
Event, Exceptions, Message Boxes, and Forms

9/15/05

CS360 Windows Programming

1

Last Time

- We covered
 - Garbage collection
 - Delegates

9/15/05

CS360 Windows Programming

2

Example Using Delegates

```
delegate void WorkStarted();
delegate void WorkProgressing();
delegate int WorkCompleted();
```

9/15/05

CS360 Windows Programming

3

Example Using Delegates

```
class Worker
{
    public void DoWork()
    {
        Console.WriteLine("Worker: work started");
        if( started != null ) started();
        Console.WriteLine("Worker: work progressing");
        if( progressing != null ) progressing();
        Console.WriteLine("Worker: work completed");
        if( completed != null )
        {
            int grade = completed();
            Console.WriteLine("Worker grade= " + grade);
        }
    }
    public WorkStarted started;
    public WorkProgressing progressing;
    public WorkCompleted completed;
}
}
```

9/15/05

CS360 Windows Programming

4

Example Using Delegates

```
class Boss
{
    public int WorkCompleted()
    {
        Console.WriteLine("Better...");
        return 4; /* out of 10 */
    }
}
class Friend
{
    public void WorkStarted()
    {
        Console.WriteLine("You can do it");
    }
    public void WorkProgressing()
    {
        Console.WriteLine("Keep it up");
    }
    public int WorkCompleted()
    {
        Console.WriteLine("You're great!");
        return 10; /* out of 10 */
    }
}
```

9/15/05

CS360 Windows Programming

5

Example Using Delegates

```
class Universe
{
    static void Main()
    {
        Worker peter = new Worker();
        Boss boss = new Boss();
        Friend jack = new Friend();
        peter.completed = new
            WorkCompleted(boss.WorkCompleted);
        peter.started = new
            WorkStarted(jack.WorkStarted);
        peter.progressing = new
            WorkProgressing(jack.WorkProgressing);
        peter.completed = new
            WorkCompleted(jack.WorkCompleted);
        peter.DoWork();
        Console.WriteLine("Main: worker completed work");
        Console.ReadLine();
    }
}
```

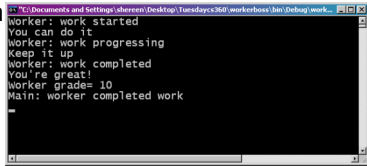
9/15/05

CS360 Windows Programming

6

Output

- What's happened to the grade and message from the Boss once work is com



```
Worker: work started
You can do it!
worker: work progressing
Keep it up
worker: work completed
You're great!
worker grade= 10
Main: worker completed work
```

9/15/05

CS360 Windows Programming

7

Delegates

- To solve the problem just experienced, we need to use events

```
class Worker
{
    public void DoWork()
    {
        .....
    }
    public event WorkStarted started;
    public event WorkProgressing progressing;
    public event WorkCompleted completed;
}
```

9/15/05

CS360 Windows Programming

8

```
static void Main()
{
    Worker peter = new Worker();
    Boss boss = new Boss();
    Friend jack = new Friend();
    peter.completed += new WorkCompleted(boss.WorkCompleted);
    peter.started += new WorkStarted(jack.WorkStarted);
    peter.progressing += new
        WorkProgressing(jack.WorkProgressing);
    peter.completed += new WorkCompleted(jack.WorkCompleted);
    peter.DoWork();

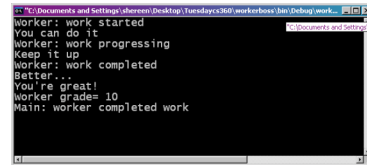
    Console.WriteLine("Main: worker completed work");
    Console.ReadLine();
}
```

9/15/05

CS360 Windows Programming

9

Output



```
Worker: work started
You can do it!
worker: work progressing
Keep it up
worker: work completed
Better..
You're great!
worker grade= 10
Main: worker completed work
```

9/15/05

CS360 Windows Programming

10

Another Problem

- But what about the grade?
- Only one grade shows up
- We need to harvest the grades (i.e. collect them)
 - How?

9/15/05

CS360 Windows Programming

11

Solution

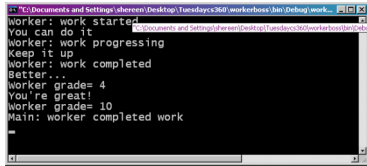
```
class Worker
{
    public void DoWork()
    {
        .....
        if( completed != null )
        {
            foreach( WorkCompleted wc in
                completed.GetInvocationList() )
            {
                int grade = wc();
                Console.WriteLine("Worker grade= " + grade);
            }
        }
    }
    public event WorkStarted started;
    public event WorkProgressing progressing;
    public event WorkCompleted completed;
}
```

9/15/05

CS360 Windows Programming

12

Output



```
.. "C:\Documents and Settings\shreem\Desktop\Tuesday's 360\workboss\bin\Debug\work...
Worker: work started
You can do it
Worker: work progressing
Keep it up
Worker: work completed
Better...
Worker grade= 4
You're great!
Worker grade= 10
Main: worker completed work
```

9/15/05

CS360 Windows Programming

13

Exceptions

```
public void SomeMethod(...)
{
    File file = new File("Readme.txt");
    ...
    file.Close();
}
```

What happens if the file doesn't exist?

