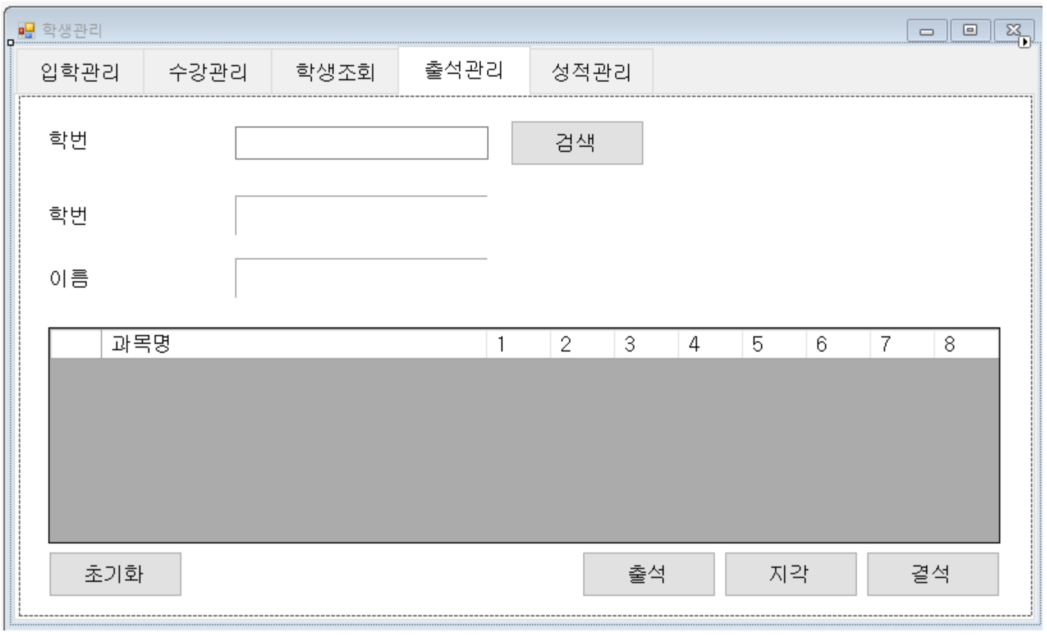
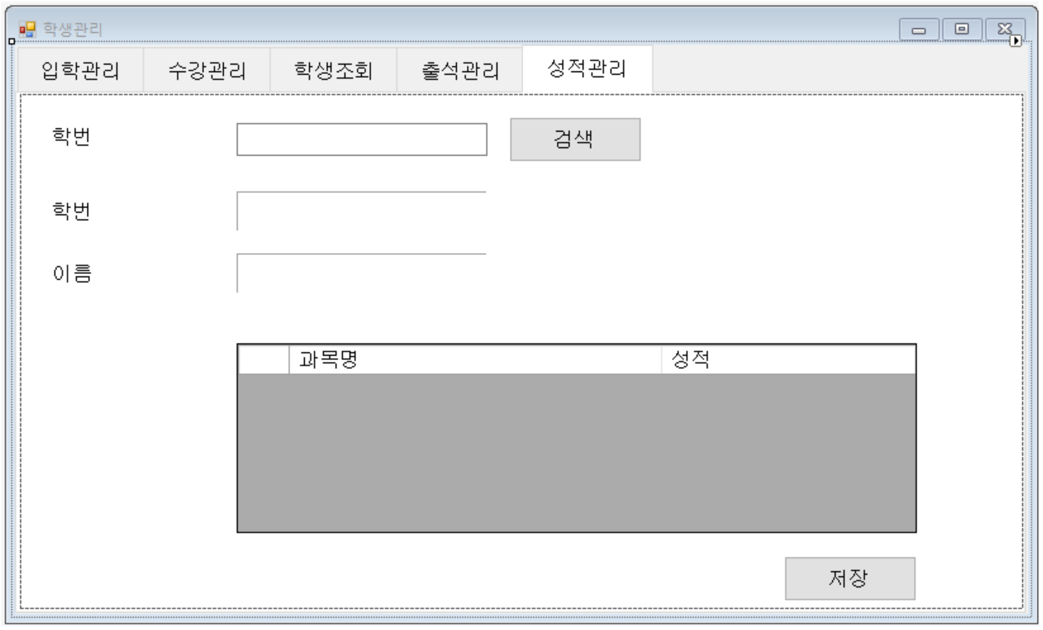
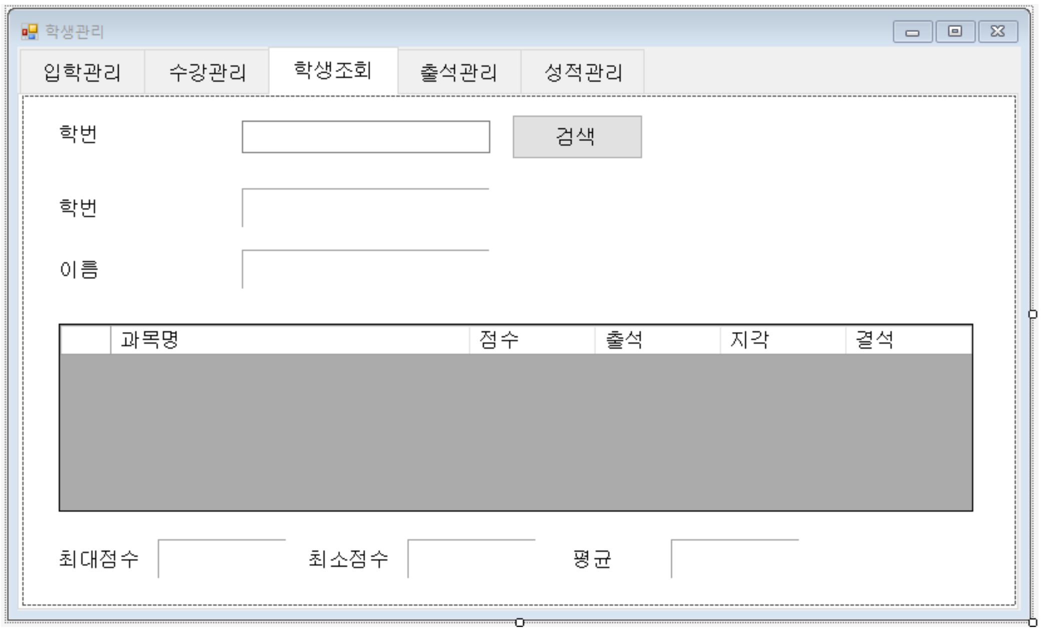
[화면]



