Anglia Ruskin University

Exceptions and File Handling

- An ‘exception’ is a type of error that disrupts the *normal flow of program execution* and can cause a program to unexpectedly terminate (or ‘crash’).

- Exception-handling is managing the exception to ensure the program executes without crashing – it can be regarded as a kind of safety measure.

Anglia Ruskin University

Operations the programmer <u>might</u> want to check for

- *Eg:*

File handling
Keyboard input
Array processing

```
int x = 57;
int y = 0;
int result = x / y;
```

Program will compile and therefore can execute BUT would crash

Unhandled Exception: DivideByZeroException…

Anglia Ruskin University

Programs can be designed to process an exception using a **try/catch** block:

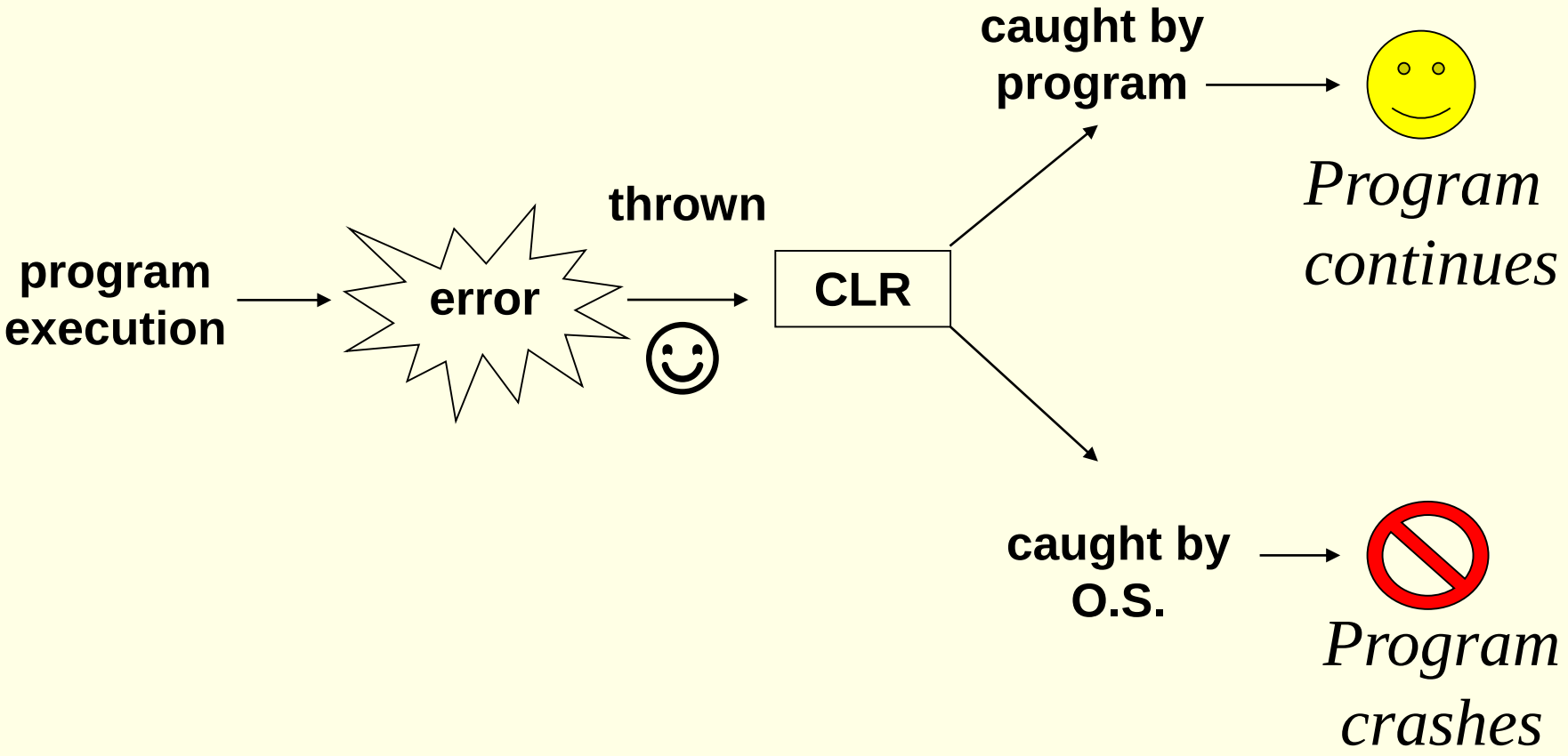
```
try
{
   // try to do this
}
catch(  objectname )
{
   // if it can't be done, do this
}

//  Other keywords include 'finally' and 'throws'
```

Anglia Ruskin University

```
int x = 57, y = 0, result;
try
{
    result = x / y;
}
catch( Exception ex )
{
    Console.WriteLine(ex.Message);
    Console.WriteLine("Result is Infinity");
}
```

Attempted to Divide By Zero.
Result is Infinity.

