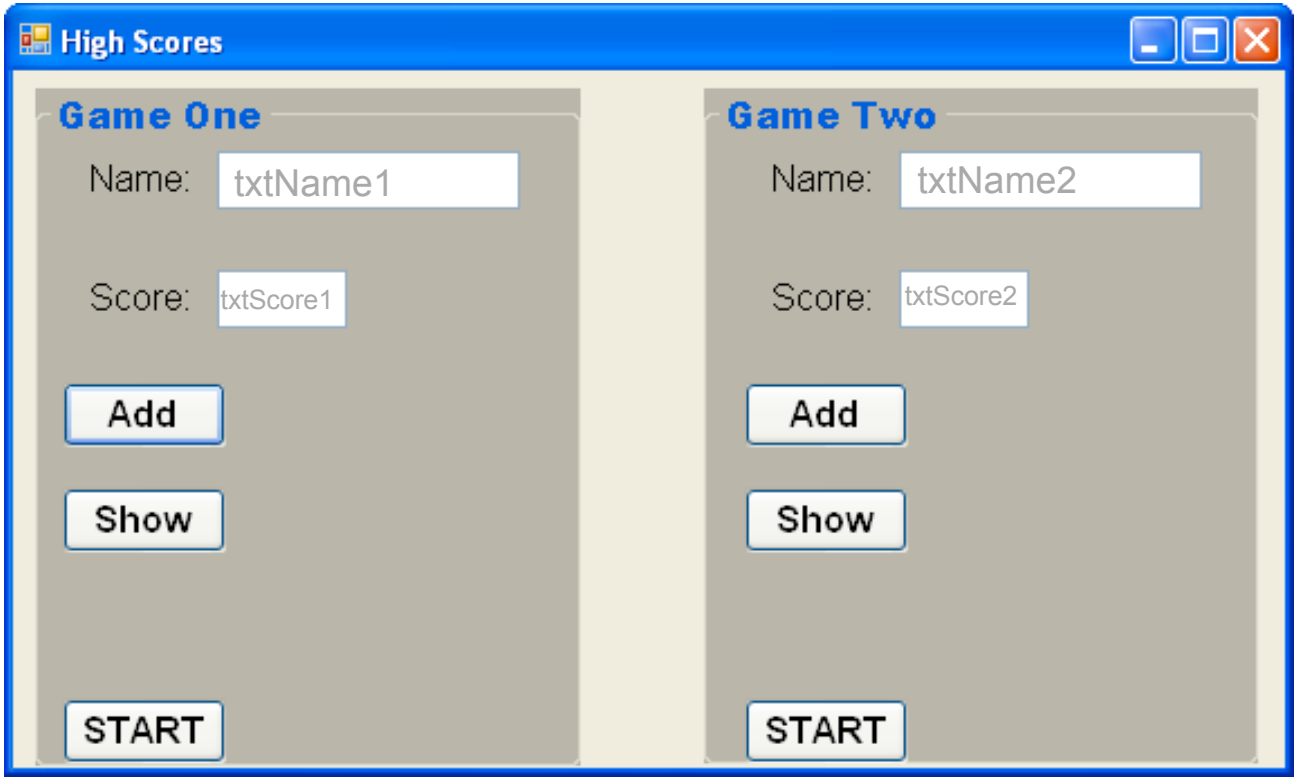
Objects and Classes Continued

Engineering 1D04, Teaching
Session 10

Recap: HighScores Example



recap: HighScores Example

```
public class HighScore
{
        private class hsRecord
        {
                public string name;
                public int score;
        }

        private const int maxElements = 10;
        private hsRecord[] hsArray = new hsRecord[maxElements];
        private int length;

        public bool add(string newName, int newScore)
        {
                . . .
        }

        public void show()
        {
                . . .
        }
}
```

no room to show this here

no room to show this here

recap: HighScores

- What about including a title for the name of the game at the time of instantiation?
- In general it is useful to be able to include initial processing for an object at the time of its instantiation.
- This is done through a *constructor*.

Constructors

- A constructor is simply a method that is executed automatically when the class is instantiated in an object.
- The name of the constructor method is exactly the same as the name of the class.
- The constructor method may also include parameters.

A Constructor for HighScores

- Consider our example. We want the constructor to include a string parameter for the title of the game.

