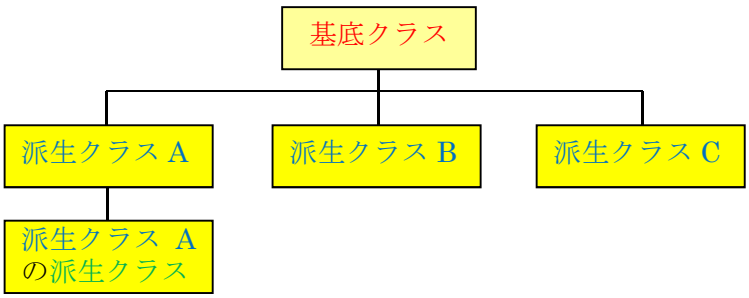
2019 年 7 月 4 日（木）実施

クラスの継承

オブジェクト指向プログラミングの基本的な属性として，親クラスのメンバを再利用，拡張，または変更する子クラスを定義することが出来る。 メンバの再利用を**継承**と呼び，継承元となるクラスを**基底クラス**と呼ぶ。また，基底クラスのメンバを継承するクラスを，**派生クラス**と呼ぶ。なお，メンバの中でコンストラクタは継承されない。

C#言語では，Java 言語と同様に単一継承のみがサポートされている。これはある基底クラスから複数の派生クラスを作ることは可能であるが，ある派生クラスを複数の基底クラスから作ることは出来ない。（下図）



派生クラスの構成は一般に次の様になる。

```
修飾子 class 派生クラス名 : 基底クラス名
{
    フィールド
    修飾子 コンストラクタ名(=派生クラス名)( 引数リスト )
    {
        初期化処理
    }
    修飾子 データ型 メソッド名 ( 引数リスト )
    {
        処理
    }
}
```

オーバーライド

派生クラスで基底クラスから継承したメソッドを再定義することを**オーバーライド**（override）と呼ぶ。C#言語では，メソッドをオーバーライドするためには，基底クラスのメソッドに virtual のキーワードを付けておいて，派生クラスでオーバーライドするメソッドには override キーワードを付ける必要がある。

カプセル化

C#言語では，アクセス修飾子によって，全てのクラスおよびクラスメンバに対して，他のクラスに提供するアクセスレベルを指定出来る。主なアクセス修飾子は次の様な働きをする。

・public　制限は無く，どこからでもアクセス出来る。
・private　同じクラス内でのみアクセス出来る。
・protected　同じクラス内または派生クラス内でのみアクセス出来る。
・internal　同じアセンブリ（アプリケーションのビルドの基本単位）内でのみアクセス出来る。

これらのアクセス修飾子の使い分けによって，外部からの影響を防ぐメンバを隠し，外部からの操作を許すメンバを公開してアクセス可能にすることを**カプセル化**と呼ぶ。

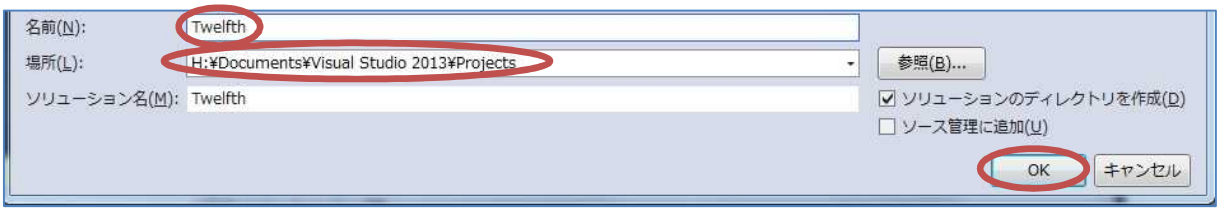
本日の課題

クラスの継承，メソッドのオーバーライド，及びカプセル化について，実例を通じて学ぶ。

手順

1）プロジェクトの作成

Visual Studio 2013 を起動したら，［ファイル］→［新規作成］→［プロジェクト］と辿って，プロジェクトを作成する。『新しいプロジェクト』ダイアログボックスでは，プログラミング言語を『Visual C#』，プロジェクトテンプレートとしては，『Windows フォームアプリケーション』を選択し，『名前』を「Twelfth」に書き換え，『場所』が「H:¥Documents¥Visual Studio 2013¥Projects」となっていることを確認してから『OK』を押す（詳細は第 1 回の教材を参照）。