학생관리
입학관리
수강관리
학생조회
출석관리
성적관리
학번
검색
학번
이름
과목명
1
2
3
4
5
6
7
8
초기화
출석
지각
결석



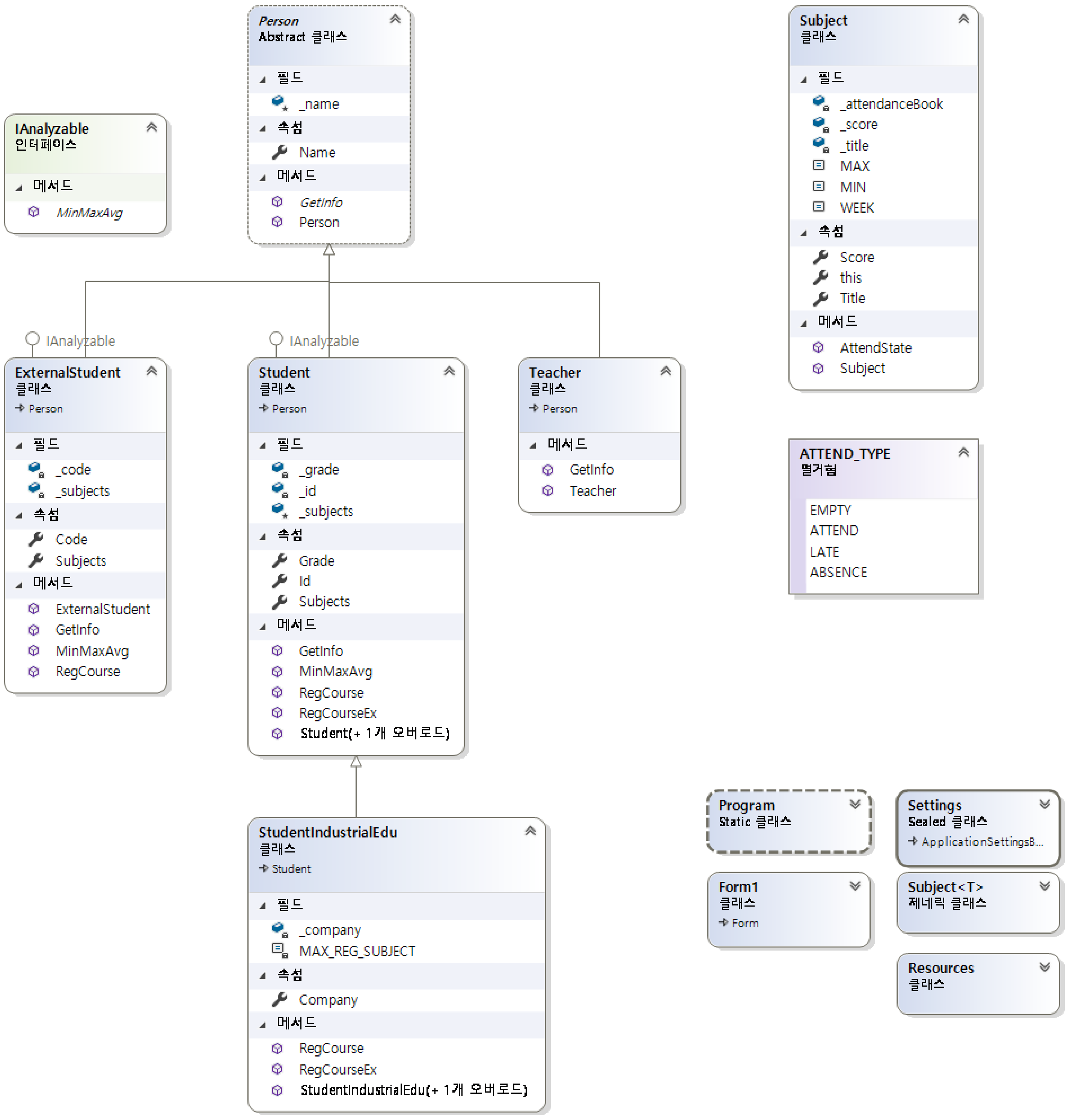
학생관리
입학관리
수강관리
학생조회
출석관리
성적관리
학번
검색
학번
이름
과목명
성적
저장



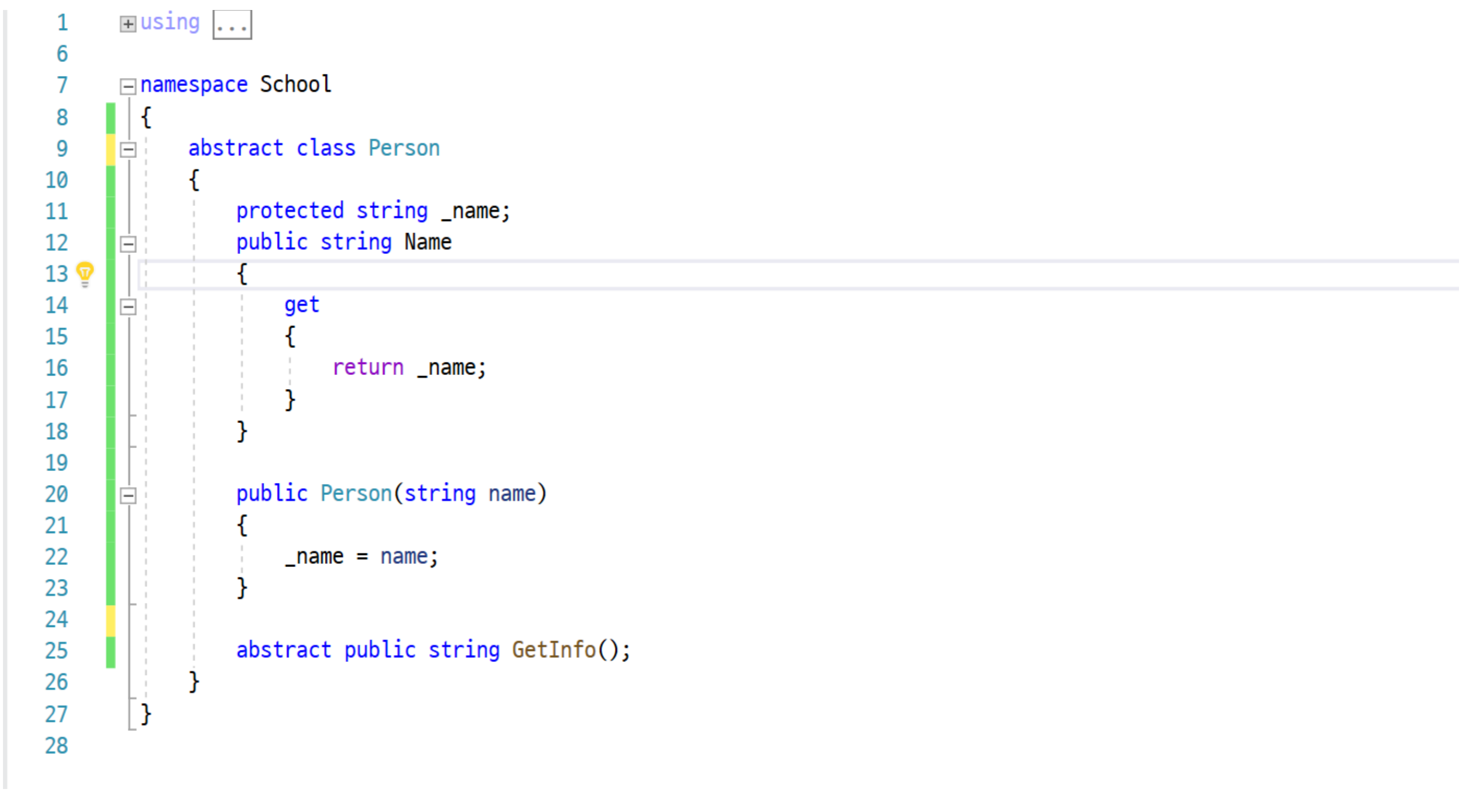
학생관리
입학관리
수강관리
학생조회
출석관리
성적관리
학번
검색
학번
이름
과목명
점수
출석
지각
결석
최대점수
최소점수
평균

1

[클래스 다이어그램]

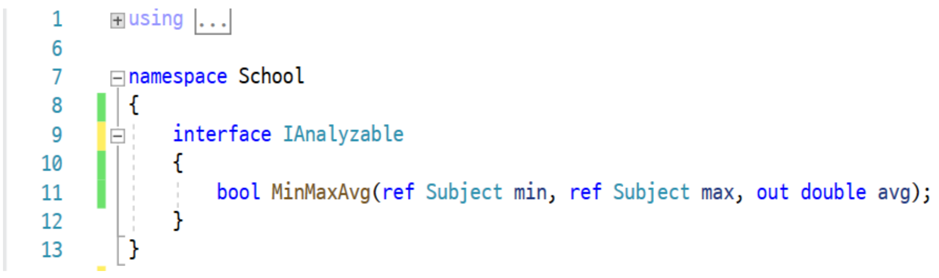

Person
Abstract 클래스
필드
_name
속성
Name
메서드
GetInfo
Person
IAnalyzable
인터페이스
메서드
MinMaxAvg
Subject
클래스
필드
_attendanceBook
_score
_title
MAX
MIN
WEEK
속성
Score
this
Title
메서드
AttendState
Subject
IAnalyzable
IAnalyzable
ExternalStudent
클래스
Person
필드
_code
_subjects
속성
Code
Subjects
메서드
ExternalStudent
GetInfo
MinMaxAvg
RegCourse
Student
클래스
Person
필드
_grade
_id
_subjects
속성
Grade
Id
Subjects
메서드
GetInfo
MinMaxAvg
RegCourse
RegCourseEx
Student(+ 1개 오버로드)
Teacher
클래스
Person
메서드
GetInfo
Teacher
ATTEND_TYPE
열거형
EMPTY
ATTEND
LATE
ABSENCE
StudentIndustrialEdu
클래스
Student
필드
_company
MAX_REG_SUBJECT
속성
Company
메서드
RegCourse
RegCourseEx
StudentIndustrialEdu(+ 1개 오버로드)
Program
Static 클래스
Settings
Sealed 클래스
ApplicationSettingsB...
Form1
클래스
Form
Subject<T>
제네릭 클래스
Resources
클래스