Constructor Example

```
public class HighScore
{
    private class hsRecord
    {
        public string name;
        public int score;
    }

    const int maxElements = 10;
    private hsRecord[] hsArray = new hsRecord[maxElements];
    private int length = 0;
    private string result;
    private string hsTitle;

    public HighScore(string title)
    {
        hsTitle = title;
    }
    . . .
```

constructor

Constructor Example

- How do we use the constructor?

when we instantiate hs1 the constructor gets invoked (called)

string parameter currently set to "Group One"

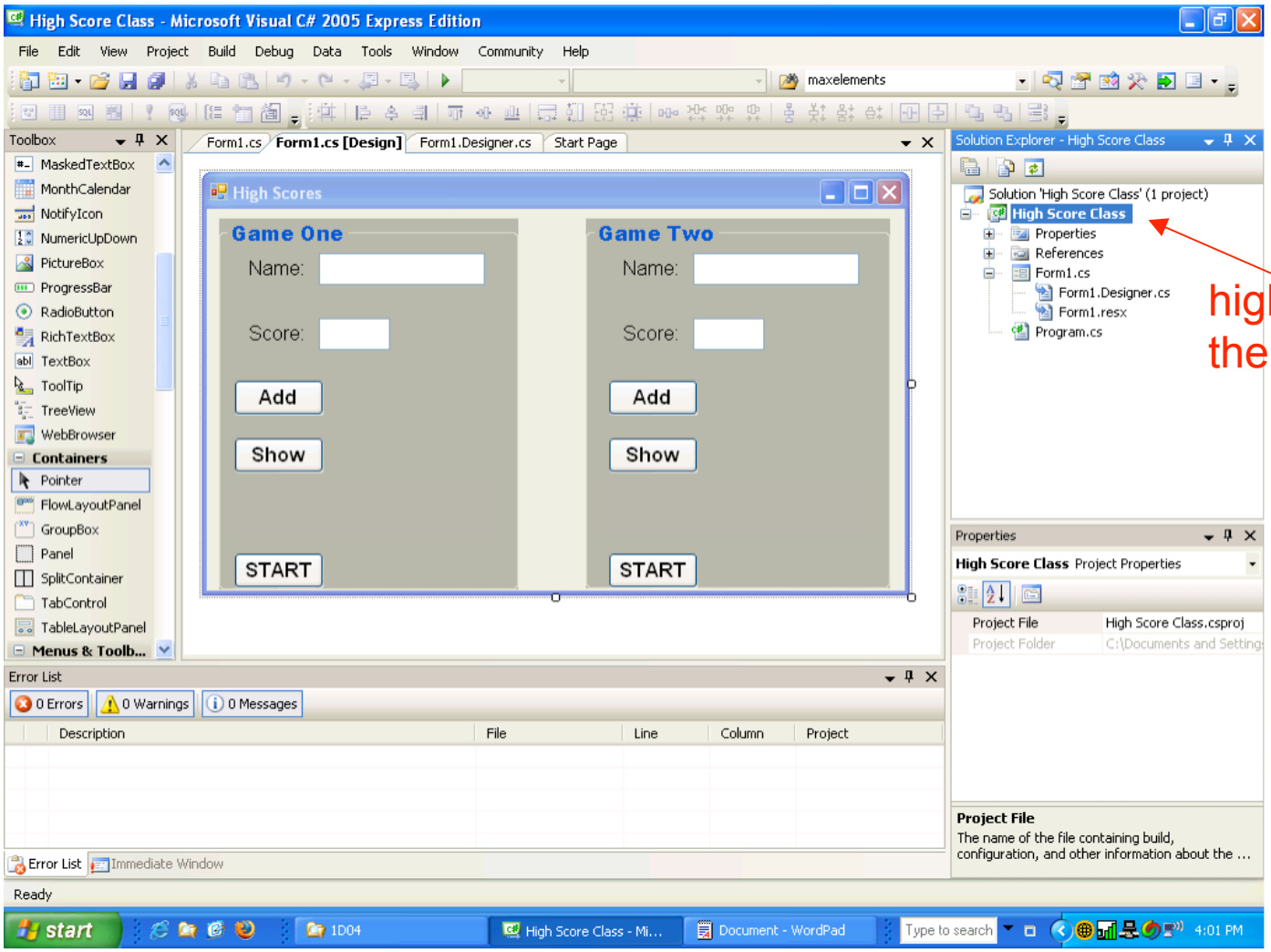
```
private void btnStart1_Click(object sender, EventArgs e)
{
    hs1 = new HighScore(gpGame1.Text);
    display(1, true);
    btnStart1.Visible = false;
    txtName1.Focus();
}
```

