Ganpat University

Faculty of Engineering & Technology

Computer Science & Engineering

Name:- Dwij Vatsal Desai

Sem:- 2

Sub: - ESFP-II

Enrollment No.:- 23162121027

Prac:- 13

Practical 13

Definition:

Complete the code for the object assigned to you to satisfy following specifications.

1. For the solving purpose of given topic practical, you need to create minimum two classes, rest as per your requirement.
2. Declare minimum six function with the same name and same signature prototype. Function should be perform the following task like (read data record, display data record, delete data record, update data record, search data record and display all data record in ascending or in descending order. Perform this point by using pure virtual function and abstract class.
3. Implement the concept of virtual function also in practical.
4. Minimum 1 constructor method should be available in the program, rest as per your requirement.
5. You must use access specifier for data member and member function declaration in program.
6. Wherever is required to use character data member in class, instead of that use compulsorily string data member.
7. Take minimum 5 data record from the user and display according to the choice of user category wise. (Minimum six different options should be there for displaying information, and if you want to add more choice option as per your program requirement, you can add it.).
8. Use all possibility filter method from stored record information:
9. After all functionality execution, you need to call the destructor function.

Code:-

```
#include <iostream>
#include <string>
using namespace std;

const int MAX_STUDENTS = 100;

class StudentManager;
```

```
class StudentDatabase;

class Student {
    friend class StudentManager;
    friend class StudentDatabase;

private:
    string Name;
    int enroll_no;
    int sem;
    bool isPresent;
    string course;

public:
    Student() : isPresent(false) {}
    ~Student() {
        cout << "Destroying student: " << Name << endl;
    }

    void markAttendance(bool present) {
        isPresent = present;
        if (!isPresent) {
            sem--;  // Decrement semester for absent students
        }
    }

    void resetSem() {
        sem = 0;
    }

    void increaseSem(int amount) {
        sem += amount;
    }

    bool operator>(const Student& stu) {
        return Name > stu.Name;
    }

    bool operator<(const Student& stu) {
        return enroll_no < stu.enroll_no;
    }

    bool operator==(const Student& stu) {
        return sem == stu.sem;
    }

    friend ostream& operator<<(ostream& out, const Student& stu) {
        out << "Name: " << stu.Name << endl;
        out << "Enrollment Number: " << stu.enroll_no << endl;
        out << "Semester: " << stu.sem << endl;
        out << "Course: " << stu.course << endl;
        return out;
    }

    friend istream& operator>>(istream& in, Student& stu) {
        cout << "Enter student name: ";
        in >> stu.Name;
        cout << "Enter enrollment number: ";
        in >> stu.enroll_no;
        cout << "Enter semester: ";
        in >> stu.sem;
        cout << "Enter course: ";
        in >> stu.course;
        return in;
    }

    void setCourse(const string& crs) {
        course = crs;
```

```
    }

    string getAttendanceStatus() const {
        return isPresent ? "Present" : "Absent";
    }
};

class StudentManager {
private:
    Student students[MAX_STUDENTS];
    int numStudents;

public:
    StudentManager() : numStudents(0) {}
    ~StudentManager() {
        cout << "Destroying StudentManager..." << endl;
    }

    int getNumStudents() const {
        return numStudents;
    }

    const Student& getStudent(int index) const {
        return students[index];
    }

    void addStudent(const Student& stu) {
        if (numStudents < MAX_STUDENTS) {
            students[numStudents] = stu;
            numStudents++;
        } else {
            cout << "Maximum number of students reached." << endl;
        }
    }

    void markAttendance(int enroll, bool present) {
        for (int i = 0; i < numStudents; ++i) {
            if (students[i].enroll_no == enroll) {
                students[i].markAttendance(present);
                cout << "Attendance marked for student with Enrollment Number " <<
enroll << endl;
                return;
            }
        }
        cout << "Student not found with Enrollment Number " << enroll << endl;
    }

    void resetSem(int enroll) {
        for (int i = 0; i < numStudents; ++i) {
            if (students[i].enroll_no == enroll) {
                students[i].resetSem();
                cout << "Semester reset for student with Enrollment Number " <<
enroll << endl;
                return;
            }
        }
        cout << "Student not found with Enrollment Number " << enroll << endl;
    }

    void increaseSem(int enroll, int amount) {
        for (int i = 0; i < numStudents; ++i) {
            if (students[i].enroll_no == enroll) {
                students[i].increaseSem(amount);
                cout << "Semester increased for student with Enrollment Number " <<
enroll << endl;
                return;
            }
        }
```

```
        cout << "Student not found with Enrollment Number " << enroll << endl;
    }
};

class StudentDatabase {
public:
    void displayByCategory(const StudentManager& stuManager) {
        int choice;
        cout << "Display students by category:" << endl;
        cout << "1. Computer Science" << endl;
        cout << "2. Mechanical Engineering" << endl;
        cout << "3. Electrical Engineering" << endl;
        cout << "Enter choice: ";
        cin >> choice;

        cout << "Students:" << endl;
        for (int i = 0; i < stuManager.getNumStudents(); ++i) {
            const Student& student = stuManager.getStudent(i);
            if (choice == 1 && student.course == "Computer Science") {
                cout << student << "Attendance: " << student.getAttendanceStatus()
<< endl;
            } else if (choice == 2 && student.course == "Mechanical Engineering") {
                cout << student << "Attendance: " << student.getAttendanceStatus()
<< endl;
            } else if (choice == 3 && student.course == "Electrical Engineering") {
                cout << student << "Attendance: " << student.getAttendanceStatus()
<< endl;
            }
        }
    }
};

int main() {
    StudentManager stuManager;
    StudentDatabase stuDatabase;

    int choice;

    do {
        cout << "\nStudent Management System\n";
        cout << "1. Enter Student Details\n";
        cout << "2. Display All Students\n";
        cout << "3. Mark Attendance\n";
        cout << "4. Reset Semester\n";
        cout << "5. Increase Semester\n";
        cout << "6. Display Students by Category\n";
        cout << "7. Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
        case 1: {
            Student stu;
            cin >> stu;
            stuManager.addStudent(stu);
            break;
        }
        case 2: {
            cout << "\nStudent Details:" << endl;
            for (int i = 0; i < stuManager.getNumStudents(); ++i) {
                cout << stuManager.getStudent(i) << "Attendance: " <<
stuManager.getStudent(i).getAttendanceStatus() << endl;
            }
            break;
        }
        case 3: {
            int studentEnroll;
            bool present;
```

```
            cout << "\nEnter student enrollment number to mark attendance: ";
            cin >> studentEnroll;
            cout << "Enter 1 for present, 0 for absent: ";
            cin >> present;
            stuManager.markAttendance(studentEnroll, present);
            break;
        }
        case 4: {
            int studentEnroll;
            cout << "\nEnter student enrollment number to reset semester: ";
            cin >> studentEnroll;
            stuManager.resetSem(studentEnroll);
            break;
        }
        case 5: {
            int studentEnroll;
            int amount;
            cout << "\nEnter student enrollment number to increase semester: ";
            cin >> studentEnroll;
            cout << "Enter amount to increase: ";
            cin >> amount;
            stuManager.increaseSem(studentEnroll, amount);
            break;
        }
        case 6: {
            stuDatabase.displayByCategory(stuManager);
            break;
        }
        case 7: {
            cout << "Exiting program..." << endl;
            break;
        }
        default:
            cout << "Invalid choice. Please try again." << endl;
        }

    } while (choice != 7);

    return 0;
}
```