Person.cs



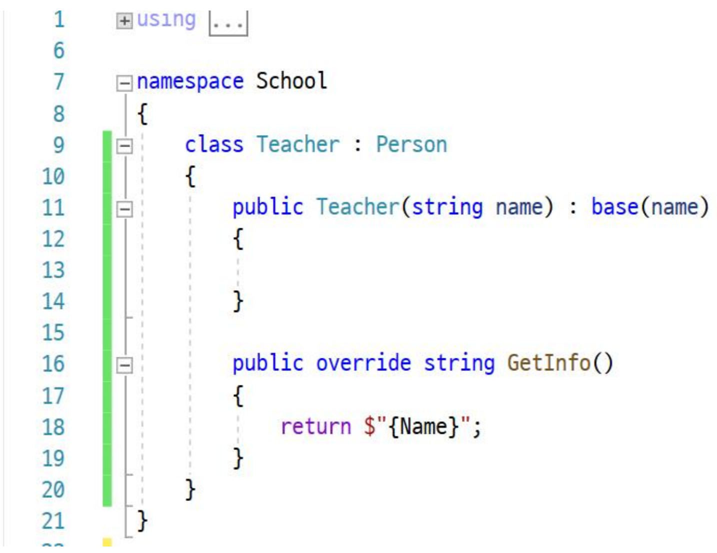
```
using ...

namespace School
{
    abstract class Person
    {
        protected string _name;
        public string Name
        {
            get
            {
                return _name;
            }
        }

        public Person(string name)
        {
            _name = name;
        }

        abstract public string GetInfo();
    }
}
```

IAnalyzable.cs



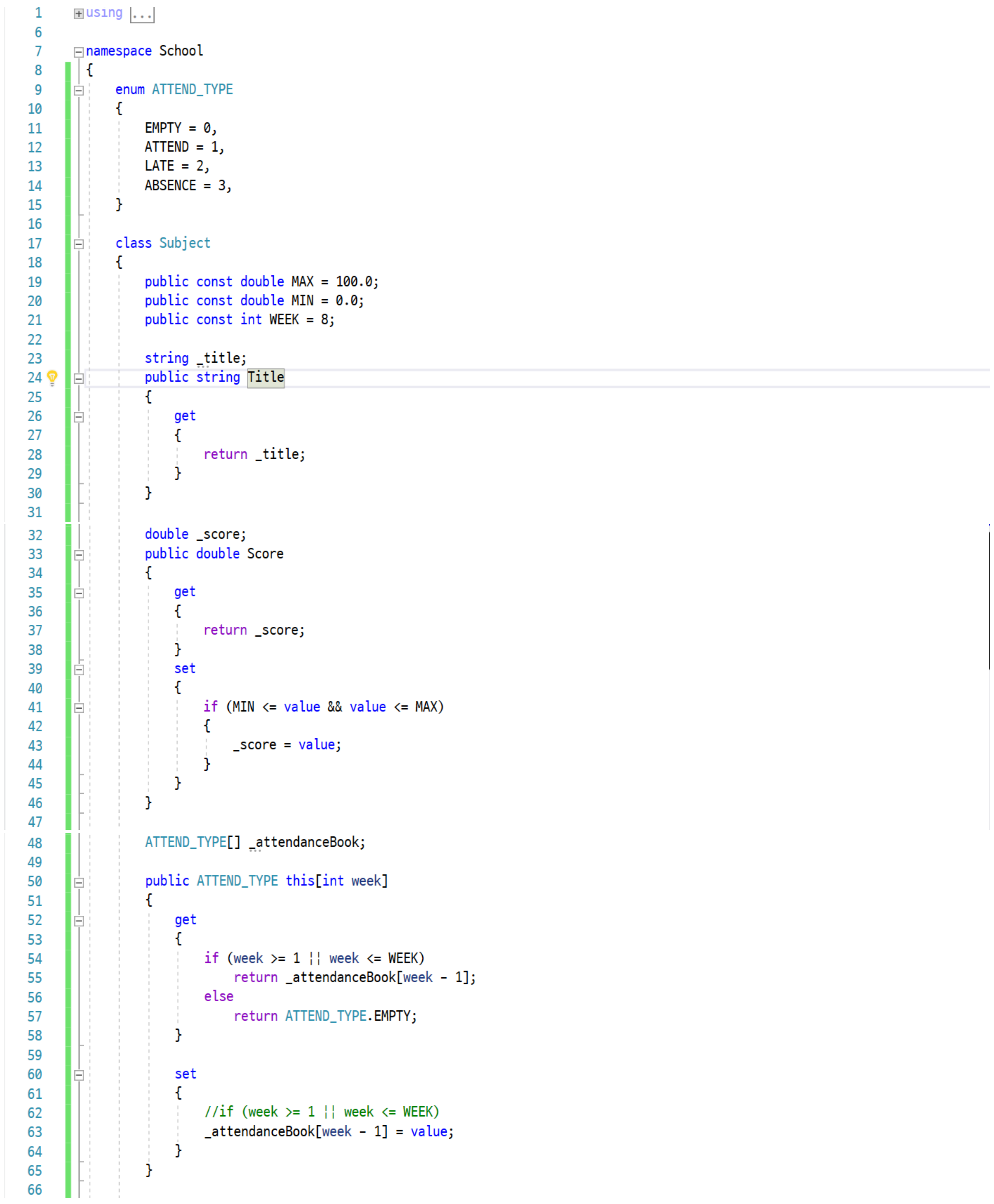
```
using ...

namespace School
{
    interface IAnalyzable
    {
        bool MinMaxAvg(ref Subject min, ref Subject max, out double avg);
    }
}
```

Teacher.cs

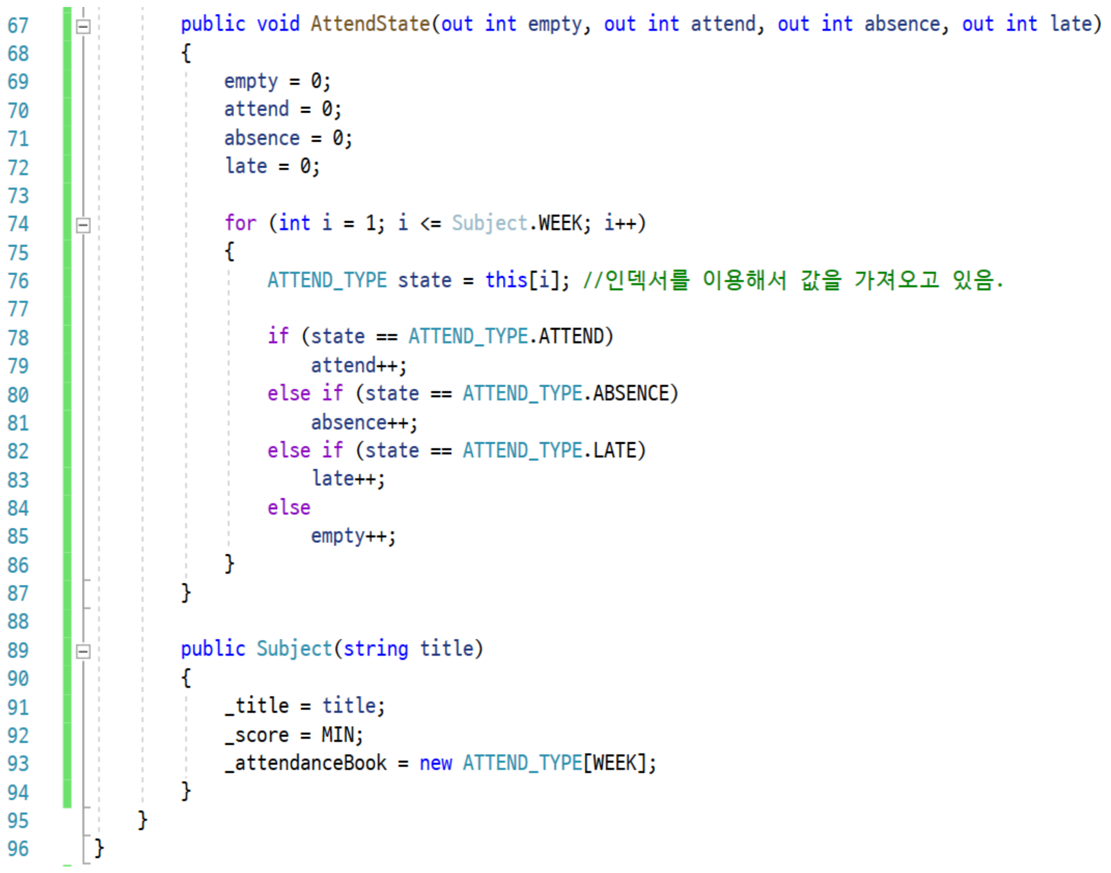


```
using ...

namespace School
{
    class Teacher : Person
    {
        public Teacher(string name) : base(name)
        {

        }

        public override string GetInfo()
        {
            return $"{Name}";
        }
    }
}
```