instantiate hs1

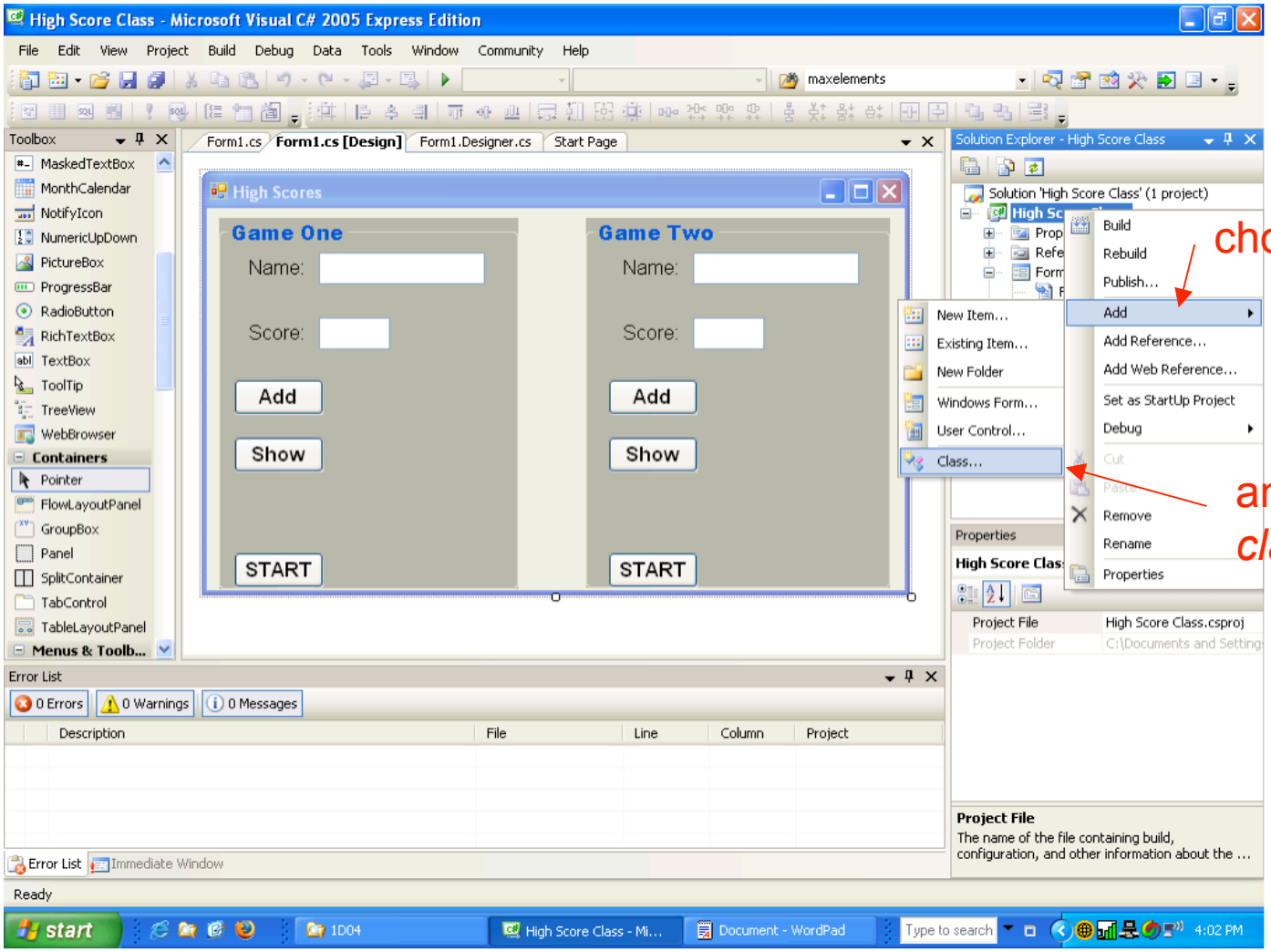
Multiple Classes in a Project

- In reasonably sized projects we typically need a number of classes, not just one.
- Each class is normally stored in its own source file.
- Visual Studio makes it easy to set up source files for classes.
- As an example, let us see how we can set up a source file for the class HighScore.

