

    void showMenu(){
        while(1){
            system("clear");
            cout<<"------------MENU------------:"<<endl;
            cout<<"1.Display all students"<<endl;
            cout<<"2.Display no of admitted students"<<endl;
            cout<<"3.New student admission"<<endl;
            cout<<"4.Expell student"<<endl;
            cout<<"5.Update student Info"<<endl;
            cout<<"6.Display student marksheet"<<endl;
            cout<<"7.Defrag file"<<endl;
            cout<<"8.Exit"<<endl;

            int choice;
            cout<<"Enter choice: ";
            cin>>choice;
            cin.ignore(100,'\n');

            switch(choice){

                case 1:
                    sfile.displayAll();
                    getchar();
                    break;

                case 2:
                    cout<<"No of students enrolled: "<<sfile.getCount()<<endl;
                    getchar();
                    break;

                case 3:{
                    student student_new;
                    student_new.setData();
                    sfile.addStudent(student_new);
                    break;
                }

                case 4:{
                    cout<<"Enter the roll no of the student: ";
                    int r;
                    cin>>r;
                    cin.ignore(100,'\n');
                    student s = sfile.removeStudent(r);
                    cout<<endl<<"Entry deleted:"<<endl;
                    s.showData();
                    getchar();
                    break;
                }

                case 5:{
                    cout<<"Enter the roll no of the student: ";
                    int r;
                    cin>>r;
                    cin.ignore(100,'\n');
                    sfile.updateInfo(r);
                    break;
                }

                case 6:{
                    cout<<"Enter the roll no of the student: ";
                    int r;
                    cin>>r;
                    cin.ignore(100,'\n');
                    int k = sfile.searchRoll(r);
                    student s = sfile.retStudent(k);
                    s.displayMarksheet();
                    getchar();
                    break;
                }

                case 7:
                    sfile.defrag();
                    break;

                case 8:
                    return;

                default:
                    cout<<"Invalid Input!"<<endl;
                    getchar();
                    break;
            }
        }
    }
};


int main(){
    char s[51];
    strcpy(s,"student.txt");
    //student.txt is the file where the data is stored
    userInterface interface(s);
    interface.showMenu();
    return 0;
}
```

Output

```
------------MENU------------:
1.Display all students
2.Display no of admitted students
3.New student admission
4.Expell student
5.Update student Info
6.Display student marksheet
7.Defrag file
8.Exit
Enter choice: 1
1. Dept wise
2. Grade wise sorted
Enter choice: 1

Roll no.: 201
Name: Anindya chakraborty
Course: Etce
Admission date: Mon Mar 18 13:05:11 2019
Marks in Physics: 89
Marks in Maths: 89
Marks in Chemestry: 89
Marks in English: 89
Marks in P.Ed: 89
Grade: 8.9

Roll no.: 301
Name: Bikram Boote
Course: Electrical
Admission date: Mon Mar 18 13:05:52 2019
Marks in Physics: 78
Marks in Maths: 78
Marks in Chemestry: 78
Marks in English: 78
Marks in P.Ed: 78
Grade: 7.8

Roll no.: 101
Name: Sahil
Course: Cse
Admission date: Mon Mar 18 13:04:53 2019
Marks in Physics: 45
Marks in Maths: 45
Marks in Chemestry: 45
Marks in English: 45
Marks in P.Ed: 45
Grade: 4.5
```

Java Assignment 4

Create a class diagram and Java code for the following system and scenario, taking into account the possibility of future extensions. "The system is a command line utility that prints a short 'quote of the day' on the user's terminal when run. To begin with the quote is selected randomly from a set of hard-coded strings within the program itself, but that might change later on -- the quotes might be based on the user's history, the time of day, the date, etc...

Scenario

1. User types "java QuoteOfTheDay" on the command line.

2. System prints out a quote of the day, with an attribution

Design

Class ***Quote*** provides methods to read the quotes from the file. Its constructor argument is the file path. The function ***genId()*** generates a random index number. The quote of that index number is read from the file. There is also a queue of a specific size, it is stored in a file. It stores the recent number of indexes displayed. The ***genId()*** function generates a random integer (within range) then checks if it is present in the queue. If it is present it recursively calls itself else return the integer. This way quotes are not repeated within a certain no of runs of the program.

Class ***MyQueue*** is ***Serializable*** so that it can be stored in a file as an object.

Class ***Display*** checks the time and date. Night time quotes, Day time quotes and Special Day quotes are saved in separate files. The ***display()*** function selects the file according to the time and date and creates a queue object with the file path as constructor argument. Finally the quote is displayed.

The code