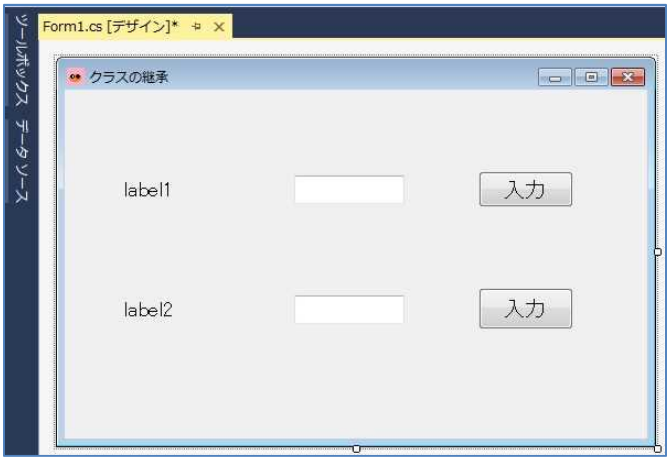


2）コントロールの配置及びフォームの作成

今後，フォーム上に配置するコントロールのプロパティのフォントサイズは全て 14 ポイントに変更するものとする。

Form1 上にラベルを 2 つ，テキストボックスを 2 つ，ボタンを 2 つ貼り付ける。それぞれのプロパティは次の様に設定する。

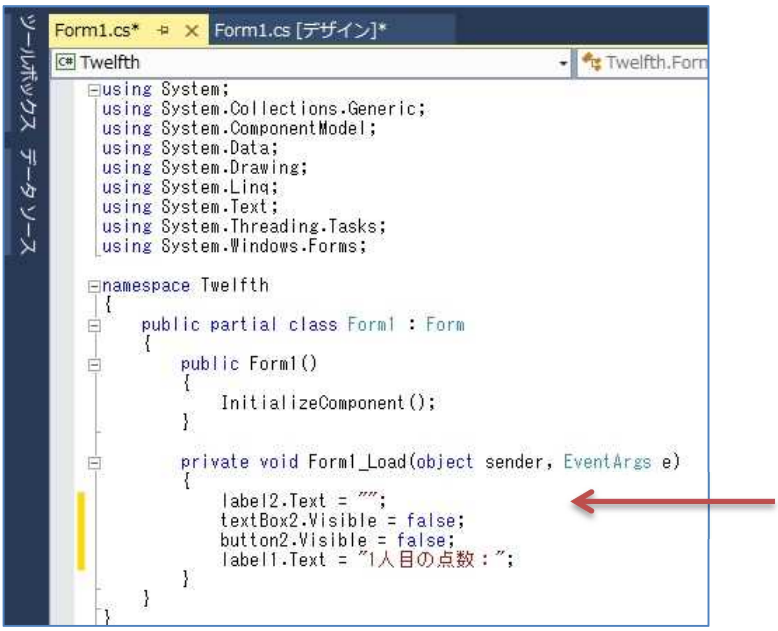
【Form1】Text　「**クラスの継承**」
　　Icon　自作のアイコン
【button1】Text　「**入力**」
【button2】Text　「**入力**」



3) コーディング

Form1 のフォームデザイナー上でコントロールが貼られていない箇所をダブルクリックして Form1.cs のプログラムのソースコードを表示する。Form1_Load メソッドのブロック内に Form1 が読み込まれた際の処理として，『label2』の『Text』プロパティに空の文字列を設定する処理，『textBox2』及び『button2』の『Visible』プロパティに「false」を設定して非表示にする処理及び『label1』の『Text』プロパティに初期の文字列を設定する処理（赤枠の部分）を記述する。