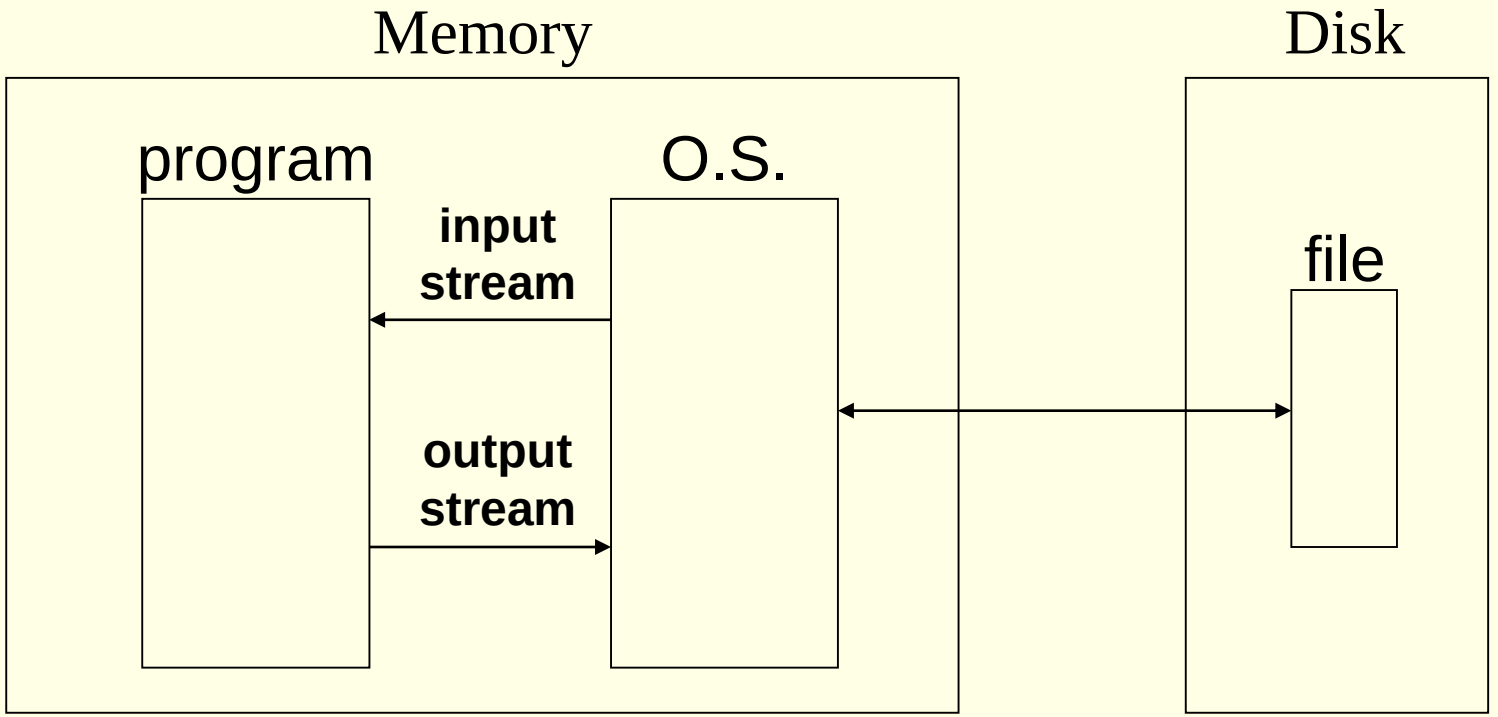
Anglia Ruskin University



Anglia Ruskin University

Streams and Files

A *stream* = one-way transmission of data in bytes



A ***file stream object*** connects a program to a file

Using System.IO;

Anglia Ruskin University

Can distinguish two kinds of file:

- Text file
- Binary file

There is ***no difference*** in how data is stored – all binary (1001001101010)

The difference is in ***how the bit patterns are interpreted***.

Anglia Ruskin University

Text Files

– data written as a series of ascii codes in sequential bytes

Eg: store the number **2681** in a *text* file

character:	2	6	8	1
ascii code:	50	54	56	49
bit pattern:	00110010	00110110	00111000	00110001

Anglia Ruskin University

Writing to a Text File

```
string line = “Audit:";
double bulbs = 3.22;

StreamWriter writer;

try
{
   writer = new StreamWriter( "output.txt" );
}
catch( IOException e )
{
   Console.WriteLine("Error writing to file.");
}

writer.WriteLine( line );
writer.WriteLine( “Number of bulbs is " + bulbs );
writer.Close();
```

In same location as .exe or can specify path

Anglia Ruskin University

Reading from a Text File

In same location as .exe or can specify path

```
string line;
try
{
   StreamReader reader =
        new StreamReader( @"C:\MyFiles\Test.txt" );

   while( reader.EndOfStream == false )
   {
      line = reader.ReadLine();
      Console.WriteLine( line );
   }
}
catch( IOException e )
{
   Console.WriteLine( "File input error" );
}
```

Anglia Ruskin University

Binary Files

– data written as a series of fixed-format sizes according to type

Eg: store the number **2681** in a *binary* file

bit pattern: 00000000 00000000 00001010 01111001

binary files – bytes in fixed format (eg int)

Anglia Ruskin University

Writing/Reading to/from a Binary File

Writing to a binary file	*Reading from a binary file*
BinaryWriter FileStream	**BinaryReader** FileStream
Write(*variable_or_value***)**	**ReadInt32()** **ReadDouble()** **ReadByte()**

Anglia Ruskin University

Text file or Binary file?

Could use a text file if the data is:

- to be readable to the eye
- to be portable
- to store small numerical values

Could use a binary file if the data is:

- to contain fixed-length records
- to store large numerical values