PV178: Programming for the CLI Environment

Seminar: Week 3

Michal Ordelt

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

March 19, 2007

Customizing output

`public virtual string ToString()`
This method returns a human-readable string that represents the current Object.

Example:
Modify `Library` class and `Book` structure from previous seminar. For displaying do not use `Print()` method, override `ToString` method and use `Console.Write()` method. Do not forget to display capacity. Use `StringBuilder` class.

Reading a text file

```
1 using System.IO;
2 \\declaring a TextReader
3 private TextReader textreader;
4 \\asigning a stream to TextReader
5 textreader = new StreamReader(path);
```

Example:
Create `GetData:IEnumerable<string>` class with constructor having one parameter of type `string`. The class reads lines from the text file specified by the parameter and returns enumerator that goes throught lines of file. Try your class on the file from IS.

Regular Expressions

A regular expression consists of two types of characters: *literals* and *metacharacters*. A literal is a character you wish to match in the target string. A metacharacter is a special symbol that acts as a command to the regular expression parser. The parser is the engine responsible for understanding the regular expression.

All the objects associated with regular expressions are in the Base Class Library namespace `System.Text.RegularExpressions`. The central class for regular expression support is Regex, which represents an immutable, compiled regular expression. Although instances of Regex can be created, the class also provides a number of useful static methods.

Regular Expressions

```
1 using System.Text.RegularExpressions;
2 private Regex rx;
3 rx = new Regex(expression);
4 if (rx.IsMatch(text)){
5       \\Do something
6 }
```

Example:
Extend the previous example as follows. The constructor have two parameters, the first is the path, the second is the regular expression, and it returns only string matching the expression. Implement one parametric constructor using two parametric constructor.

Regular Expressions

Regex.Replace Method

```
public string Replace (
    string input,
    string replacement
)
```

Example:
Extend the previous example again. The returned strings will have all rude words replaced by -.

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;
```

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

```
1 Encoding encoding = Encoding.GetEncoding(codepage)
2 encoding.GetBytes(string)
3 encoding.GetString(byte[])
```

```
public StreamWriter (
    string path,
    bool append,
    Encoding encoding
)
```

Example:
Create a class that convert text file from one coding to another and saves it as new file. Use exceptions to cover all possible problems, that may occur.