Exceptions

08101 Programming 1
C# Programming

Exceptions

- There are two kinds of programming error
- Compilation error
  - Compiler complains that our source is not valid C#
- Run time error
  - Program crashes when it runs
- Most program crashes take the form of exceptions

Exceptional Circumstances

- Exceptions are created when your program does something bad:
  - Trying to parse an invalid string
  - Trying to position the cursor off the screen
  - Trying to use a location outside the bounds of an array
  - ...there are many others...
- The C# language has exceptions built in for these bad things

What is an exception?

- An exception is a lump of data which describes something bad which has just happened
- You can use it to find out why your program just failed or you can ignore it
- Exceptions are "thrown" by a program or by the run time system
- If an exception is not "caught" your program will fail

UNIVERSITY OF Hull

Causing exceptions

```
int i;
i = int.Parse("One");
```

- This code is doomed to failure
- The `Parse` method is expecting a string that contains a number as a sequence of digits
- If it is given text it throws an exception

Catching exceptions

```
try
{
   i = int.Parse("One");
}
catch
{
   Console.WriteLine("Invalid number");
}
```

- If the code in the try block causes an exception execution transfers to the catch

Safe Software

- By putting potentially dangerous code into a try block we can stop it from crashing our programs
- We could then use a loop to repeat the operation if an exception has been thrown
- We should put this behaviour into our number reading methods so that users cannot crash our programs

Exception objects

- If we wish, we can capture the exception that has been thrown
- We can then use this to allow our program to respond to particular errors or output extra diagnostic information
- The exception can be picked up by the catch clause

Catching exception details

```
try
{
  i = int.Parse("One");
}
catch (Exception e)
{
  Console.WriteLine(e.Message);
  Console.WriteLine(e.StackTrace);
}
```

- We can use properties in the exception object to display further information

Finally and “last wishes”

- Sometimes you want some code that will run whether the exception is thrown or not
  - This code could close files or release resources
- The `try - catch` construction can have a finally clause added to the end
- Statements in this clause are obeyed whether or not the exception is thrown

The Finally Clause

```
try
{
  // Code that might throw an exception
}
catch
{
  // Code to deal with exception
}
finally
{
  // Code that is always obeyed
}
```

Exception Types

- There are a variety of exceptions which can be thrown, depending on the source of the error
  - File open exceptions, overflow exceptions, array subscript exceptions
- You can even create your own exception types
- However, if you catch the Exception type this will have the effect of catching any of these

Throwing your own exceptions

- You can make your program throw exceptions

```
throw new Exception ("Bang");
```

- This creates a new exception instance and then throws it
- If your program does not catch this exception it will be stopped as with any other

Exception Etiquette

- Exceptions should be reserved for really bad things happening
  - You should not reject a window width value by throwing an exception
- They allow a program to fail in a managed way, so that you can get control when something bad happens
- If your program encounters something that will stop it continuing, it should throw an exception

Summary

- Exceptions are a way of managing errors
- They are an object that describes something bad that has just happened
- They are used when a program has good reasons why it cannot continue
- A program can deal with exceptions by using the try – catch – finally construction