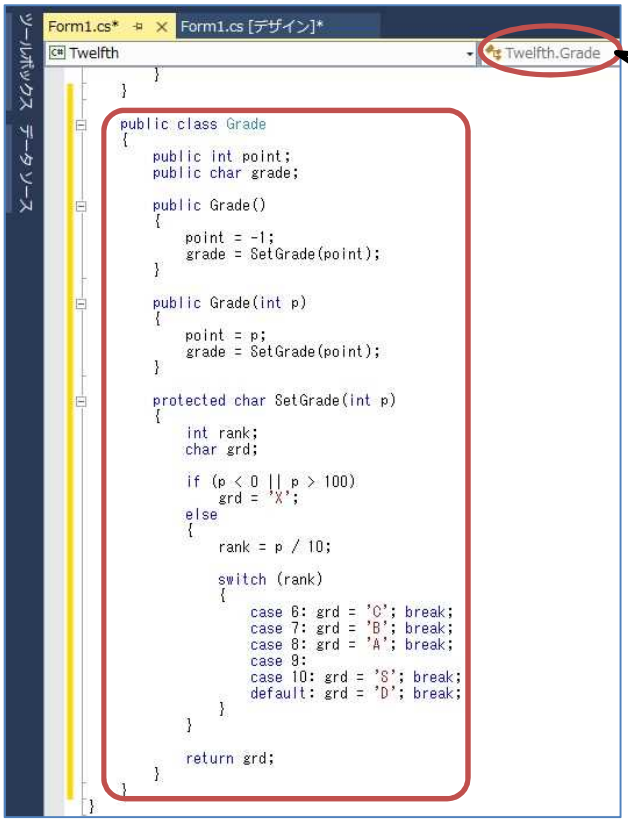
```
        private void Form1_Load(object sender, EventArgs e)
        {
            label2.Text = "";
            textBox2.Visible = false;
            button2.Visible = false;
            label1.Text = "1 人目の点数：";
        }
```



Form1 クラスより下に Grade クラス，Grade2 クラス及び Grade3 クラスの定義を記述する。ここで，Grade2 クラスは Grade クラスの派生クラス，Grade3 クラスは Grade2 クラスの派生クラスである。SetGrade メソッドは protected なので，Form1 クラスからはアクセス出来ない。

```
public class Grade
{
    public int point;
    public char grade;

    public Grade()
    {
        point = -1;
        grade = SetGrade(point);
    }
```

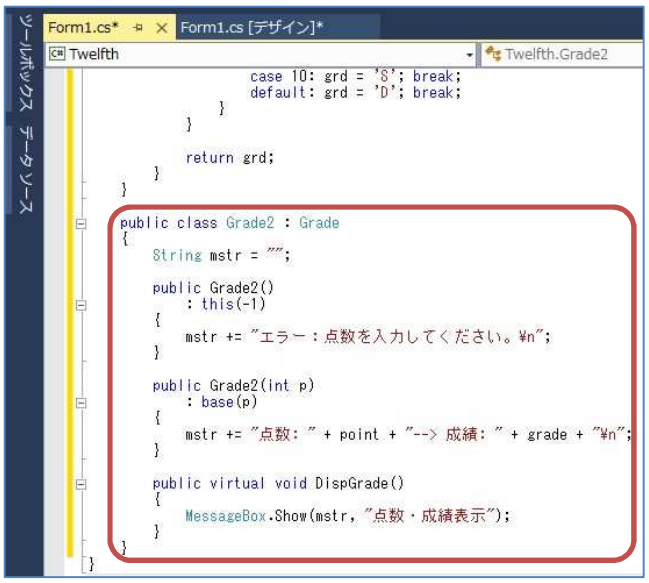
```
        public Grade(int p)
        {
            point = p;
            grade = SetGrade(point);
        }

        protected char SetGrade(int p)
        {
            int rank;
            char grd;

            if (p < 0 || p > 100)
                grd = 'X';
            else
            {
                rank = p / 10;

                switch (rank)
                {
                    case 6: grd = 'C'; break;
                    case 7: grd = 'B'; break;
                    case 8: grd = 'A'; break;
                    case 9:
                    case 10: grd = 'S'; break;
                    default: grd = 'D'; break;
                }
            }

            return grd;
        }
    }
```



