Винятки

- Переваги винятків
- Схема використання винятків
- Особливості catch-блоків
- Особливості finaly
- class Exception
- Конструювання об'єктів Exception
- Властивості класу Exception
- Винятки FCL
- ArgumentException
- Приклади підходів

Переваги винятків

- Аргумент винятку точно описує ситуацію
- Відокремлення коду обробки
- Виняток не можна проігнорувати
- **`unhandled exception`** == у стекові викликівCLR не знайшов відповідного **`catch`** **–>** негайне знищення потоку

Приклад використання винятків

```
FileStream s = null;

try
{
    s = new FileStream(curName, FileMode.Open);
     ...
}
catch (FileNotFoundException e)
{
    Console.WriteLine("file {0} not found", e.FileName);
}
catch (IOException)
{
    Console.WriteLine("some IO exception occurred");
}
catch
{
    Console.WriteLine("some unknown error occurred");
}
finally
{
    if (s != null) s.Close();
}
```

Схема використання винятків

```
void SomeMethod() {
    try {
      // Code requiring graceful recovery or common cleanup operations.
    }
    catch (InvalidOperationException) {
      // Code that recovers from an InvalidCastException or any derived type
    }
    catch (IOException) {
      // Code that recovers from a IOException or any derived type
    }
    catch (Exception e) {
      // Code that recovers from any CLS-compliant exception.
      // When catching a CLS compliant exception, you usually rethrow the exception
      throw;
    }
    catch {
    // Code that recovers from any exception, CLS-compliant or not.
    // When catching any exception, you usually rethrow the exception.
      throw;
    }
    finally {
    // Code that cleans up any operations started within the try block.
    // The code in this block ALWAYS executes, regardless of whether an exception is thrown.
    }
    // Code below the finally block executes if no exception is thrown within the try block
    //  or if a catch block catches the exception and doesn't throw or rethrow an exception.
}
```

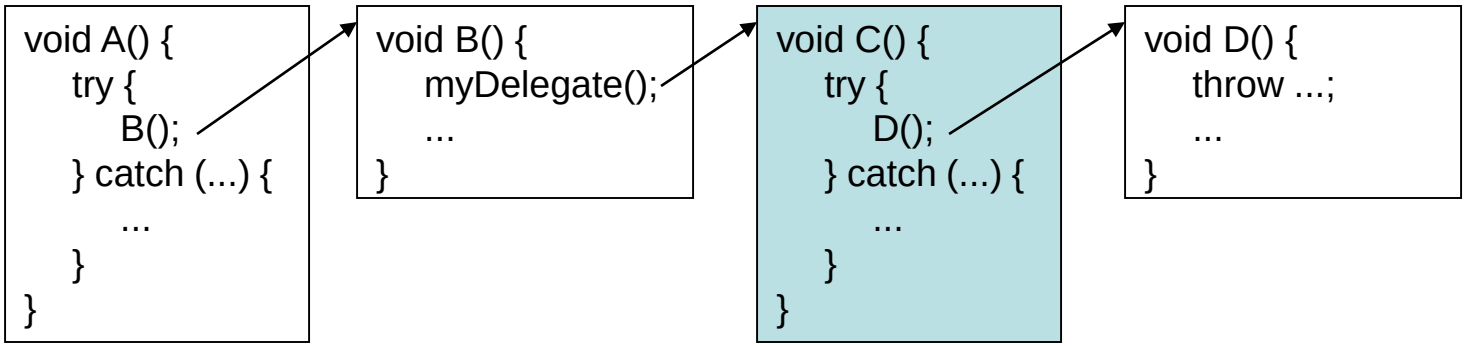
Особливості catch-блоків

- обробітники переглядаються послідовно щодо співпадіння винятку з аргументом
- розміщення `catch` в порядку поглиблення спеціалізації
- `catch{}`
  - якщо тип винятку пропущено, то мають на увазі `System.Exception` і будь-який похідний
  - будь-який (несумісний з CLR)

Особливості `finally`

- **`try`** може мати не більше одного **`finally`**
- **`finally`** завжди буде виконано
- у **`finally`** варто уникати коду, який може генерувати винятки
- якщо в **`finally`** згенерується виняток:
  - CLR продовжить виконання всіх операторів блоку **`finally`**
  - попередній виняток ігнорується
  - після виконання операторів **`finally`** почнеться обробка нового винятку

