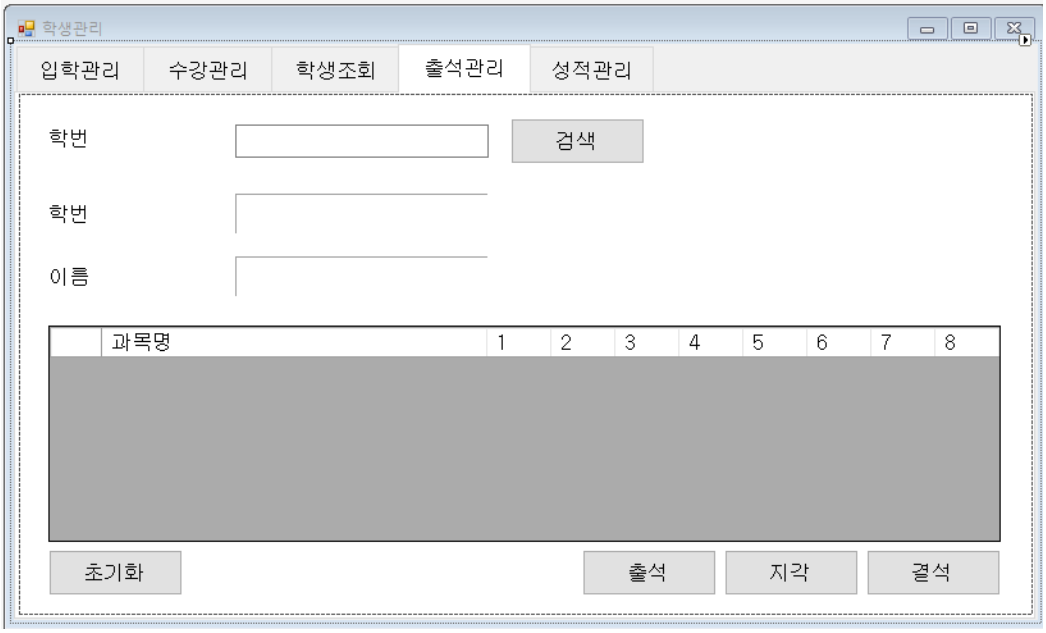
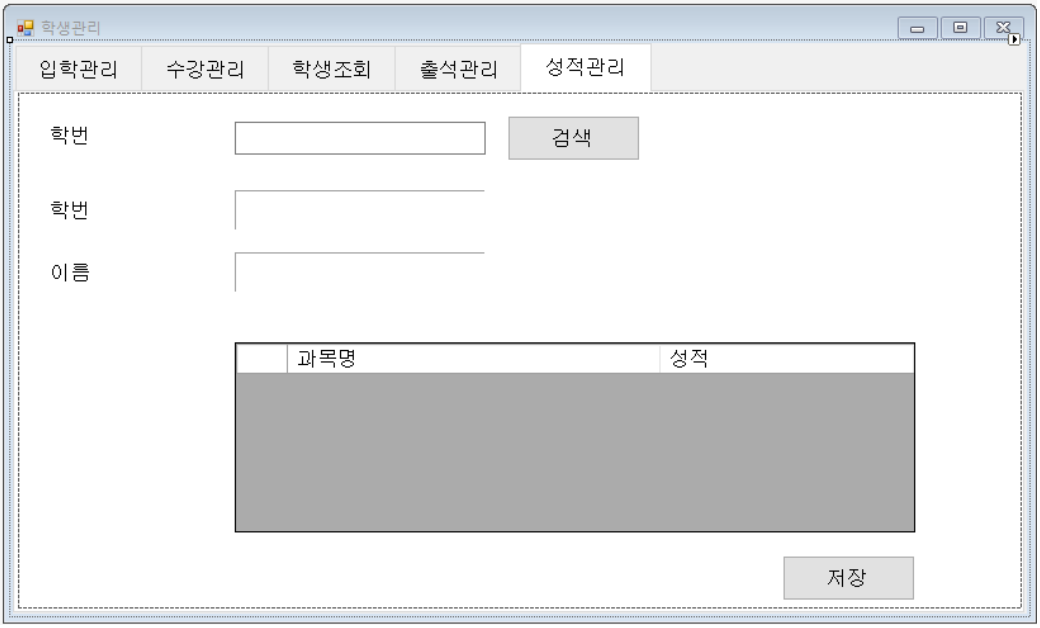