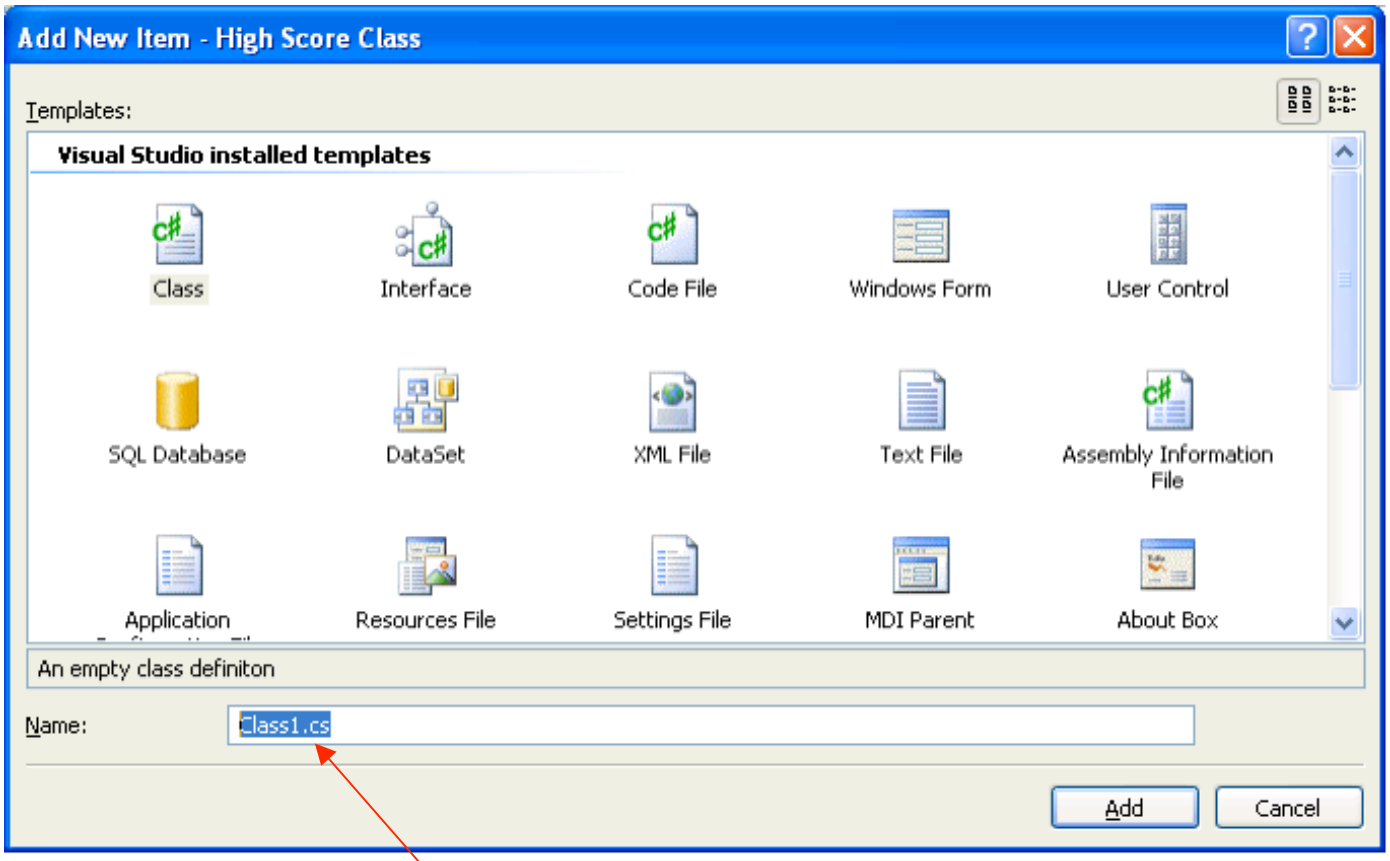
Source Files for Classes



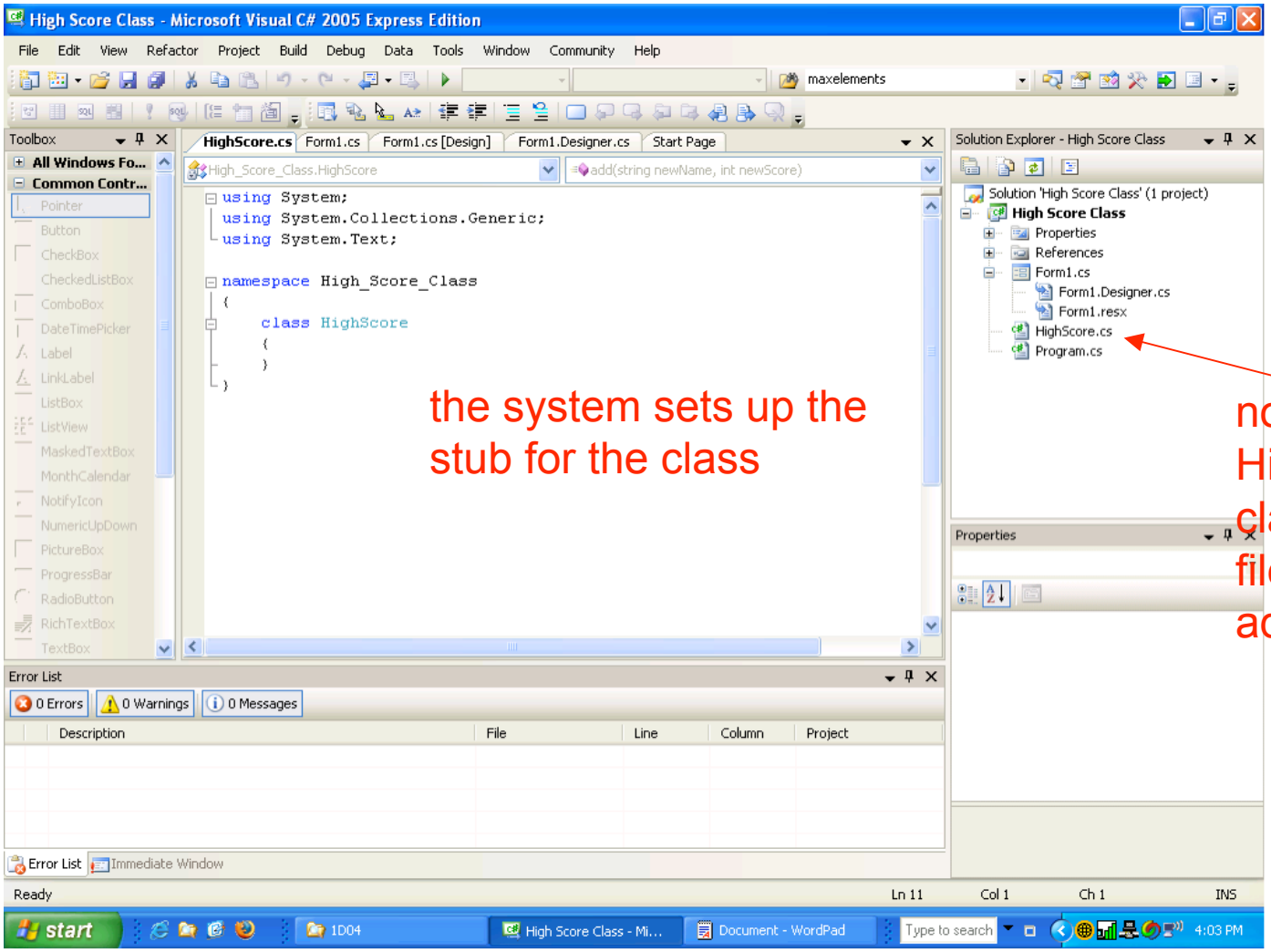
Source Files for Classes



Source Files for Classes



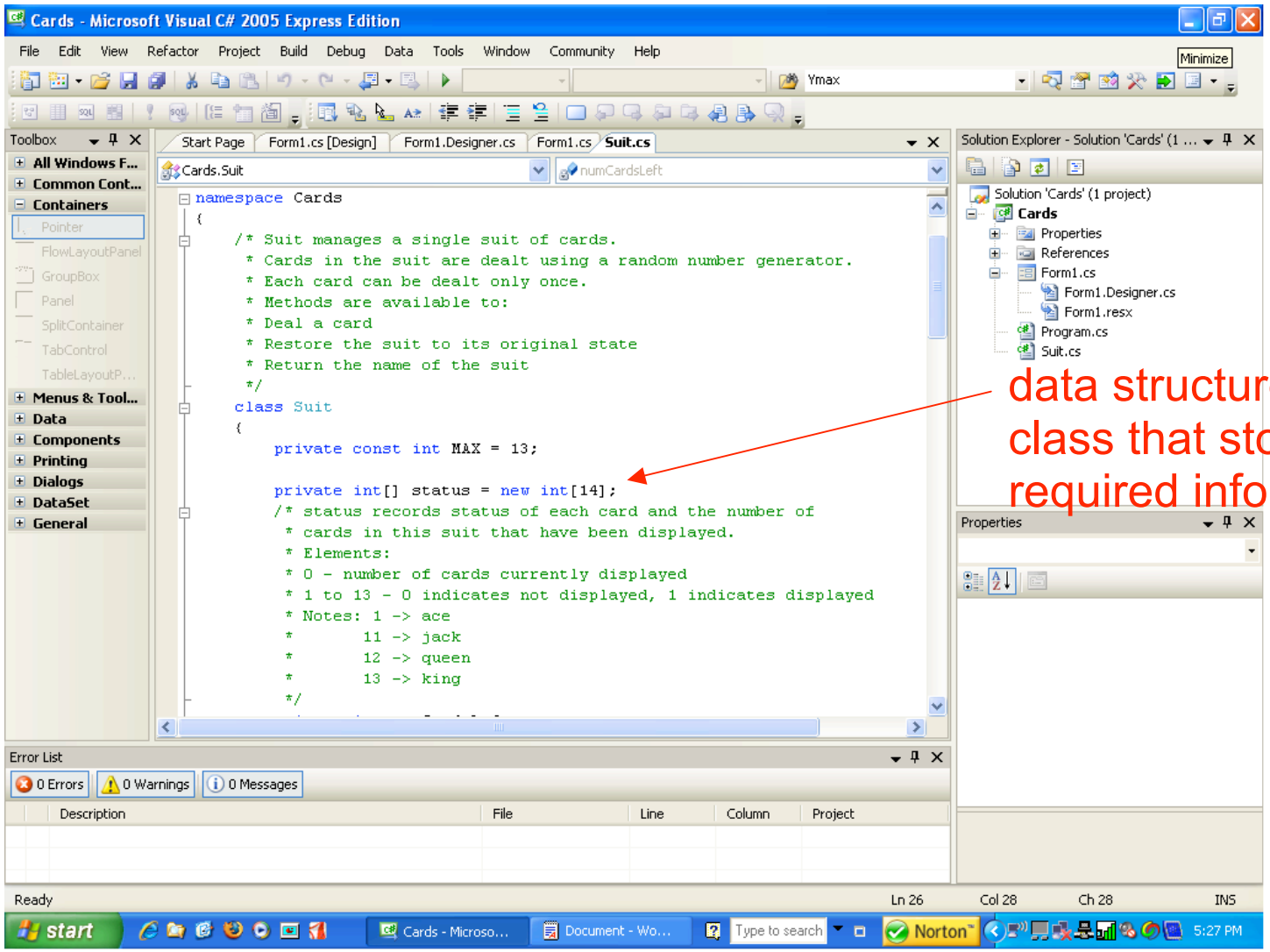
Source Files for Classes



Another Example

- Let's try another example
- Create a class that manages a suit of cards
  - We need to identify the suit (spades, etc)
  - We want to "deal" cards from the suit in random order
  - We need to be able to go back to the initial settings
  - We also need to be able to access the name of the suit - "Spades", "Hearts" etc.

Suit of Cards



data structure in the class that stores the required info

More Cards

```
private int numCardsLeft;
private string name;

private Random rand;
private Random start = new Random();

// Constructor - and gets name of suit and seed for random #
public Suit(string s, int seed)
{
    int r;
    name = s;
    r = start.Next(seed);
    rand = new Random(r);
    status[0] = 0;
    numCardsLeft = MAX;
}
```

class variables

lots of work to make it generate different numbers for each instance

More Cards

```
public string deal()
{
      string card;
      int i, c, n;

      if (numCardsLeft == 0) return null;
      else
      {
          n = rand.Next(1,numCardsLeft+1);
          //Find which card
          c = 0;
          for (i = 1; i <= n; )
          {
              c++;
              if (status[c] == 0) i++;
          }
          status[c] = 1;
          status[0]++;
          numCardsLeft--;
```