Subject.cs

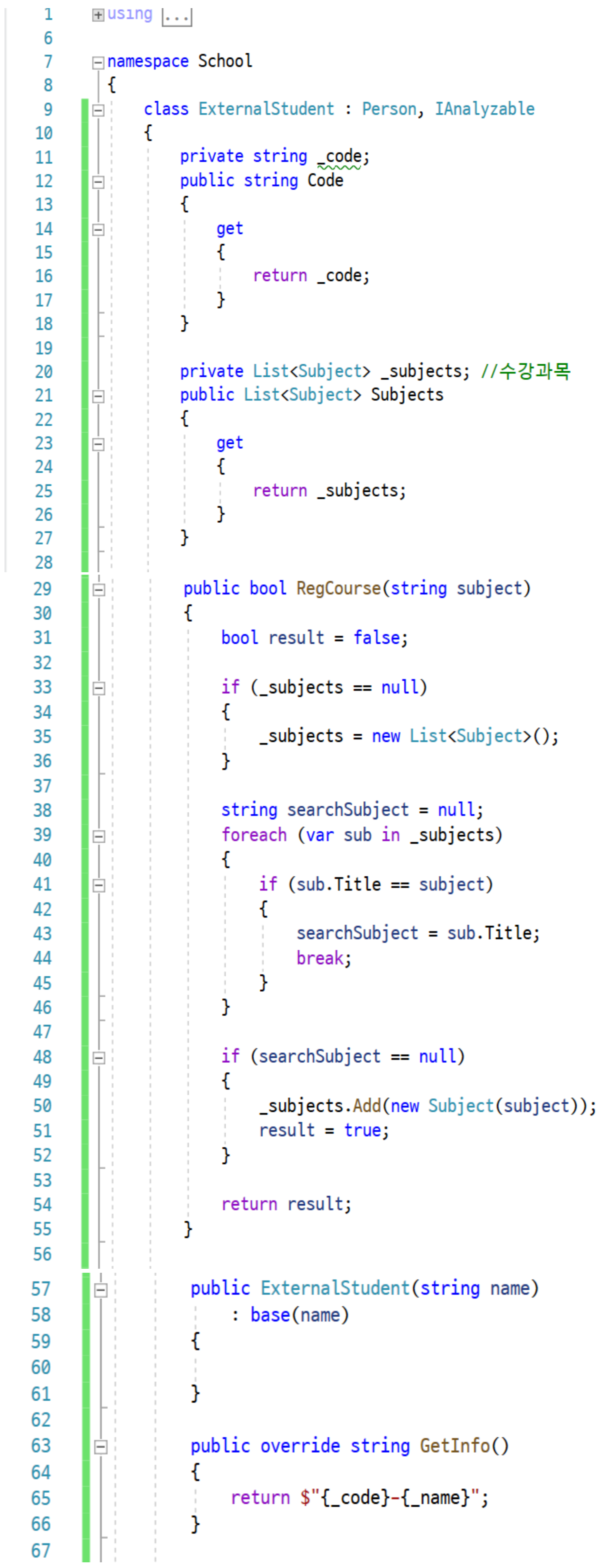


```csharp
using ...

namespace School
{
    enum ATTEND_TYPE
    {
        EMPTY = 0,
        ATTEND = 1,
        LATE = 2,
        ABSENCE = 3,
    }

    class Subject
    {
        public const double MAX = 100.0;
        public const double MIN = 0.0;
        public const int WEEK = 8;

        string _title;
        public string Title
        {
            get
            {
                return _title;
            }
        }

        double _score;
        public double Score
        {
            get
            {
                return _score;
            }
            set
            {
                if (MIN <= value && value <= MAX)
                {
                    _score = value;
                }
            }
        }

        ATTEND_TYPE[] _attendanceBook;

        public ATTEND_TYPE this[int week]
        {
            get
            {
                if (week >= 1 || week <= WEEK)
                    return _attendanceBook[week - 1];
                else
                    return ATTEND_TYPE.EMPTY;
            }

            set
            {
                //if (week >= 1 || week <= WEEK)
                _attendanceBook[week - 1] = value;
            }
        }

```

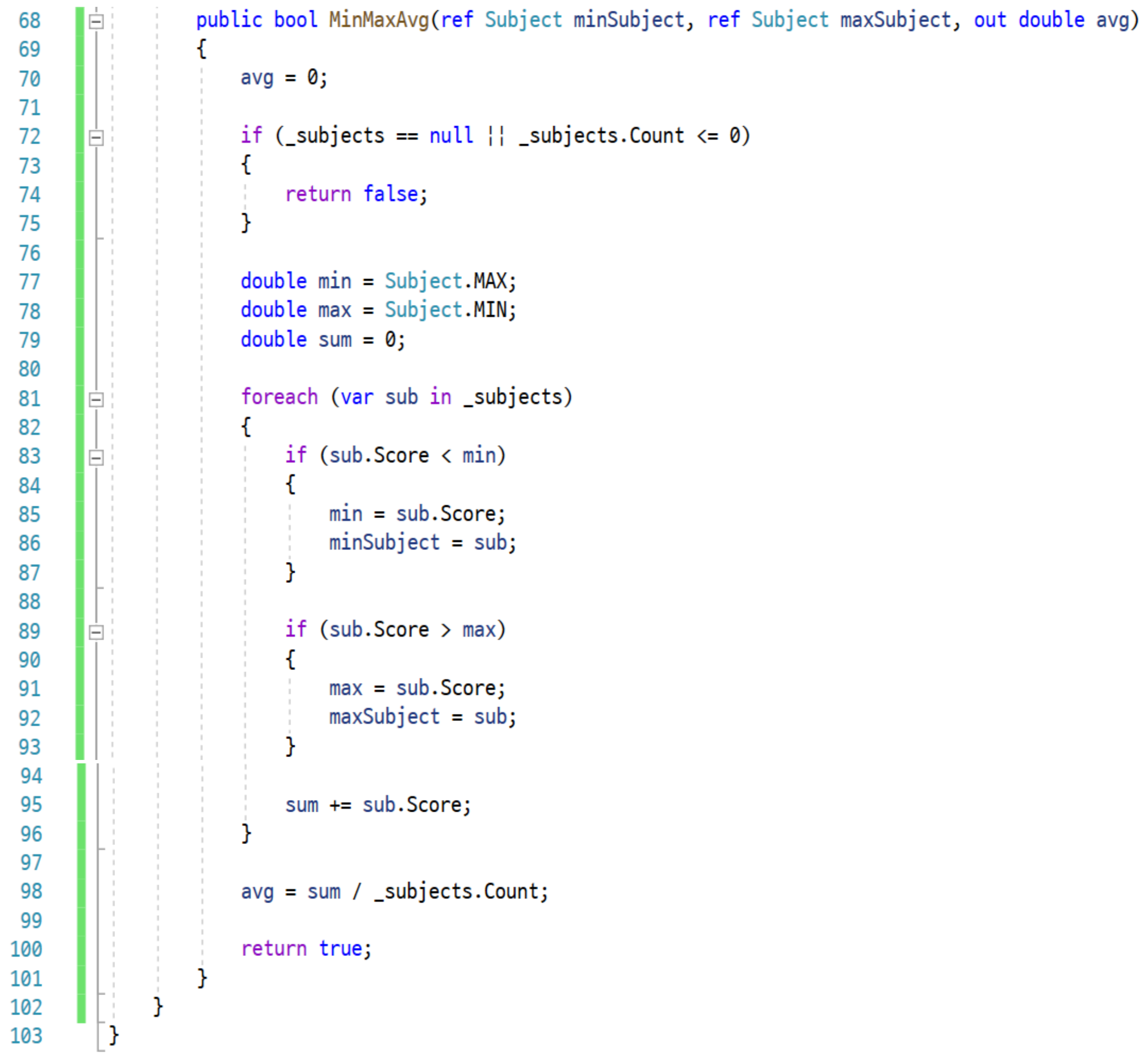


```
        public void AttendState(out int empty, out int attend, out int absence, out int late)
        {
            empty = 0;
            attend = 0;
            absence = 0;
            late = 0;

            for (int i = 1; i <= Subject.WEEK; i++)
            {
                ATTEND_TYPE state = this[i]; //인덱서를 이용해서 값을 가져오고 있음.

                if (state == ATTEND_TYPE.ATTEND)
                    attend++;
                else if (state == ATTEND_TYPE.ABSENCE)
                    absence++;
                else if (state == ATTEND_TYPE.LATE)
                    late++;
                else
                    empty++;
            }
        }

        public Subject(string title)
        {
            _title = title;
            _score = MIN;
            _attendanceBook = new ATTEND_TYPE[WEEK];
        }
    }
}
```