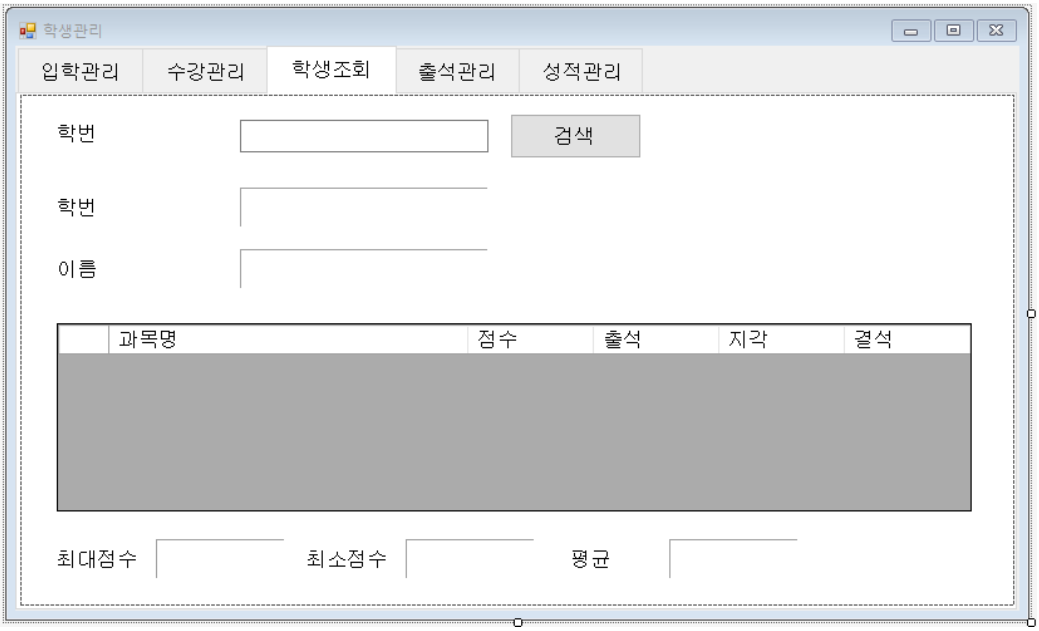
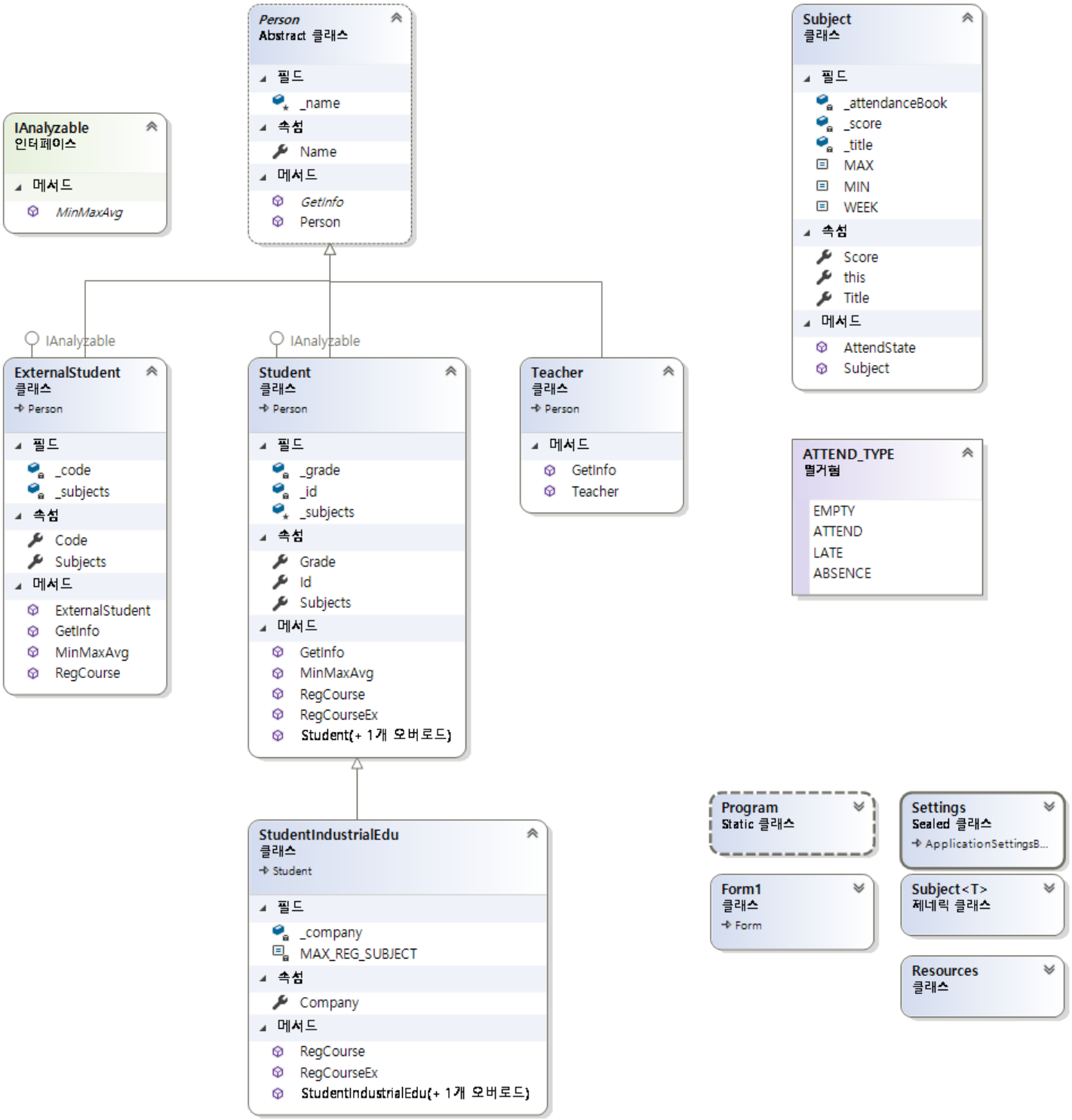
[화면]


