Exceptions

Introduction

- An **exception** is an indication of a problem that occurred during a program's execution.
- Exception handling enables you to create apps that can handle exceptions—in many cases allowing a program to continue executing as if no problems were encountered.
- Exception handling enables you to write clear, **robust** and more **fault-tolerant programs**.

Exceptions

- Some circumstances are beyond programmer's control
  - You have assumed nothing unusual would occur
- Have probably experienced unhandled exceptions being thrown
- Unless provisions are made for handling exceptions, your program may crash or produce erroneous results
  - Unhandled exception

Raising an Exception

- Error encountered – no recovery
  - Raise or throw an exception
  - Execution halts in the current method and the Common Language Runtime (CLR) attempts to locate an exception handler

- Exception handler: block of code to be executed when a certain type of error occurs
  - If no exception handler is found in current method, exception is thrown back to the calling method

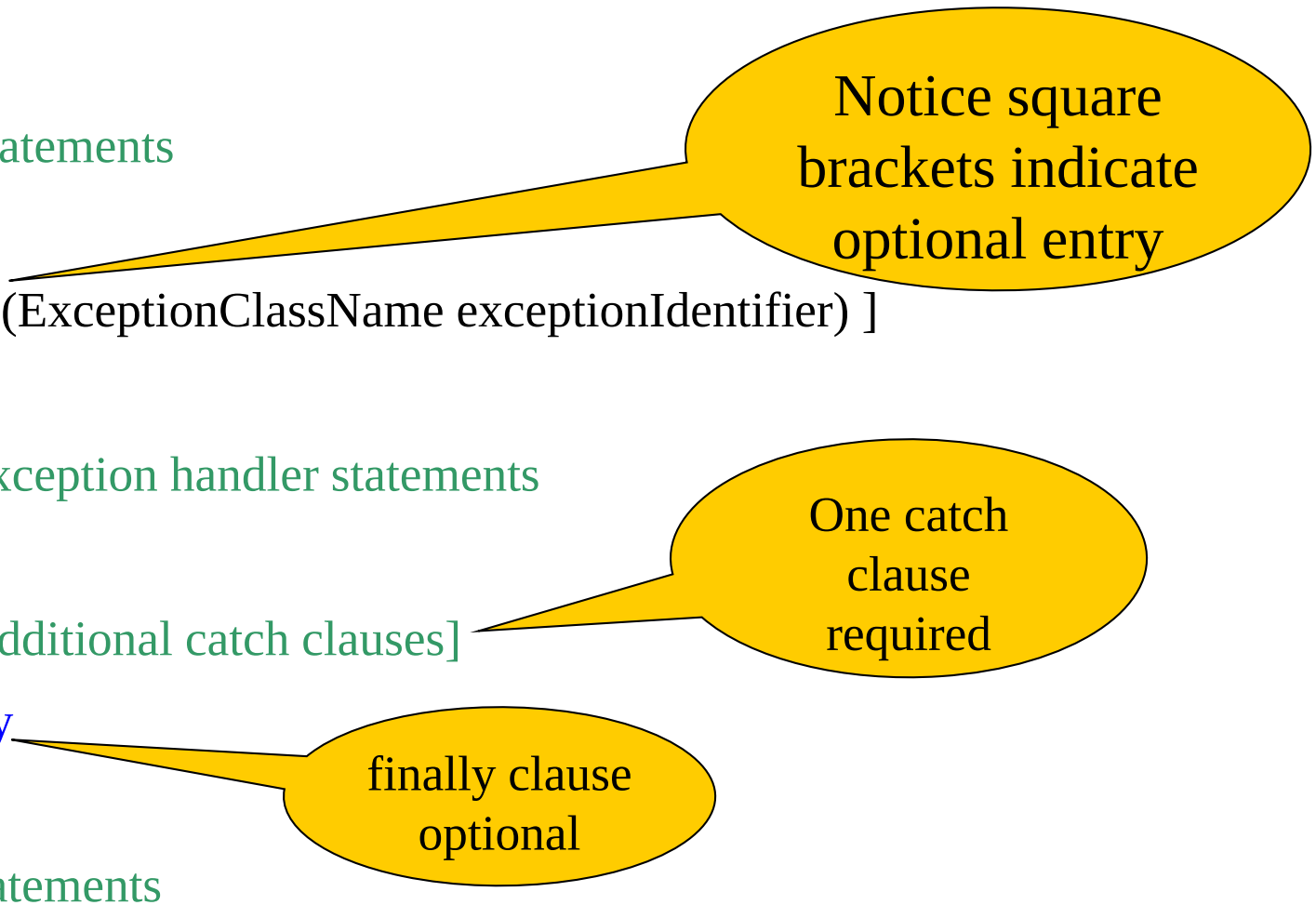
Exception-Handling Techniques

- If event creates a problem frequently, best to use conditional expressions to catch and fix problem
  - Execution is slowed down when CLR has to halt a method and find an appropriate event handler
- Exception-handling techniques are for serious errors that occur infrequently
- Exceptions classes integrated within the Framework Class Library
  - Used with the try…catch…finally program constructs

Try…Catch…Finally Blocks

- Code that may create a problem is placed in the try block
- Code to deal with the problem (the exception handler) is placed in catch blocks
  - Catch clause
- Code to be executed whether an exception is thrown or not is placed in the finally block

```
try
{
    // Statements
}
catch [ (ExceptionClassName exceptionIdentifier) ]
{
    // Exception handler statements
}
:   // [additional catch clauses]
[ finally
{
   // Statements
} ]
```



Try…Catch…Finally Blocks

- Generic catch clause
  - Omit argument list with the catch
  - Any exception thrown is handled by executing code within that catch block
- Control is never returned into the try block after an exception is thrown
- Using a try…catch block can keep the program from terminating abnormally

finally Block

- Programs frequently request and release resources dynamically.