ExternalStudent.cs



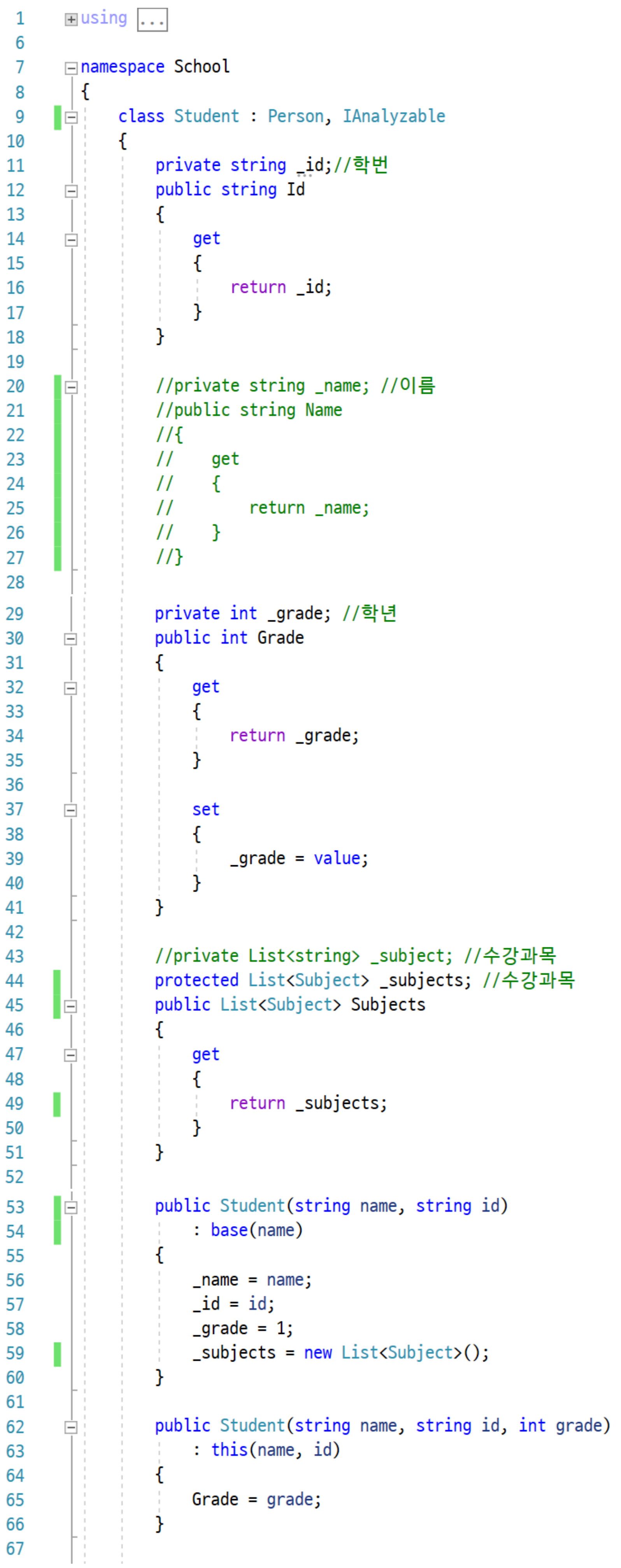
```
using ...

namespace School
{
    class ExternalStudent : Person, IAnalyzable
    {
        private string _code;
        public string Code
        {
            get
            {
                return _code;
            }
        }

        private List<Subject> _subjects; //수강과목
        public List<Subject> Subjects
        {
            get
            {
                return _subjects;
            }
        }

        public bool RegCourse(string subject)
        {
            bool result = false;

            if (_subjects == null)
            {
                _subjects = new List<Subject>();
            }

            string searchSubject = null;
            foreach (var sub in _subjects)
            {
                if (sub.Title == subject)
                {
                    searchSubject = sub.Title;
                    break;
                }
            }

            if (searchSubject == null)
            {
                _subjects.Add(new Subject(subject));
                result = true;
            }

            return result;
        }

        public ExternalStudent(string name)
            : base(name)
        {

        }

        public override string GetInfo()
        {
            return $"{_code}-{_name}";
        }

```



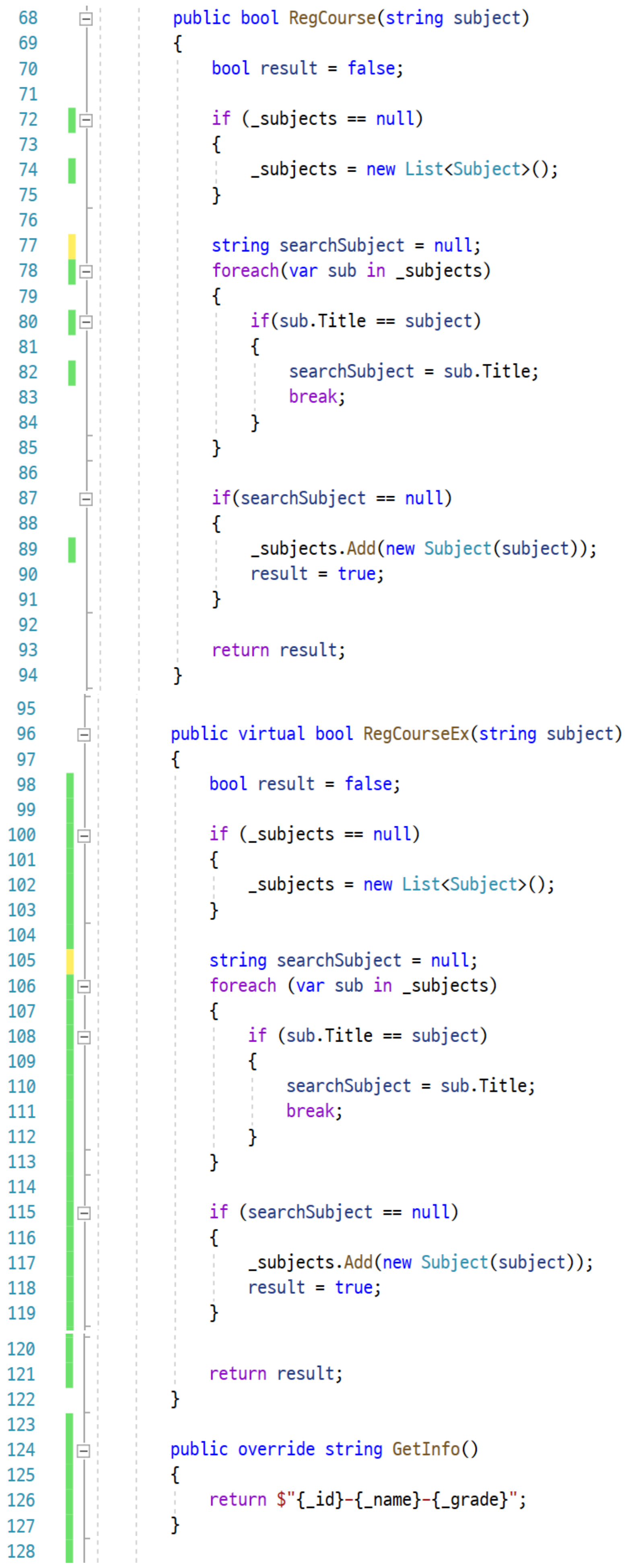
```
        public bool MinMaxAvg(ref Subject minSubject, ref Subject maxSubject, out double avg)
        {
            avg = 0;

            if (_subjects == null || _subjects.Count <= 0)
            {
                return false;
            }

            double min = Subject.MAX;
            double max = Subject.MIN;
            double sum = 0;

            foreach (var sub in _subjects)
            {
                if (sub.Score < min)
                {
                    min = sub.Score;
                    minSubject = sub;
                }

                if (sub.Score > max)
                {
                    max = sub.Score;
                    maxSubject = sub;
                }

                sum += sub.Score;
            }

            avg = sum / _subjects.Count;

            return true;
        }
    }
}
```