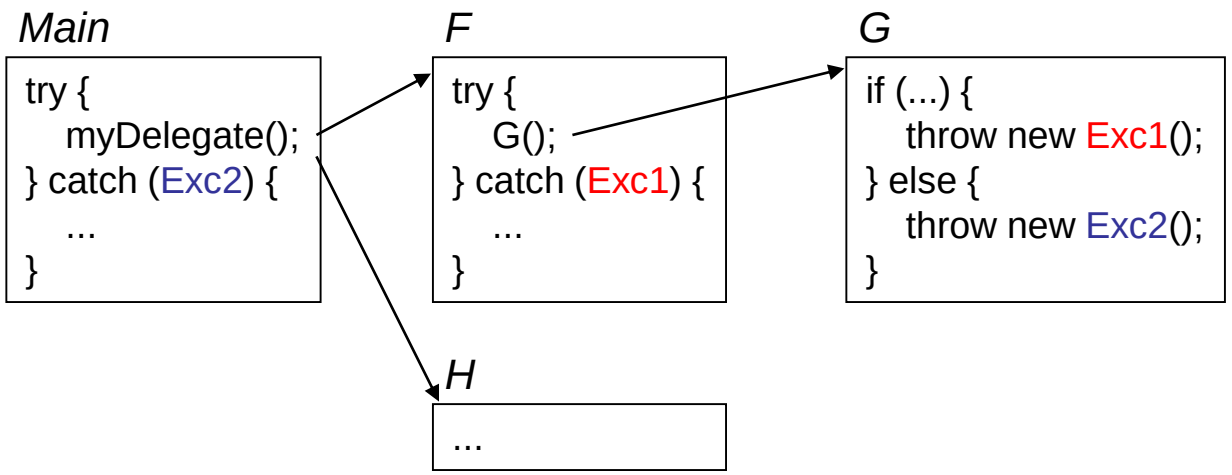
Exceptions in Delegates



called as a delegate

При обробці винятків делегати трактуються як звичайні методи

Exceptions in Multicast Delegates



- If Exc1 is thrown in G() it is caught in F() and then H() is called.
- If Exc2 is thrown in G() it is caught in Main() and H() is not called any more.

class Exception

```
[SerializableAttribute] [ComVisibleAttribute(true)]
[ClassInterfaceAttribute(ClassInterfaceType.None)]
public class Exception : ISerializable, _Exception
```

```
[InterfaceTypeAttribute(ComInterfaceType.InterfaceIsDual)]
[CLSCompliantAttribute(false)] [ComVisibleAttribute(true)]
[GuidAttribute("b36b5c63-42ef-38bc-a07e-0b34c98f164a")]
public interface _Exception
```

Exposes the public members of the System.Exception class to unmanaged code.

Конструювання об'єктів `Exception`

Name	Description
`Exception ()`	Initializes a new instance of the **`Exception`** class.
`Exception (String)`	Initializes a new instance of the **`Exception`** class with a specified error message.
`Exception (SerializationInfo, StreamingContext)`	Initializes a new instance of the **`Exception`** class with serialized data.
`Exception (String, Exception)`	Initializes a new instance of the **`Exception`** class with a specified error message and a reference to the inner exception that is the cause of this exception.

Properties of the System.Exception

Property	Access	Type	Description
Message	Read-only	**String**	Contains helpful text indicating why the exception was thrown. The message should be localized.
Data	Read-only	**IDictionary**	Gets a collection of key/value pairs that provide additional, user-defined information about the exception.
Source	Read/write	**String**	Contains the name of the assembly that generated the exception.
StackTrace	Read-only	**String**	Contains the names and signatures of methods called that led up to the exception being thrown. This property is very useful for debugging.
TargetSite	Read-only	**MethodBase**	Contains the method that threw the exception.
HelpLink	Read-only	**String**	Contains a URL (such as *file://C:\MyApp\Help.htm#-MyExceptionHelp*) to documentation that can help a user understand the exception.
InnerException	Read-only	**Exception**	Indicates the previous exception if the current exception was raised while handling an exception. This field is usually **null**. The **Exception type** also offers a public **GetBaseException** method that traverses the linked list of inner exceptions and returns the originally thrown exception.

Основні категорії винятків

- Системні винятки, **System namespace**

```
[SerializableAttribute]
[ComVisibleAttribute(true)]
public class SystemException : Exception
```

- Винятки, що виникають в програмі, **System namespace**

```
[SerializableAttribute]
[ComVisibleAttribute(true)]
public class ApplicationException : Exception
```

Винятки FCL

- System.Exception
  - System.ApplicationException
    - System.Reflection.InvalidFilterCriteriaException
    - System.Reflection.TargetException
    - System.Reflection.TargetInvocationException
    - System.Reflection.TargetParameterCountException
  - System.IO.IsolatedStorage.IsolatedStorageException
  - System.SystemException
    - System.AppDomainUnloadedException
    - System.ArgumentException
      - System.ArgumentNullException
      - System.ArgumentOutOfRangeException
      - System.DuplicateWaitObjectException
    - System.ArithmeticException
      - System.DivideByZeroException
      - System.NotFiniteNumberException
      - System.OverflowException
    - System.ArrayTypeMismatchException
    - System.BadImageFormatException
    - System.ExecutionEngineException
    - .....

ArgumentException

```
[SerializableAttribute]
[ComVisibleAttribute(true)]
public class ArgumentException
        : SystemException, ISerializable
```

```
public virtual string ParamName { get; }
```

```
[SerializableAttribute]
[ComVisibleAttribute(true)]
public class ArgumentNullException : ArgumentException
```

The exception that is thrown when a null reference is passed to a method that does not accept it as a valid argument.

Приклади підходів

• коректне поновлення виконання

```
public String CalculateSpreadsheetCell(Int32 row, Int32 column) {
    String result;
    try {
        result = /* Code to calculate value of a spreadsheet's cell*/
    }
    catch (DivideByZeroException) {
        result = "Can't show value: Divide by zero";
    }
    return result;
}
```

• приховування деталей реалізації

```
public Int32 SomeMethod(Int32 x){
    try {
        return 100 / x;
    }
    catch (DivideByZeroException e) {
      throw new ArgumentOutOfRangeException("x can't be 0", e);
    }
}
```