PV178: Programming for the CLI Environment

Seminar: Week 4

Michal Ordelt, Tomáš Pochop

Institute of Computer Science and **Faculty of Informatics**
Masaryk University

March 25, 2007

Regular expressions

```
1 1 using System.Text.RegularExpressions;
2 2 private Regex rx;
3 3 rx = new Regex (pattern);
4 4 if(rx.IsMatch(inputText)){
5 5 \\some match found, do something
6 6 }
```

Some regular expressions elements

Quantifiers:

+ One or more matches
* Zero, one or more matches

Character classes:

\d Decimal digit
\s Any whitespace
. Any character except \n

For more regular expressions language elements see MSDN.

Regular expressions grouping

1. `(?'groupname'`*`pattern`*`)`
   Captures the matched substring into a group groupname. The string used for groupname must not contain any punctuation and it cannot begin with a number.
2. Class `MatchCollection` represents the set of successful matches found by iteratively applying a regular expression pattern to the input string.
3. `MatchCollection Regex.Matches(string input)` method searches an input string for all occurrences of a regular expression and returns all the successful matches as if Match were called numerous times.
4. `Match.Groups["`*`groupname`*`"].Value` returns the matched substring for group groupname.

Regular expressions grouping example

```
1 //capture subexpressions groups number and text
2 rx = new Regex(@"(?'number'\d+)\s+(?'text'.*)");
3 MatchCollection matches = rx.Matches(someString);
4 //go through all matched elements
5 foreach(Match match in matches){
6   //access groups
7   string num = match.Groups["number"].Value;
8   string text = match.Groups["text"].Value;
9 }
```

Regular Expressions Task: Subtitle timing, part 1

A simple .sub subtitle file contains a lot of lines in format: {*startframe*}{*endframe*}*Text to be shown*. Write an application that moves the subtitle forward or back by a specified amount of time.
First implement class `Line` representing a single line of the subtitle file. This class stores the startframe, endframe and text, and provide the method `Move(int64 seconds, double framerate)` which moves the startframe and endframe by the specified number of second, at the specified framerate. The overriden method `ToString()` returns the line in the .sub format.

Hints:

Framerate means number of frames per second
The example .sub file has framerate 23.976
To convert `double` to `Int64` use `Convert.ToInt64(double)`

Regular Expressions Task: Subtitle timing, part 2

Next implement class `Subtitle : IEnumerable<Line>` representing the subtitle file. This class reads one by one all lines from the input file, using regular expresions fills Line class and returns it as an enumerator.
Move each line by the specified amount of seconds and write it to the output file.
The input file, framerate, time and output file are set from command line as arguments.

Hints:

To read text from file use TextReader class
Use encoding Encoding.GetEncoding(1250) for file I/O
Remember to close all streams, readers and writers.

Writing to a text file

To write to a file use the TextWriter class. Example:

```
1  //write to the file file.out
2  FileStream fs = File.Create("file.out");
3  TextWriter wr;
4  //assign the textwriter to the filestream
5  wr = new StreamWriter(fs,
6     Encoding.GetEncoding(1250));
7  //write a line to the file
8  wr.WriteLine("Written in file");
9  wr.Close(); //close the writer
10 fs.Close(); //close the stream
```

Exceptions

C#, like many object-oriented languages, handles errors and abnormal conditions with exceptions. An exception is an object that encapsulates information about an unusual program occurrence.
It is important to distinguish between bugs, errors, and exceptions. A bug is a programmer mistake that should be fixed before the code is shipped. Exceptions are not a protection against bugs. Although a bug might cause an exception to be thrown, you should not rely on exceptions to handle your bugs. Rather, you should fix the bug.
When your program encounters an exceptional circumstance, such as running out of memory, it throws (or "raises") an exception. When an exception is thrown, execution of the current function halts and the stack is unwound until an appropriate exception handler is found.

Exceptions

To signal an abnormal condition in a C# class, you throw an exception. To do this, use the keyword throw. This code creates a new instance of System.Exception and then throws it:

```
1 Exception exception;
2 exception = new Exception("Something bad happened")
3 throw exception;
```

Exceptions handling

In C#, an exception handler is called a catch block and is created with the catch keyword.

```
try{
  //an exception can occur here
}
catch(ExceptionType e){
  //here we catch an ExceptionType exception
  //and all derived from ExceptionType too
}
finally{
  //this code is always executed
}
```

Exception class: useful members

1. `string Message`
   The error message that explains the reason for the exception.
2. `string StackTrace`
   A string that describes the contents of the call stack
3. `string ToString()`
   The default implementation obtains the name of the class that threw the current exception, the message, the result of calling ToString on the inner exception, and the StackTrace.

Own exceptions

You can create your own exception classes by deriving from the Exception class. Example:

```
1  //derive from the Exception class
2  public class MyException: Exception {
3     //constructor with message
4     public MyException(string message)
5        : base(message) {
6        } //ctor with message and inner exception
7     public Exception(string message, Exception inner)
8        : base(message, inner) {
9        }
10 }
```

Raise MyException:

```
1  throw new MyException("My exception occured!");
```

Exceptions task

Add exceptions handling to the Subtitle timing application and cover all possible problems, that may occur (can't read from input file, can't write to output file, wrong format of line, etc.). Try to use your own exception too.