1

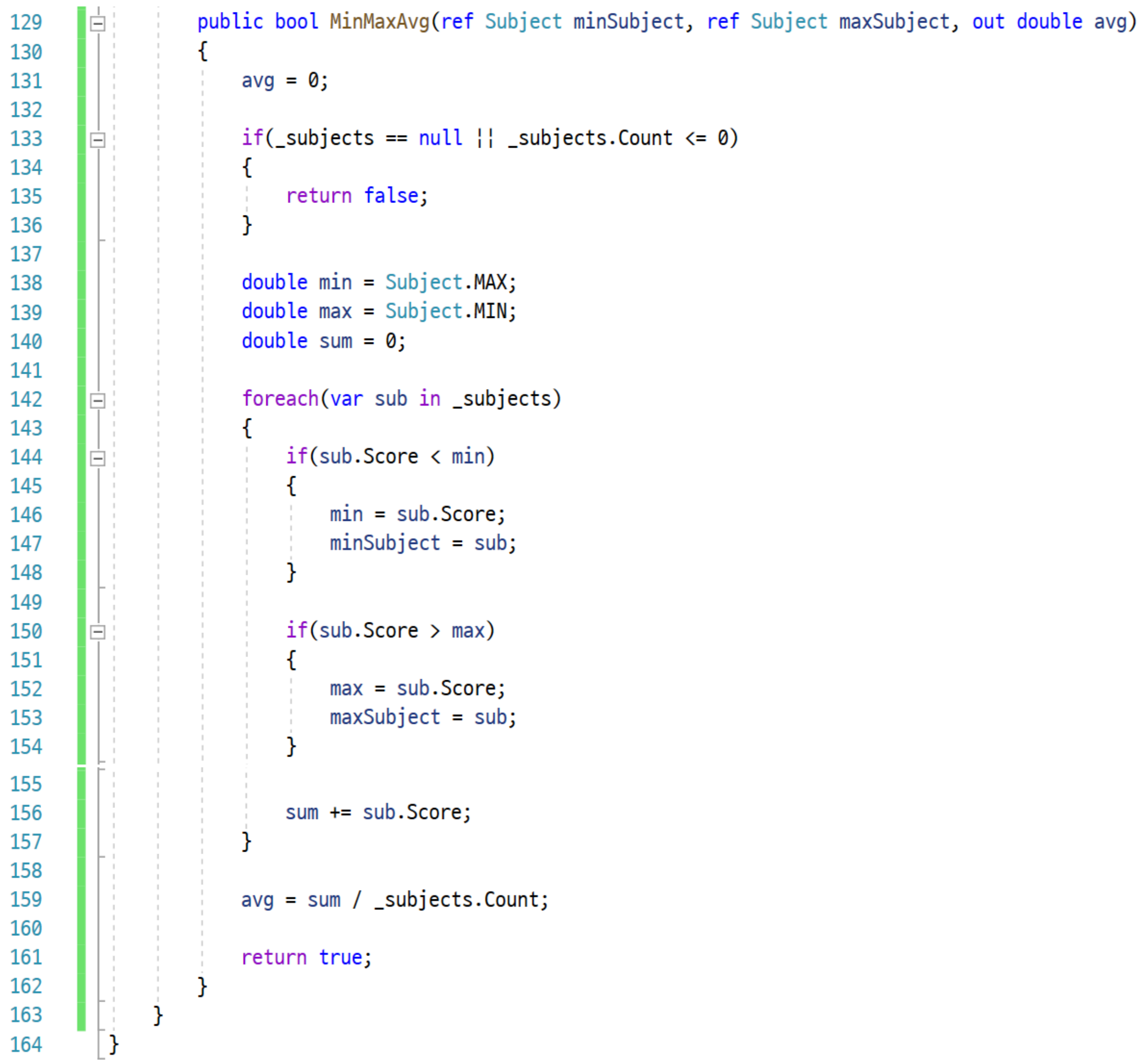
Student.cs



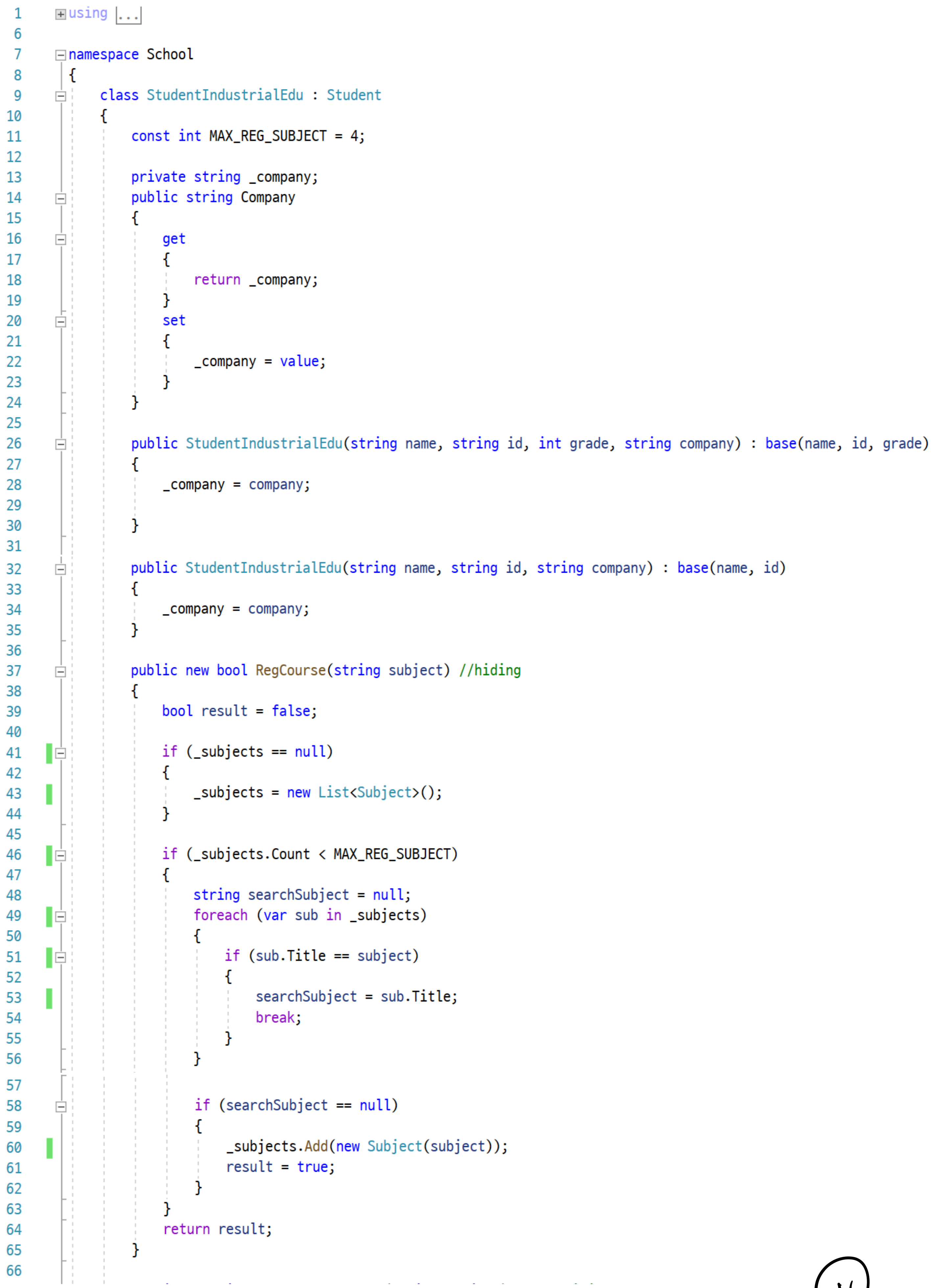
```csharp
using ...

namespace School
{
    class Student : Person, IAnalyzable
    {
        private string _id;//학번
        public string Id
        {
            get
            {
                return _id;
            }
        }

        //private string _name; //이름
        //public string Name
        //{
        //    get
        //    {
        //        return _name;
        //    }
        //}

        private int _grade; //학년
        public int Grade
        {
            get
            {
                return _grade;
            }

            set
            {
                _grade = value;
            }
        }

        //private List<string> _subject; //수강과목
        protected List<Subject> _subjects; //수강과목
        public List<Subject> Subjects
        {
            get
            {
                return _subjects;
            }
        }

        public Student(string name, string id)
            : base(name)
        {
            _name = name;
            _id = id;
            _grade = 1;
            _subjects = new List<Subject>();
        }

        public Student(string name, string id, int grade)
            : this(name, id)
        {
            Grade = grade;
        }