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

[클래스 다이어그램]


IAnalyzable
인터페이스
메서드
MinMaxAvg
Person
Abstract 클래스
필드
_name
속성
Name
메서드
GetInfo
Person
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

```
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;
            }
        }

```

```
67          public void AttendState(out int empty, out int attend, out int absence, out int late)
68          {
69              empty = 0;
70              attend = 0;
71              absence = 0;
72              late = 0;
73
74              for (int i = 1; i <= Subject.WEEK; i++)
75              {
76                  ATTEND_TYPE state = this[i]; //인덱서를 이용해서 값을 가져오고 있음.
77
78                  if (state == ATTEND_TYPE.ATTEND)
79                      attend++;
80                  else if (state == ATTEND_TYPE.ABSENCE)
81                      absence++;
82                  else if (state == ATTEND_TYPE.LATE)
83                      late++;
84                  else
85                      empty++;
86              }
87          }
88
89          public Subject(string title)
90          {
91              _title = title;
92              _score = MIN;
93              _attendanceBook = new ATTEND_TYPE[WEEK];
94          }
95      }
96  }
```

ExternalStudent.cs

```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}";
        }

```

```
68          public bool MinMaxAvg(ref Subject minSubject, ref Subject maxSubject, out double avg)
69          {
70              avg = 0;
71
72              if (_subjects == null || _subjects.Count <= 0)
73              {
74                  return false;
75              }
76
77              double min = Subject.MAX;
78              double max = Subject.MIN;
79              double sum = 0;
80
81              foreach (var sub in _subjects)
82              {
83                  if (sub.Score < min)
84                  {
85                      min = sub.Score;
86                      minSubject = sub;
87                  }
88
89                  if (sub.Score > max)
90                  {
91                      max = sub.Score;
92                      maxSubject = sub;
93                  }
94
95                  sum += sub.Score;
96              }
97
98              avg = sum / _subjects.Count;
99
100             return true;
101         }
102     }
103 }
```