Output:-

```
"C:\Users\dwijd\OneDrive\Documents\collage practicals\ESFP-II\Practical_13.exe"

Student Management System
1. Enter Student Details
2. Display All Students
3. Mark Attendance
4. Reset Semester
5. Increase Semester
6. Display Students by Category
7. Exit
Enter your choice:1
 Enter student name:DWij
 Enter enrollment number:27
 Enter semester:2
 Enter course:CS
 Destroying student: DWij

Student Management System
1. Enter Student Details
2. Display All Students
3. Mark Attendance
4. Reset Semester
5. Increase Semester
6. Display Students by Category
7. Exit
Enter your choice:1
  Enter student name:ender
  Enter enrollment number:28
  Enter semester:3
  Enter course:BTech
  Destroying student: ender

 Student Management System
 1. Enter Student Details
 2. Display All Students
 3. Mark Attendance
 4. Reset Semester
 5. Increase Semester
 6. Display Students by Category
 7. Exit
 Enter your choice:2

 Student Details:
 Name: DWij
 Enrollment Number: 27
 Semester: 2
 Course: CS
 Attendance: Absent
 Name: ender
 Enrollment Number: 28
 Semester: 3
 Course: BTech
```

```
Attendance: Absent

Student Management System
1. Enter Student Details
2. Display All Students
3. Mark Attendance
4. Reset Semester
5. Increase Semester
6. Display Students by Category
7. Exit
Enter your choice:3

Enter student enrollment number to mark attendance:27
 Enter 1 for present, 0 for absent:1
 Attendance marked for student wit
h Enrollment Number 27

Student Management System
1. Enter Student Details
2. Display All Students
3. Mark Attendance
4. Reset Semester
5. Increase Semester
6. Display Students by Category
7. Exit
```

```
Enter your choice:4

Enter student enrollment number to reset semester:27
 Semester reset for student with Enrollment Number 27

Student Management System
1. Enter Student Details
2. Display All Students
3. Mark Attendance
4. Reset Semester
5. Increase Semester
6. Display Students by Category
7. Exit
Enter your choice:2

Student Details:
Name: DWij
Enrollment Number: 27
Semester: 0
Course: CS
Attendance: Present
Name: ender
Enrollment Number: 28
Semester: 3
Course: BTech
Attendance: Absent
```

```
Student Management System
1. Read Student Data
2. Display Student Data
3. Delete Student Data
4. Update Student Data
5. Search Student Data
6. Display All Students (Ascending)
7. Display All Students (Descending)
8. Exit
Enter your choice:7
 Name: Dwij
Enrollment Number: 27
Course: CS
Semester: 2

Name: krihsna
Enrollment Number: 25
Course: CS
Semester: 2
```