```



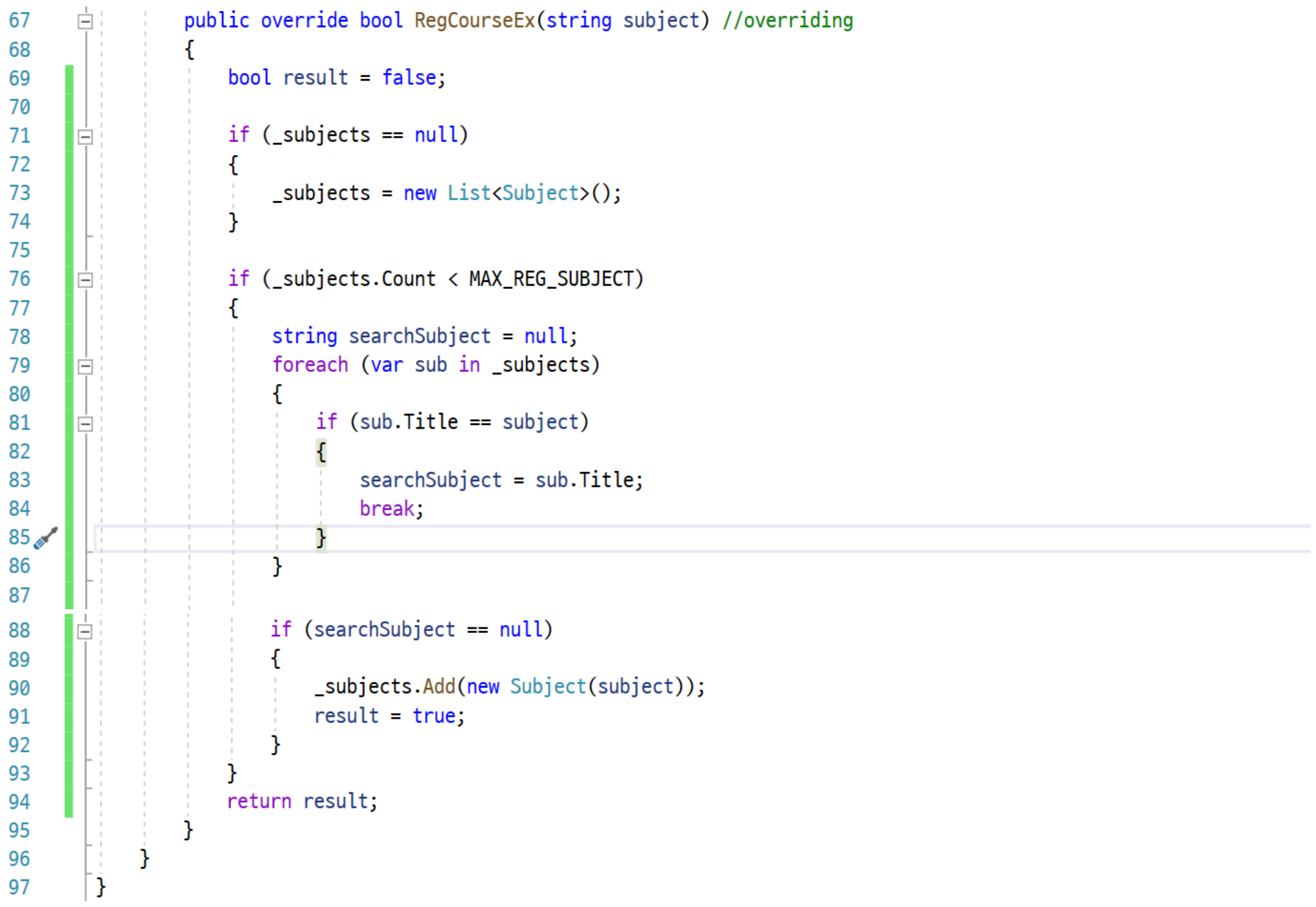
```
public bool RegCourse(string subject)
{
    bool result = false;

    if (_subjects == null)
    {
        _subjects = new List<Subject>();
    }

    string searchSubject = null;
    foreach(var sub in _subjects)
    {
        if(sub.Title == subject)
        {
            searchSubject = sub.Title;
            break;
        }
    }

    if(searchSubject == null)
    {
        _subjects.Add(new Subject(subject));
        result = true;
    }

    return result;
}

public virtual bool RegCourseEx(string subject)
{
    bool result = false;

    if (_subjects == null)
    {
        _subjects = new List<Subject>();
    }

    string searchSubject = null;
    foreach (var sub in _subjects)
    {
        if (sub.Title == subject)
        {
            searchSubject = sub.Title;
            break;
        }
    }

    if (searchSubject == null)
    {
        _subjects.Add(new Subject(subject));
        result = true;
    }

    return result;
}

public override string GetInfo()
{
    return $"{_id}-{_name}-{_grade}";
}

```


```
        public bool MinMaxAvg(ref Subject minSubject, ref Subject maxSubject, out double avg)
        {
            avg = 0;

            if(_subjects == null || _subjects.Count <= 0)
            {
                return false;
            }

            double min = Subject.MAX;
            double max = Subject.MIN;
            double sum = 0;

            foreach(var sub in _subjects)
            {
                if(sub.Score < min)
                {
                    min = sub.Score;
                    minSubject = sub;
                }

                if(sub.Score > max)
                {
                    max = sub.Score;
                    maxSubject = sub;
                }

                sum += sub.Score;
            }

            avg = sum / _subjects.Count;

            return true;
        }
    }
}
```

StudentIndustrialEdu.cs



```
using ...

namespace School
{
    class StudentIndustrialEdu : Student
    {
        const int MAX_REG_SUBJECT = 4;

        private string _company;
        public string Company
        {
            get
            {
                return _company;
            }
            set
            {
                _company = value;
            }
        }

        public StudentIndustrialEdu(string name, string id, int grade, string company) : base(name, id, grade)
        {
            _company = company;

        }

        public StudentIndustrialEdu(string name, string id, string company) : base(name, id)
        {
            _company = company;
        }

        public new bool RegCourse(string subject) //hiding
        {
            bool result = false;

            if (_subjects == null)
            {
                _subjects = new List<Subject>();
            }

            if (_subjects.Count < MAX_REG_SUBJECT)
            {
                string searchSubject = null;
                foreach (var sub in _subjects)
                {
                    if (sub.Title == subject)
                    {
                        searchSubject = sub.Title;
                        break;
                    }
                }

                if (searchSubject == null)
                {
                    _subjects.Add(new Subject(subject));
                    result = true;
                }
            }
            return result;
        }

```



```
        public override bool RegCourseEx(string subject) //overriding
        {
            bool result = false;

            if (_subjects == null)
            {
                _subjects = new List<Subject>();
            }

            if (_subjects.Count < MAX_REG_SUBJECT)
            {
                string searchSubject = null;
                foreach (var sub in _subjects)
                {
                    if (sub.Title == subject)
                    {
                        searchSubject = sub.Title;
                        break;
                    }
                }

                if (searchSubject == null)
                {
                    _subjects.Add(new Subject(subject));
                    result = true;
                }
            }
            return result;
        }
    }
}
```