どのクラスを記述
中かはここに注目

Grade()及び Grade(int p)の様に 2 個のコンストラクタが定義されている場合には，インスタンスを生成する際に引数無しとするか，int 型の引数を 1 個受け渡すかによって，引数の並びが同一の方のコンストラクタが読み込まれて初期化処理が行われる。

```
public class Grade2 : Grade
{
    String mstr = "";

    public Grade2()
        : this(-1)
    {
        mstr += "エラー：点数を入力してください。¥n";
    }

    public Grade2(int p)
        : base(p)
    {
        mstr += "点数: " + point + "--> 成績: " + grade + "¥n";
    }

    public virtual void DispGrade()
    {
        MessageBox.Show(mstr, "点数・成績表示");
    }
}
```



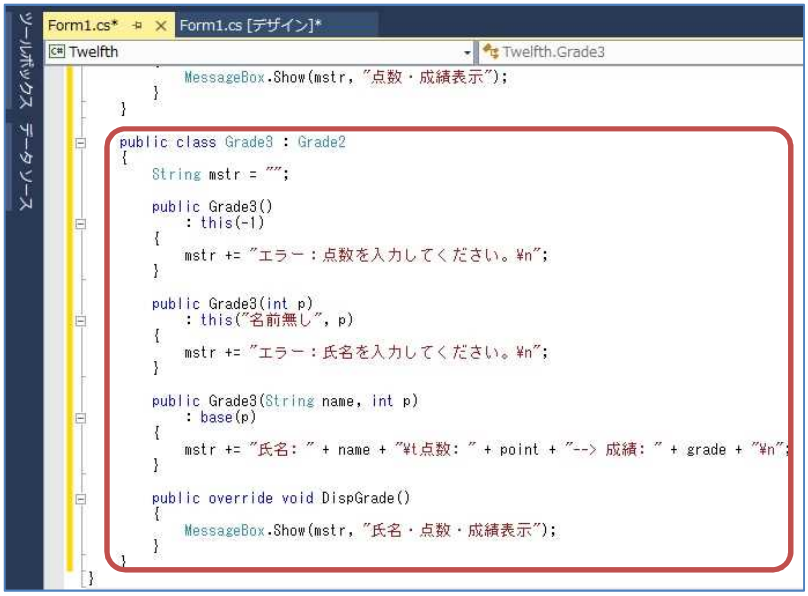
「: this(-1)」はこのクラスの引数が 1 個のコンストラクタに-1 を渡して実行する。

「: base(p)」は基底クラス（この場合は Grade クラス）の引数が 1 個のコンストラクタに p を渡して実行する。その結果，point 及び grade に値が代入される。

DispGrade メソッドには virtual キーワードを付けて，オーバーライド可能にしている。

```
public class Grade3 : Grade2
{
    String mstr = "";

    public Grade3()
        : this(-1)
    {
        mstr += "エラー：点数を入力してください。¥n";
    }

    public Grade3(int p)
        : this("名前無し", p)
    {
        mstr += "エラー：氏名を入力してください。¥n";
    }
```