Student.cs

```
1   using ...
6
7   namespace School
8   {
9       class Student : Person, IAnalyzable
10      {
11          private string _id;//학번
12          public string Id
13          {
14              get
15              {
16                  return _id;
17              }
18          }
19
20          //private string _name; //이름
21          //public string Name
22          //{
23          //    get
24          //    {
25          //        return _name;
26          //    }
27          //}
28
29          private int _grade; //학년
30          public int Grade
31          {
32              get
33              {
34                  return _grade;
35              }
36
37              set
38              {
39                  _grade = value;
40              }
41          }
42
43          //private List<string> _subject; //수강과목
44          protected List<Subject> _subjects; //수강과목
45          public List<Subject> Subjects
46          {
47              get
48              {
49                  return _subjects;
50              }
51          }
52
53          public Student(string name, string id)
54              : base(name)
55          {
56              _name = name;
57              _id = id;
58              _grade = 1;
59              _subjects = new List<Subject>();
60          }
61
62          public Student(string name, string id, int grade)
63              : this(name, id)
64          {
65              Grade = grade;
66          }
67
```

```
68      public bool RegCourse(string subject)
69      {
70          bool result = false;
71
72          if (_subjects == null)
73          {
74              _subjects = new List<Subject>();
75          }
76
77          string searchSubject = null;
78          foreach(var sub in _subjects)
79          {
80              if(sub.Title == subject)
81              {
82                  searchSubject = sub.Title;
83                  break;
84              }
85          }
86
87          if(searchSubject == null)
88          {
89              _subjects.Add(new Subject(subject));
90              result = true;
91          }
92
93          return result;
94      }
95
96      public virtual bool RegCourseEx(string subject)
97      {
98          bool result = false;
99
100         if (_subjects == null)
101         {
102             _subjects = new List<Subject>();
103         }
104
105         string searchSubject = null;
106         foreach (var sub in _subjects)
107         {
108             if (sub.Title == subject)
109             {
110                 searchSubject = sub.Title;
111                 break;
112             }
113         }
114
115         if (searchSubject == null)
116         {
117             _subjects.Add(new Subject(subject));
118             result = true;
119         }
120
121         return result;
122     }
123
124     public override string GetInfo()
125     {
126         return $"{_id}-{_name}-{_grade}";
127     }
128
```

```
129         public bool MinMaxAvg(ref Subject minSubject, ref Subject maxSubject, out double avg)
130         {
131             avg = 0;
132
133             if(_subjects == null || _subjects.Count <= 0)
134             {
135                 return false;
136             }
137
138             double min = Subject.MAX;
139             double max = Subject.MIN;
140             double sum = 0;
141
142             foreach(var sub in _subjects)
143             {
144                 if(sub.Score < min)
145                 {
146                     min = sub.Score;
147                     minSubject = sub;
148                 }
149
150                 if(sub.Score > max)
151                 {
152                     max = sub.Score;
153                     maxSubject = sub;
154                 }
155
156                 sum += sub.Score;
157             }
158
159             avg = sum / _subjects.Count;
160
161             return true;
162         }
163     }
164 }
```

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;
        }
```

```
67          public override bool RegCourseEx(string subject) //overriding
68          {
69              bool result = false;
70
71              if (_subjects == null)
72              {
73                  _subjects = new List<Subject>();
74              }
75
76              if (_subjects.Count < MAX_REG_SUBJECT)
77              {
78                  string searchSubject = null;
79                  foreach (var sub in _subjects)
80                  {
81                      if (sub.Title == subject)
82                      {
83                          searchSubject = sub.Title;
84                          break;
85                      }
86                  }
87
88                  if (searchSubject == null)
89                  {
90                      _subjects.Add(new Subject(subject));
91                      result = true;
92                  }
93              }
94              return result;
95          }
96      }
97  }
```

Form1.cs

```
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);
```

```
84
85              lblEntranceResult.Text
86                  = $"등록학생의 정보입니다.\r\n학번 : {student.Id}\r\n이름 : {student.Name}\r\n학년 : {student.Grade}\r\n";
87
88              StudentIndustrialEdu studentIE = student as StudentIndustrialEdu;
89              if (studentIE != null)
90              {
91                  lblEntranceResult.Text += $"[산학과정] 소속회사:{studentIE.Company}";
92              }
93          }
94
95          private void btnRegCourseSearch_Click(object sender, EventArgs e)
96          {
97              if (_students == null || _students.Count <= 0)
98              {
99                  MessageBox.Show("검색할 학생이 없습니다.");
100                 return;
101             }
102
103             _studentRegCourse = null;
104             foreach (var stu in _students)
105             {
106                 if (stu.Id == tbxRegCourseSearchId.Text)
107                 {
108                     _studentRegCourse = stu;
109                     break;
110                 }
111             }
112
113             if (_studentRegCourse == null)
114             {
115                 lblRegCourseId.Text = string.Empty;
116                 lblRegCourseName.Text = string.Empty;
117                 lbxRegCourse.Items.Clear();
118
119                 MessageBox.Show("해당 학생을 찾을 수 없습니다.");
120             }
121             else
122             {
123                 lblRegCourseId.Text = _studentRegCourse.Id;
124                 lblRegCourseName.Text = _studentRegCourse.Name;
125                 lbxRegCourse.Items.Clear();
126                 foreach (var sub in _studentRegCourse.Subjects)
127                 {
128                     lbxRegCourse.Items.Add(sub);
129                 }
130             }
131         }
132
133         private void btnRegCourseHiding_Click(object sender, EventArgs e)
134         {
135             if (_studentRegCourse != null)
136             {
137                 if (_studentRegCourse.RegCourse(tbxRegCourseName.Text))//hiding
138                 {
139                     lbxRegCourse.Items.Add(tbxRegCourseName.Text);
140                 }
141                 else
142                 {
143                     MessageBox.Show("등록실패");
144                 }
145             }
146         }
147
```