Form1.cs

```csharp
using ...

namespace School
{
    public partial class Form1 : Form
    {
        const int MAX_GRADE = 4;
        List<Student> _students;

        Student _studentRegCourse = null;
        Student _studentAttend = null;
        Student _studentScore = null;
        Student _studentView = null;

        public Form1()
        {
            InitializeComponent();

            _students = new List<Student>();

            for (int i = 1; i <= MAX_GRADE; i++)
            {
                cbxEntranceGrade.Items.Add(i.ToString());
            }
            cbxEntranceGrade.SelectedIndex = 0;
        }

        private void chkIndustrialEdu_CheckedChanged(object sender, EventArgs e)
        {
            pnlEntranceCompany.Visible = chkIndustrialEdu.Checked;
        }

        private void btnEntrance_Click(object sender, EventArgs e)
        {
            string temp_str = null;
            int temp_int = 0;

            int grade = 1;
            int.TryParse(cbxEntranceGrade.SelectedItem.ToString(), out grade);

            int year = DateTime.Now.Year - (grade - 1);
            string id = string.Empty;
            if (_students != null && _students.Count > 0)
            {
                temp_str = year.ToString("0000");
                temp_str = _students.Where(m => m.Id.StartsWith(temp_str))
                                .OrderByDescending(m => m.Id)
                                .Select(m => m.Id)
                                .FirstOrDefault();

                if (string.IsNullOrEmpty(temp_str))
                {
                    id = $"{year:0000}0001";
                }
                else
                {
                    int.TryParse(temp_str.Substring(4, 4), out temp_int);
                    id = id = $"{year:0000}{temp_int + 1:0000}";
                }
            }
            else
            {
                id = $"{year:0000}0001";
            }

            Student student = null;
            if (chkIndustrialEdu.Checked)
            {
                student = new StudentIndustrialEdu(tbxEntranceName.Text, id, grade, tbxEntranceCompany.Text);
            }
            else
            {
                student = new Student(tbxEntranceName.Text, id, grade);
            }
            _students.Add(student);
            lblEntranceResult.Text
                = $"등록학생의 정보입니다.\r\n학번 : {student.Id}\r\n이름 : {student.Name}\r\n학년 : {student.Grade}\r\n";

            StudentIndustrialEdu studentIE = student as StudentIndustrialEdu;
            if (studentIE != null)
            {
                lblEntranceResult.Text += $"[산학과정] 소속회사:{studentIE.Company}";
            }
        }

        private void btnRegCourseSearch_Click(object sender, EventArgs e)
        {
            if (_students == null || _students.Count <= 0)
            {
                MessageBox.Show("검색할 학생이 없습니다.");
                return;
            }

            _studentRegCourse = null;
            foreach (var stu in _students)
            {
                if (stu.Id == tbxRegCourseSearchId.Text)
                {
                    _studentRegCourse = stu;
                    break;
                }
            }

            if (_studentRegCourse == null)
            {
                lblRegCourseId.Text = string.Empty;
                lblRegCourseName.Text = string.Empty;
                lbxRegCourse.Items.Clear();

                MessageBox.Show("해당 학생을 찾을 수 없습니다.");
            }
            else
            {
                lblRegCourseId.Text = _studentRegCourse.Id;
                lblRegCourseName.Text = _studentRegCourse.Name;
                lbxRegCourse.Items.Clear();
                foreach (var sub in _studentRegCourse.Subjects)
                {
                    lbxRegCourse.Items.Add(sub);
                }
            }
        }

        private void btnRegCourseHiding_Click(object sender, EventArgs e)
        {
            if (_studentRegCourse != null)
            {
                if (_studentRegCourse.RegCourse(tbxRegCourseName.Text))//hiding
                {
                    lbxRegCourse.Items.Add(tbxRegCourseName.Text);
                }
                else
                {
                    MessageBox.Show("등록실패");
                }
            }
        }

        private void btnRegCourseOverriding_Click(object sender, EventArgs e)
        {
            if (_studentRegCourse != null)
            {
                if (_studentRegCourse.RegCourseEx(tbxRegCourseName.Text))//overridng**
                {
                    lbxRegCourse.Items.Add(tbxRegCourseName.Text);
                }
                else
                {
                    MessageBox.Show("등록실패");
                }
            }
        }

        private void btnAttendSearch_Click(object sender, EventArgs e)
        {
            if (_students == null || _students.Count <= 0)
            {
                MessageBox.Show("검색할 학생이 없습니다.");
                return;
            }

            _studentAttend = null;
            foreach (var stu in _students)
            {
                if (stu.Id == tbxAttendSearchId.Text)
                {
                    _studentAttend = stu;
                    break;
                }
            }

            if (_studentAttend == null)
            {
                lblAttendId.Text = string.Empty;
                lblAttendName.Text = string.Empty;
                dgvAttend.Rows.Clear();

                MessageBox.Show("해당 학생을 찾을 수 없습니다.");
            }
            else
            {
                lblAttendId.Text = _studentAttend.Id;
                lblAttendName.Text = _studentAttend.Name;

                SetAttendState();
            }
        }

        private void SetAttendState()
        {
            dgvAttend.Rows.Clear();

            foreach (Subject sub in _studentAttend.Subjects)
            {
                int index = dgvAttend.Rows.Add();
                dgvAttend.Rows[index].Cells[0].Value = sub.Title;

                for (int i = 1; i <= Subject.WEEK; i++)
                {
                    dgvAttend.Rows[index].Cells[i].Value = GetStringState(sub, i);
                }
            }
        }

        private string GetStringState(Subject sub, int week)
        {
            if (1 > week || week > Subject.WEEK)
            {
                return string.Empty;
            }

            ATTEND_TYPE state = sub[week];
            string display = string.Empty;

            switch (state)
            {
                case ATTEND_TYPE.ABSENCE:
                    display = "결";
                    break;
                case ATTEND_TYPE.ATTEND:
                    display = "출";
                    break;
                case ATTEND_TYPE.LATE:
                    display = "지";
                    break;
            }

            return display;
        }

        private void btnAttend_Click(object sender, EventArgs e)
        {
            Button button = sender as Button;

            if (_studentAttend == null) return;
            if (dgvAttend.SelectedCells.Count != 1) return;
            if (dgvAttend.SelectedCells[0].ColumnIndex < 1) return;

            int week = dgvAttend.SelectedCells[0].ColumnIndex;
            int sub_index = dgvAttend.SelectedCells[0].RowIndex;
            Subject subject = _studentAttend.Subjects[sub_index];
            ATTEND_TYPE state = ATTEND_TYPE.EMPTY;

            if (button == btnAttend1) state = ATTEND_TYPE.EMPTY;
            else if (button == btnAttend2) state = ATTEND_TYPE.ATTEND;
            else if (button == btnAttend3) state = ATTEND_TYPE.LATE;
            else if (button == btnAttend4) state = ATTEND_TYPE.ABSENCE;
            else return;

            try
            {
                subject[week] = state;
                dgvAttend.SelectedCells[0].Value = GetStringState(subject, week);

            }
            catch (Exception ex)
            {
                dgvAttend.SelectedCells[0].Value = string.Empty;
            }
        }

        private void btnScoreSearch_Click(object sender, EventArgs e)
        {
            if (_students == null || _students.Count <= 0)
            {
                MessageBox.Show("검색할 학생이 없습니다.");
                return;
            }

            _studentScore = null;
            foreach (var stu in _students)
            {
                if (stu.Id == tbxScoreSearchId.Text)
                {
                    _studentScore = stu;
                    break;
                }
            }

            if (_studentScore == null)
            {
                lblScoreId.Text = string.Empty;
                lblScoreName.Text = string.Empty;
                dgvScore.Rows.Clear();

                MessageBox.Show("해당 학생을 찾을 수 없습니다.");
            }
            else
            {
                lblScoreId.Text = _studentScore.Id;
                lblScoreName.Text = _studentScore.Name;

                SetScoreState();
            }
        }

        private void SetScoreState()
        {
            dgvScore.Rows.Clear();
            foreach (Subject sub in _studentScore.Subjects)
            {
                int index = dgvScore.Rows.Add();
                dgvScore.Rows[index].Cells[0].Value = sub.Title;
                dgvScore.Rows[index].Cells[1].Value = sub.Score.ToString("F1");
            }
        }

        private void btnScoreSave_Click(object sender, EventArgs e)
        {
            if (_studentScore == null) return;
            if (dgvScore.Rows.Count <= 0) return;

            for (int i = 0; i < dgvScore.Rows.Count; i++)
            {
                if (double.TryParse(dgvScore.Rows[i].Cells[1].Value.ToString(), out double score))
                {
                    _studentScore.Subjects[i].Score = score;
                }
            }

            SetScoreState();
        }

        private void btnViewSearch_Click(object sender, EventArgs e)
        {
            if (_students == null || _students.Count <= 0)
            {
                MessageBox.Show("검색할 학생이 없습니다.");
                return;
            }

            _studentView = null;
            foreach (var stu in _students)
            {
                if (stu.Id == tbxViewSearchId.Text)
                {
                    _studentView = stu;
                    break;
                }
            }

            if (_studentView == null)
            {
                lblViewId.Text = string.Empty;
                lblViewName.Text = string.Empty;
                dgvView.Rows.Clear();
                lblViewAvg.Text = string.Empty;
                lblViewMax.Text = string.Empty;
                lblViewMin.Text = string.Empty;

                MessageBox.Show("해당 학생을 찾을 수 없습니다.");
            }
            else
            {
                lblViewId.Text = _studentView.Id;
                lblViewName.Text = _studentView.Name;

                SetViewState();
            }
        }

        private void SetViewState()
        {
            dgvView.Rows.Clear();

            foreach (Subject sub in _studentView.Subjects)
            {
                int index = dgvView.Rows.Add();

                dgvView.Rows[index].Cells[0].Value = sub.Title;
                dgvView.Rows[index].Cells[1].Value = sub.Score.ToString("F1");

                //int empty, attend, absence, late;
                sub.AttendState(out int empty, out int attend, out int absence, out int late);

                dgvView.Rows[index].Cells[2].Value = attend;
                dgvView.Rows[index].Cells[3].Value = late;
                dgvView.Rows[index].Cells[4].Value = absence;
            }

            Subject maxSubject = null;
            Subject minSubject = null;
            double avg;
            if (_studentAttend.MinMaxAvg(ref minSubject, ref maxSubject, out avg))
            {
                lblViewAvg.Text = avg.ToString("F1");
                lblViewMax.Text = maxSubject.Score.ToString("F1");
                lblViewMin.Text = minSubject.Score.ToString("F1");
            }
            else
            {
                lblViewAvg.Text = string.Empty;
                lblViewMax.Text = string.Empty;
                lblViewMin.Text = string.Empty;
            }
        }
    }
}
```