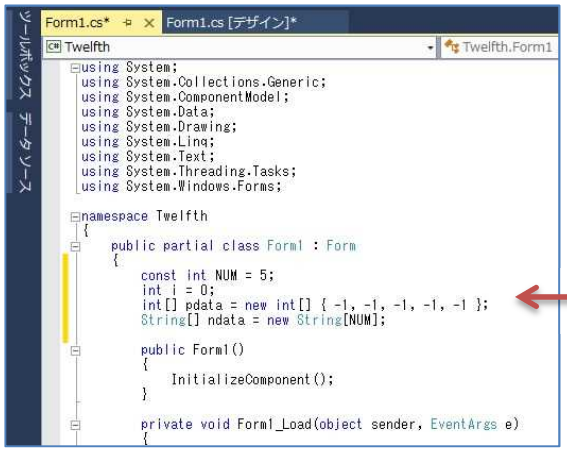
```
        public Grade3(String name, int p)
            : base(p)
        {
            mstr += "氏名: " + name + "¥t 点数: " + point + "--> 成績: " + grade + "¥n";
        }

        public override void DispGrade()
        {
            MessageBox.Show(mstr, "氏名・点数・成績表示");
        }
    }
```



DispGrade メソッドには override キーワードを付けて，オーバーライドしている。

Form1 のフォームデザイナー上で『button1』及び『button2』をダブルクリックして，Form1.cs のプログラムのソースコードを表示する。先ず，クラス全体に適用可能な定数，変数及び配列の宣言を Form1 クラスの冒頭に記述する。

```
const int NUM = 5;
int i = 0;
int[] pdata = new int[] { -1, -1, -1, -1, -1 };
String[] ndata = new String[NUM];
```



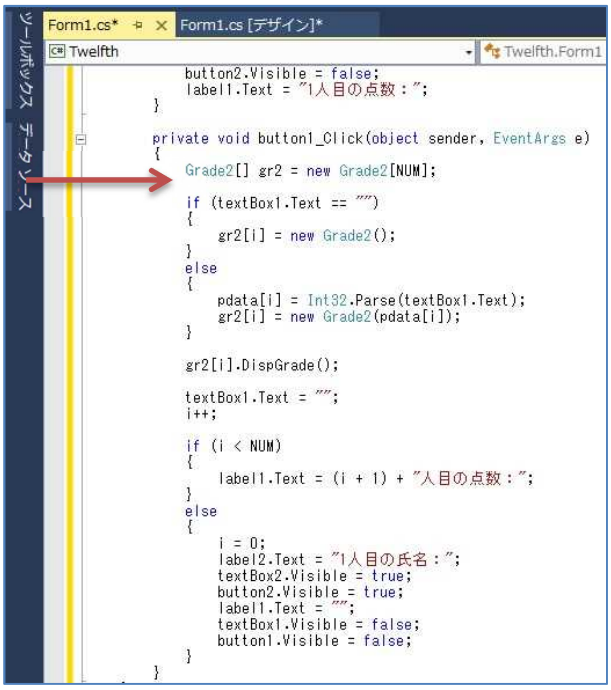
ここで，pdata は点数データを格納する配列，ndata は氏名データを格納する配列である。