```
148         private void btnRegCourseOverriding_Click(object sender, EventArgs e)
149         {
150             if (_studentRegCourse != null)
151             {
152                 if (_studentRegCourse.RegCourseEx(tbxRegCourseName.Text))//overridng**
153                 {
154                     lbxRegCourse.Items.Add(tbxRegCourseName.Text);
155                 }
156                 else
157                 {
158                     MessageBox.Show("등록실패");
159                 }
160             }
161         }
162
163         private void btnAttendSearch_Click(object sender, EventArgs e)
164         {
165             if (_students == null || _students.Count <= 0)
166             {
167                 MessageBox.Show("검색할 학생이 없습니다.");
168                 return;
169             }
170
171             _studentAttend = null;
172             foreach (var stu in _students)
173             {
174                 if (stu.Id == tbxAttendSearchId.Text)
175                 {
176                     _studentAttend = stu;
177                     break;
178                 }
179             }
180
181             if (_studentAttend == null)
182             {
183                 lblAttendId.Text = string.Empty;
184                 lblAttendName.Text = string.Empty;
185                 dgvAttend.Rows.Clear();
186
187                 MessageBox.Show("해당 학생을 찾을 수 없습니다.");
188             }
189             else
190             {
191                 lblAttendId.Text = _studentAttend.Id;
192                 lblAttendName.Text = _studentAttend.Name;
193
194                 SetAttendState();
195             }
196         }
197
198         private void SetAttendState()
199         {
200             dgvAttend.Rows.Clear();
201
202             foreach (Subject sub in _studentAttend.Subjects)
203             {
204                 int index = dgvAttend.Rows.Add();
205                 dgvAttend.Rows[index].Cells[0].Value = sub.Title;
206
207                 for (int i = 1; i <= Subject.WEEK; i++)
208                 {
209                     dgvAttend.Rows[index].Cells[i].Value = GetStringState(sub, i);
210                 }
211             }
212         }
213
```

```
214         private string GetStringState(Subject sub, int week)
215         {
216             if (1 > week || week > Subject.WEEK)
217             {
218                 return string.Empty;
219             }
220
221             ATTEND_TYPE state = sub[week];
222             string display = string.Empty;
223
224             switch (state)
225             {
226                 case ATTEND_TYPE.ABSENCE:
227                     display = "결";
228                     break;
229                 case ATTEND_TYPE.ATTEND:
230                     display = "출";
231                     break;
232                 case ATTEND_TYPE.LATE:
233                     display = "지";
234                     break;
235             }
236
237             return display;
238         }
239
240         private void btnAttend_Click(object sender, EventArgs e)
241         {
242             Button button = sender as Button;
243
244             if (_studentAttend == null) return;
245             if (dgvAttend.SelectedCells.Count != 1) return;
246             if (dgvAttend.SelectedCells[0].ColumnIndex < 1) return;
247
248             int week = dgvAttend.SelectedCells[0].ColumnIndex;
249             int sub_index = dgvAttend.SelectedCells[0].RowIndex;
250             Subject subject = _studentAttend.Subjects[sub_index];
251             ATTEND_TYPE state = ATTEND_TYPE.EMPTY;
252
253             if (button == btnAttend1) state = ATTEND_TYPE.EMPTY;
254             else if (button == btnAttend2) state = ATTEND_TYPE.ATTEND;
255             else if (button == btnAttend3) state = ATTEND_TYPE.LATE;
256             else if (button == btnAttend4) state = ATTEND_TYPE.ABSENCE;
257             else return;
258
259             try
260             {
261                 subject[week] = state;
262                 dgvAttend.SelectedCells[0].Value = GetStringState(subject, week);
263
264             }
265             catch (Exception ex)
266             {
267                 dgvAttend.SelectedCells[0].Value = string.Empty;
268             }
269         }
270
271         private void btnScoreSearch_Click(object sender, EventArgs e)
272         {
273             if (_students == null || _students.Count <= 0)
274             {
275                 MessageBox.Show("검색할 학생이 없습니다.");
276                 return;
277             }
278
279             _studentScore = null;
280             foreach (var stu in _students)
281             {
282                 if (stu.Id == tbxScoreSearchId.Text)
283                 {
284                     _studentScore = stu;
285                     break;
286                 }
287             }
288
```