- Operating systems typically prevent more than one program from manipulating a file.
- Therefore, the program should close the file (i.e., release the resource) so other programs can use it.
- If the file is not closed, a resource leak occurs.

finally Block

Moving Resource-Release Code to a finally Block

- Exceptions often occur when an app processes resources that require explicit release.
- Regardless of whether a program experiences exceptions, the program should close the file when it is no longer needed.
- C# provides the finally block, which is guaranteed to execute regardless of whether an exception occurs.
- This makes the finally block ideal to release resources from the corresponding try block.
- Local variables in a try block cannot be accessed in the corresponding finally block, so variables that must be accessed in both should be declared before the try block.

Built-in Exceptions

Table 11-2 Derived classes of SystemException

Exception classes derived from the SystemException class	Description of circumstances causing an exception to be thrown
`System.ArgumentException`	One of the arguments provided to a method is invalid.
`System.ArithmeticException`	There are errors in an arithmetic, casting, or conversion operation. Has derived members of System.DivideByZeroException and System.OverflowException.
`System.ArrayTypeMismatchException`	An attempt is made to store an element of the wrong type within an array.
`System.FormatException`	The format of an argument does not meet the parameter specifications.

Table 11-2 Derived classes of SystemException

Exception classes derived from the SystemException class	Description of circumstances causing an exception to be thrown
`System.IndexOutOfRangeException`	An attempt is made to access an element of an array with an index that is outside the bounds of the array.
`System.InvalidCastException`	There is invalid casting or explicit conversion.
`System.IO.IOException`	An I/O error occurs.
`System.NullReferenceException`	There is an attempt to dereference a null `object` reference.
`System.OutOfMemoryException`	There is not enough memory to continue the execution of a program.
`System.RankException`	An array with the wrong number of dimensions is passed to a method.

Filtering Multiple Exceptions

- Can include multiple catch clauses
- Enables writing code specific to thrown exception
- Should be placed from most specific to the most generic