次に，button1_Click メソッド及び button2_Click メソッドのブロック内にそれぞれのボタンがクリックされた際の処理（赤枠の部分）を記述していく。

```
private void button1_Click(object sender, EventArgs e)
{
    Grade2[] gr2 = new Grade2[NUM];                 // インスタンスの配列 gr2

    if (textBox1.Text == "")
    {
        gr2[i] = new Grade2();                      // 配列要素もインスタンス化
    }
    else
    {
        pdata[i] = Int32.Parse(textBox1.Text);
        gr2[i] = new Grade2(pdata[i]);              // 配列要素もインスタンス化
    }

    gr2[i].DispGrade();

    textBox1.Text = "";
    i++;

    if (i < NUM)
    {
        label1.Text = (i + 1) + "人目の点数：";
    }
    else
    {
        i = 0;
        label2.Text = "1 人目の氏名：";
        textBox2.Visible = true;
        button2.Visible = true;
        label1.Text = "";
        textBox1.Visible = false;
        button1.Visible = false;
    }
}
```

```
private void button2_Click(object sender, EventArgs e)
{
    Grade3[] gr3 = new Grade3[NUM];                 // インスタンスの配列 gr3

    if (textBox2.Text == "")
    {
        gr3[i] = new Grade3(pdata[i]);              // 配列要素もインスタンス化
    }
    else
    {
        ndata[i] = textBox2.Text;
        gr3[i] = new Grade3(ndata[i], pdata[i]);    // 配列要素もインスタンス化
    }

    gr3[i].DispGrade();
```

```
            textBox2.Text = "";
            i++;

            if (i < NUM)
            {
                label2.Text = (i + 1) + "人目の氏名：";
            }
            else
            {
                i = 0;
                label1.Text = "1 人目の点数：";
                textBox1.Visible = true;
                button1.Visible = true;
                label2.Text = "";
                textBox2.Visible = false;
                button2.Visible = false;
            }
        }
```



```
        button2.Visible = false;
        label1.Text = "1人目の点数：";
    }

    private void button1_Click(object sender, EventArgs e)
    {
        Grade2[] gr2 = new Grade2[NUM];

        if (textBox1.Text == "")
        {
            gr2[i] = new Grade2();
        }
        else
        {
            pdata[i] = Int32.Parse(textBox1.Text);
            gr2[i] = new Grade2(pdata[i]);
        }

        gr2[i].DispGrade();

        textBox1.Text = "";
        i++;

        if (i < NUM)
        {
            label1.Text = (i + 1) + "人目の点数：";
        }
        else
        {
            i = 0;
            label2.Text = "1人目の氏名：";
            textBox2.Visible = true;
            button2.Visible = true;
            label1.Text = "";
            textBox1.Visible = false;
            button1.Visible = false;
        }
    }
```

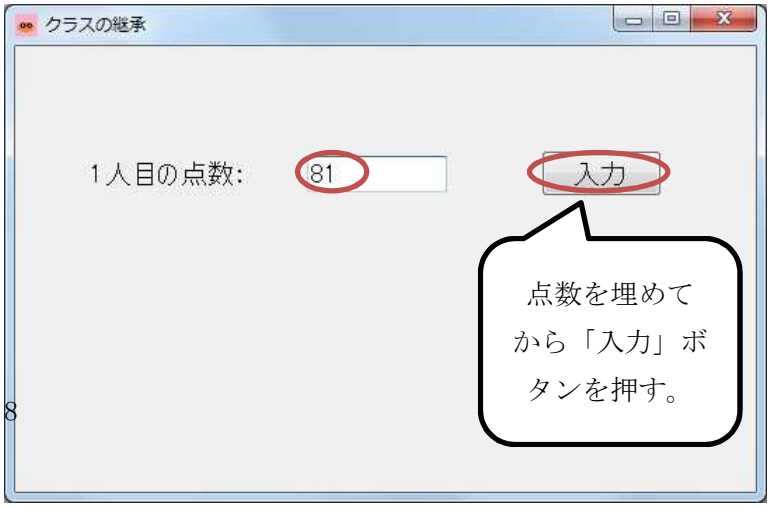
```
            textBox1.Visible = false;
            button1.Visible = false;
        }
    }

    private void button2_Click(object sender, EventArgs e)
    {
        Grade3[] gr3 = new Grade3[NUM];

        if (textBox2.Text == "")
        {
            gr3[i] = new Grade3(pdata[i]);
        }
        else
        {
            ndata[i] = textBox2.Text;
            gr3[i] = new Grade3(ndata[i], pdata[i]);
        }

        gr3[i].DispGrade();

        textBox2.Text = "";
        i++;

        if (i < NUM)
        {
            label2.Text = (i + 1) + "人目の氏名：";
        }
        else
        {
            i = 0;
            label1.Text = "1人目の点数：";
            textBox1.Visible = true;
            button1.Visible = true;
            label2.Text = "";
            textBox2.Visible = false;
            button2.Visible = false;
        }
    }
}

public class Grade
```