generates a number in the range 1 through numCardsLeft

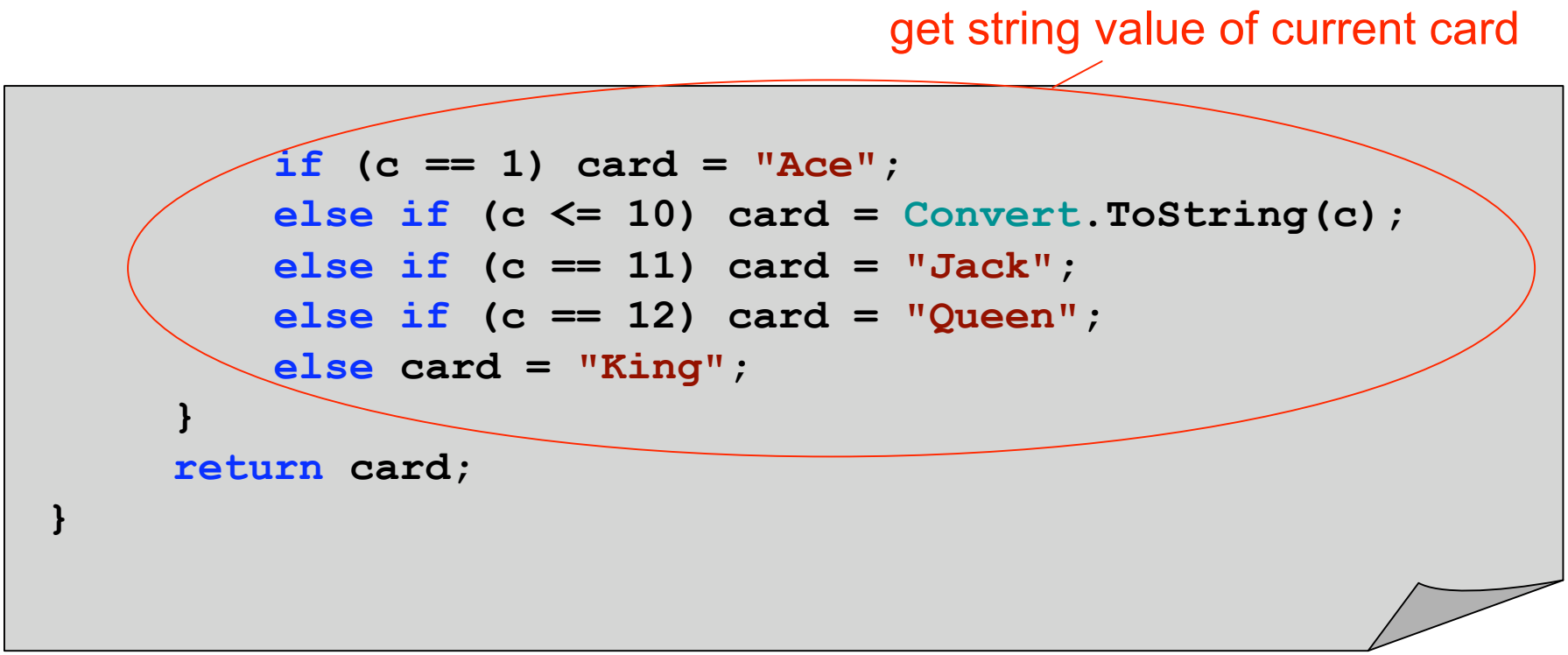
don't count cards where status is 1

mark it as "used"