```
289             if (_studentScore == null)
290             {
291                 lblScoreId.Text = string.Empty;
292                 lblScoreName.Text = string.Empty;
293                 dgvScore.Rows.Clear();
294 
295                 MessageBox.Show("해당 학생을 찾을 수 없습니다.");
296             }
297             else
298             {
299                 lblScoreId.Text = _studentScore.Id;
300                 lblScoreName.Text = _studentScore.Name;
301 
302                 SetScoreState();
303             }
304         }
305 
306         private void SetScoreState()
307         {
308             dgvScore.Rows.Clear();
309             foreach (Subject sub in _studentScore.Subjects)
310             {
311                 int index = dgvScore.Rows.Add();
312                 dgvScore.Rows[index].Cells[0].Value = sub.Title;
313                 dgvScore.Rows[index].Cells[1].Value = sub.Score.ToString("F1");
314             }
315         }
316 
317         private void btnScoreSave_Click(object sender, EventArgs e)
318         {
319             if (_studentScore == null) return;
320             if (dgvScore.Rows.Count <= 0) return;
321 
322             for (int i = 0; i < dgvScore.Rows.Count; i++)
323             {
324                 if (double.TryParse(dgvScore.Rows[i].Cells[1].Value.ToString(), out double score))
325                 {
326                     _studentScore.Subjects[i].Score = score;
327                 }
328             }
329 
330             SetScoreState();
331         }
332 
333         private void btnViewSearch_Click(object sender, EventArgs e)
334         {
335             if (_students == null || _students.Count <= 0)
336             {
337                 MessageBox.Show("검색할 학생이 없습니다.");
338                 return;
339             }
340 
341             _studentView = null;
342             foreach (var stu in _students)
343             {
344                 if (stu.Id == tbxViewSearchId.Text)
345                 {
346                     _studentView = stu;
347                     break;
348                 }
349             }
350 
351             if (_studentView == null)
352             {
353                 lblViewId.Text = string.Empty;
354                 lblViewName.Text = string.Empty;
355                 dgvView.Rows.Clear();
356                 lblViewAvg.Text = string.Empty;
357                 lblViewMax.Text = string.Empty;
358                 lblViewMin.Text = string.Empty;
359 
360                 MessageBox.Show("해당 학생을 찾을 수 없습니다.");
361             }
```

```
362             else
363             {
364                 lblViewId.Text = _studentView.Id;
365                 lblViewName.Text = _studentView.Name;
366
367                 SetViewState();
368             }
369         }
370
371         private void SetViewState()
372         {
373             dgvView.Rows.Clear();
374
375             foreach (Subject sub in _studentView.Subjects)
376             {
377                 int index = dgvView.Rows.Add();
378
379                 dgvView.Rows[index].Cells[0].Value = sub.Title;
380                 dgvView.Rows[index].Cells[1].Value = sub.Score.ToString("F1");
381
382                 //int empty, attend, absence, late;
383                 sub.AttendState(out int empty, out int attend, out int absence, out int late);
384
385                 dgvView.Rows[index].Cells[2].Value = attend;
386                 dgvView.Rows[index].Cells[3].Value = late;
387                 dgvView.Rows[index].Cells[4].Value = absence;
388             }
389
390             Subject maxSubject = null;
391             Subject minSubject = null;
392             double avg;
393             if (_studentAttend.MinMaxAvg(ref minSubject, ref maxSubject, out avg))
394             {
395                 lblViewAvg.Text = avg.ToString("F1");
396                 lblViewMax.Text = maxSubject.Score.ToString("F1");
397                 lblViewMin.Text = minSubject.Score.ToString("F1");
398             }
399             else
400             {
401                 lblViewAvg.Text = string.Empty;
402                 lblViewMax.Text = string.Empty;
403                 lblViewMin.Text = string.Empty;
404             }
405         }
406     }
407 }
```