```java
import java.io.*;
import java.sql.Time;
import java.time.LocalDateTime;
import java.util.*;

//queue objects will store the index of the last maxLen no of quotes displayed
class MyQueue implements Serializable {
    int[] a;
    int maxLen;
    int end;

    MyQueue(int max) {
        maxLen = max;
        end = 0;
        a = new int[maxLen];
        for (int i = 0; i < maxLen; i++) {
            a[i] = -1;
        }
    }

    void push(int x) {
        a[end] = x;
        end = (end + 1) % maxLen;
    }

    boolean search(int x) {
        for (int i = 0; i < maxLen; i++) {
            if (a[i] == x)
                return true;
        }
        return false;
    }

    void show() {
        for (int i = 0; i < maxLen; i++) {
            if (a[i] != -1) {
                System.out.print(a[i] + " ");
            }
        }
        System.out.println();
    }
}

class Quote {

    // set variables here
    private static final int max = 10;
    private static final int maxq = 20;

    //stores the file address
    private static final String prev = new String("./quotes/prevQueue");
    MyQueue q;
    String file;

    //constructor
    Quote(String fileString) {
        file = new String(fileString);
        q = new MyQueue(max);
        try {
            ObjectInputStream oin = new ObjectInputStream(new FileInputStream(prev));
            q = (MyQueue) oin.readObject();
            // q.show();
        } catch (FileNotFoundException e) {
            // e.printStackTrace();
        } catch (IOException e) {
            // e.printStackTrace();
        } catch (ClassNotFoundException e) {
            // e.printStackTrace();
        }
    }

    //displays the quotes according to the index number (quoteNumber)
    public void showQuote(int quoteNumber) {
        try {
            BufferedReader br = new BufferedReader(new FileReader(file));
            String quote = new String();
            for (int i = 0; i < quoteNumber; i++) {
                quote = br.readLine();
            }
            System.out.println(quote);
        } catch (FileNotFoundException nf) {
            System.out.println("File not found");
        } catch (IOException ef) {
            System.out.println("IO exceptio occurred");
        }

        // pushing shown index to queue
        q.push(quoteNumber);

        // updating file
        try {
            ObjectOutputStream oout = new ObjectOutputStream(new FileOutputStream(prev, false));
            oout.writeObject(q);
        } catch (FileNotFoundException e) {
            // e.printStackTrace();
        } catch (IOException e) {
            // e.printStackTrace();
        } // catch (ClassNotFoundException e) {
          // e.printStackTrace();
          // }

    }

    //generates unique id (recursively)
    int genId() {
        int random = (int) new Random().nextInt(maxq - 1) + 1;
        if (q.search(random))
            return genId();
        else
            return random;
    }

    //calls the showQuote to show the quote
    public void show() {
        showQuote(genId());
    }
}

class Display {
    static public void display() {
        int hour = LocalDateTime.now().getHour();
        int day = LocalDateTime.now().getDayOfMonth();
        int month = LocalDateTime.now().getMonthValue();

        //selects the file according to the time and day.

        if (day == 15 && month == 8) {
            Quote q = new Quote("./quotes/independenceDay");
            q.show();
        }

        else if (hour > 19 || hour < 4) {
            Quote q = new Quote("./quotes/night");
            q.show();
        }

        else {
            Quote q = new Quote("./quotes/day");
            q.show();
        }
    }
}

public class quoteOfTheDay {
    public static void main(String[] args) {
        Display.display();
    }
}
```

Output

```
$ java quoteOfTheDay
Night is to see the dreams and day is to make them true. So its good to sleep now and see the
dreams. Good Night!

$ java quoteOfTheDay
May you dream of lovely things and to find them real.

$ java quoteOfTheDay
We are what we repeatedly do. Excellence then, is not an act, but a habit. | Aristotle
```