increase #cards used

decrease #cards left

More Cards

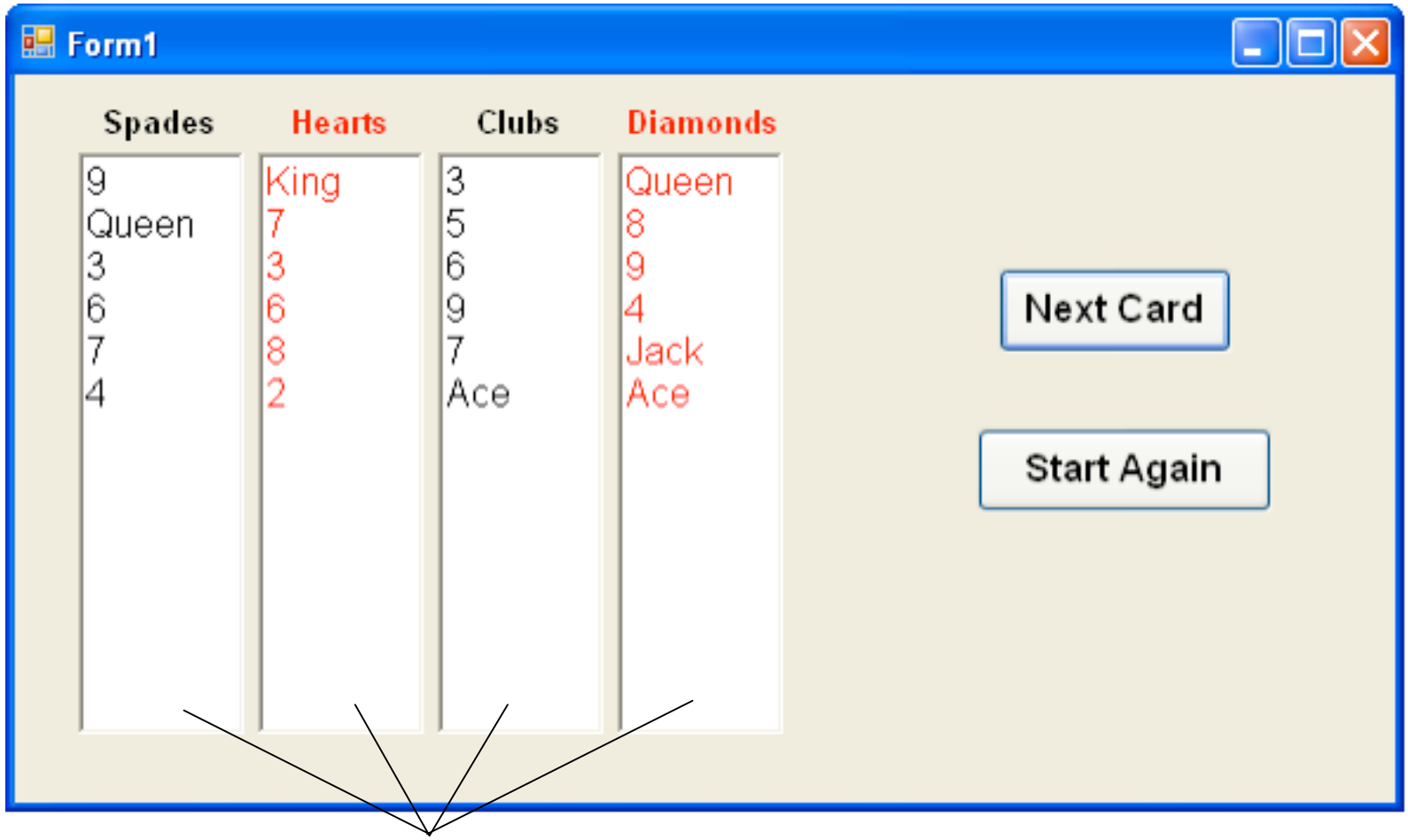


More Cards

gets everything set back to its original state

```
public void restore()
{
    for (int i = 0; i <= MAX; i++)
        status[i] = 0;
    numCardsLeft = MAX;
}
```

Using the Class



Each of these controlled by the class we made

Using the Class

- So - how do we use the class to create the 4 objects we need?

More Cards

```
public partial class Form1 : Form
{
    Suit spades = new Suit("Spades", 200);
    Suit hearts = new Suit("Hearts", 300);
    Suit clubs = new Suit("Clubs", 400);
    Suit diamonds = new Suit("Diamonds", 500);
```

Declare 4 objects.

Note the use of the constructor to set the suit name and a seed for the random number generator.

More Cards

```
private void btnNextCard_Click(object sender, EventArgs e)
{
    string s, s1, s2, s3, s4;
    s1 = rtbSpades.Text;
    s2 = rtbHearts.Text;
    s3 = rtbClubs.Text;
    s4 = rtbDiamonds.Text;
    s = spades.deal();
    if (s != null)
    {
        if (s1 != "") s1 += "\n";
        s1 += s;
        btnStartAgain.Visible = true;
    }
    s = hearts.deal();
    if (s != null)
    {
        if (s2 != "") s2 += "\n";
        s2 += s;
    }
```

get the current string to add to it

get next card in that suit

More Cards

```
    s = clubs.deal();
    if (s != null)
    {
        if (s3 != "") s3 += "\n";
        s3 += s;
    }
    s = diamonds.deal();
    if (s != null)
    {
        if (s4 != "") s4 += "\n";
        s4 += s;
    }
    rtbSpades.Text = s1;
    rtbHearts.Text = s2;
    rtbClubs.Text = s3;
    rtbDiamonds.Text = s4;
}
```

replace string with new entry

More Cards

```
private void btnStartAgain_Click(object sender, EventArgs e)
{
    btnStartAgain.Visible = false;
    spades.restore();
    rtbSpades.Text = "";
    hearts.restore();
    rtbHearts.Text = "";
    clubs.restore();
    rtbClubs.Text = "";
    diamonds.restore();
    rtbDiamonds.Text = "";
}
```

A Little More on Classes

- We have just scratched the surface on object-oriented programming.
- The focus of this course is algorithms and their implementation in C# - so finer details on C# are out of scope.
- However, for interest and completeness you may want to read a little on *inheritance* and *polymorphism* - especially *inheritance*. ☺