4) プログラムの実行・最終確認

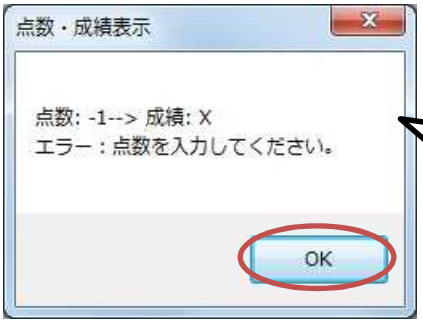
『すべてを保存』ボタンを押してから，『開始』ボタンを押して，プログラムを実行する。エラーが出ている場合には，修正してから保存，開始と進む。

今回のプログラムでは，点数をテキストボックスに入力して入力ボタンを押すという操作を 5 回繰り返すと氏名入力のラベル，テキストボックス及びボタンが現れる。

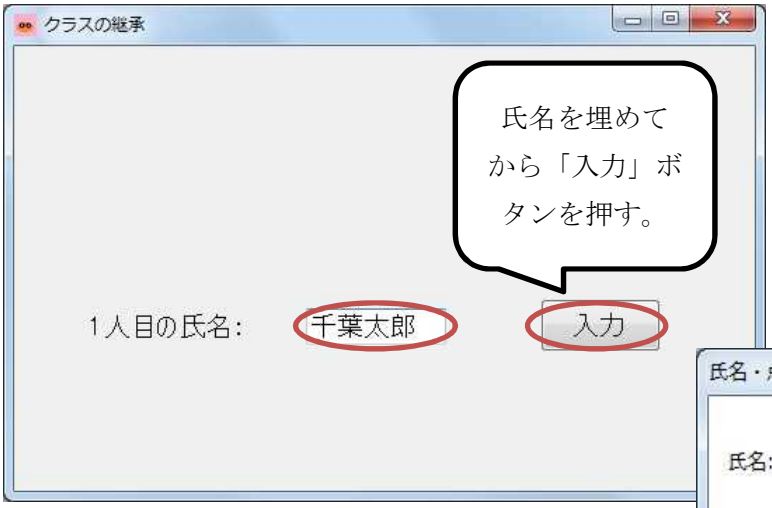
（次のページの図はその一例）



点数を埋めてから「入力」ボタンを押す。

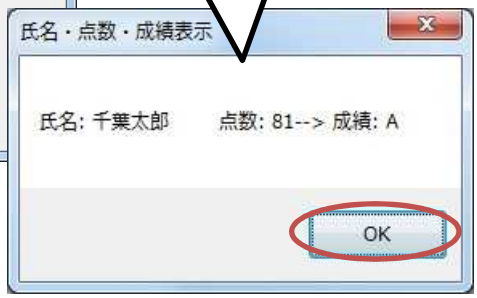


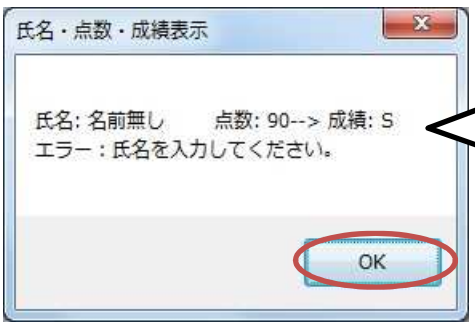
点数を埋めずに「入力」ボタンを押すとこうなる。



氏名を埋めてから「入力」ボタンを押す。

点数は pdata[i] によって引き継がれている。





氏名を埋めずに「入力」ボタンを押すとこうなる。

確認を終えたら，プログラムを終了する。

【ファイルが保存されている場所】　H:¥Documents¥Visual Studio 2013¥Projects¥Twelfth¥Twelfth

提出物：
1）フォームのデザインファイル Form1.Designer.cs をメールに添付して提出する。
2）フォームを含むソースファイル Form1.cs をメールに添付して提出する。
3）質問を記述したファイル Questions_12th.txt に解答を書き込んで保存し，メールに添付して提出する。