What happens if an error occurs in here?

9/15/05

CS360 Windows Programming

14

Exceptions: the general idea

```
try
{
    some code that might throw an exception
    more code
}
catch (most specific exception)
{
    handle the exception
}
catch (less specific exception)
{
    handle the exception
}
catch (any exception)
{
    handle the exception
}
finally
{
    do this no matter what
}
still more code
```

9/15/05

CS360 Windows Programming

15

Exceptions

```
using System.IO;
public void SomeMethod(...)
{
    File file = null;
    try
    {
        file = new File("Readme.txt");
        more code
    }
    catch (FileNotFoundException e)
    {
        Console.WriteLine("File " + e.FileName + " not found");
    }
    catch (Exception e)
    {
        Console.WriteLine(e);
    }
    finally
    {
        if (file != null)
            file.Close();
    }
}
```

9/15/05

CS360 Windows Programming

16

Exceptions

- In a catch block, you can:
 - Rethrow the same exception, notifying code higher in the call stack
 - Throw a different exception, giving additional information to code higher in the call stack
 - Handle the exception and fall out the bottom of the catch block

9/15/05

CS360 Windows Programming

17

Exceptions

- Remember that:
 - Exceptions are not always "errors"
 - Exceptions are not always infrequent
 - Sometimes it's best not to catch an exception where it occurs
 - There is a performance hit for exceptions

9/15/05

CS360 Windows Programming

18

Windows Programming

- Although the title of this course is “Windows Programming”, we have still not done any windows programming
- All our programs have been console applications
- What exactly is the difference between a console application and a Windows application?

9/15/05

CS360 Windows Programming

19

Message Boxes

- The easiest form of a Windows application is one that displays a message box

```
class MessageBoxHelloWorld
{
    public static void Main()
    {
        System.Windows.Forms.MessageBox.Show(
            "Hello, world!");
    }
}
```

- `System.Windows.Forms` is a namespace
- `MessageBox` is a class in that namespace
- `Show` is a static method in the `MessageBox` class

9/15/05

CS360 Windows Programming

20

Message Boxes



9/15/05

CS360 Windows Programming

21

Message Boxes

- What other forms of the method show are provided by the FCL?
- How can we find that out?
- How can we change the buttons that are displayed?
- How can we change the icon displayed?

9/15/05

CS360 Windows Programming

22

Message Boxes

- How can we make the buttons do different things?

```
DialogResult dr = MessageBox.Show(
    "Do you want to create a new file?",
    "WonderWord",
    MessageBoxButtons.YesNoCancel,
    MessageBoxIcon.Question);

if (dr == DialogResult.Yes)
{
    // "Yes" processing
}
else if (dr == DialogResult.No)
{
    // "No" processing
}
else
{
    // "Cancel" processing
}
```

9/15/05

CS360 Windows Programming

23

Another Message Box

```
using System;
using System.Windows.Forms;

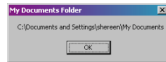
class MyDocumentsFolder
{
    public static void Main()
    {
        MessageBox.Show(
            Environment.GetFolderPath(
                Environment.SpecialFolder.Personal),
            "My Documents Folder");
    }
}
```

9/15/05

CS360 Windows Programming

24

Output



9/15/05

CS360 Windows Programming

25

Forms

- Message boxes are not Windows applications
- Windows applications need what has traditionally been called *windows* but are now called *forms*
- Forms consist of
 - Caption bar (title bar)
 - Menu bar
 - Client area

9/15/05

CS360 Windows Programming

26

Forms

```
using System.Windows.Forms;
```

```
class ShowForm
{
    public static void Main()
    {
        Form form = new Form();
    }
}
```

- But where is the form?

9/15/05

CS360 Windows Programming

27

Forms

```
using System.Windows.Forms;
```

```
class ShowForm
{
    public static void Main()
    {
        Form form = new Form();
        form.Show();
    }
}
```

- Don't Blink!

9/15/05

CS360 Windows Programming

28

Forms

- How can we make the form stay up for a while?
 - Add a delay!

```
using System.Threading;
using System.Windows.Forms;

class ShowFormAndSleep
{
    public static void Main()
    {
        Form form = new Form();
        form.Show();
        Thread.Sleep(2500);
        form.Text = "My First Form";
        Thread.Sleep(2500);
    }
}
```

9/15/05

CS360 Windows Programming

29

Your Turn

- For the remainder of the class, you will develop your own Windows applications following the guidelines in your handout

9/15/05

CS360 Windows Programming

30

Summary

- Completed talking about
 - Delegates
 - Exception handling
- Started talking about
 - Windows programming using message boxes
 - Windows programming using forms