EXCEPTIONS

Vincent Uhlmann
IT-Akademie Dr. Heuer GmbH

EXCEPTIONS / AUSNAHMEN

- Ereignisse die während der Ausführung des Programmes auftreten
  - Division durch 0
  - Invalide Typumwandlung
  - ...
- Unterbrechen den normalen Ablauf des Programms
- Erlauben uns auf Fehler zu reagieren
- Korrekte Handhabung verbessert die Robustheit und Stabilität des Systems

EXCEPTIONS / AUSNAHMEN

- Built-in Exception
  - DivideByZeroException
  - FileNotFoundException
  - ...

- Benutzerdefinierte Exceptions
  - LoginFailedException
  - LogoutFailedException
  - ...

BENUTZERDEFINIERTE EXCEPTIONS

- Basisklasse jeder Exception ist Exception

```
public class TestException : Exception
{
    public TestException(string msg) : base(msg)
    {
    }
}
```

THROW

- Schlüsselwort throw wird verwendet, um eine Ausnahme auszulösen
- Built-in Exceptions sowie Benutzerdefinierte Exceptions können so ausgelöst werden

```
throw new NotImplementedException();
```

EXCEPTION HANDLING

- try/catch ermöglicht uns das abfangen von Fehlern
- catch besteht aus runden Klammern, dem Typ der Exception sowie einem Namen für den Zugriff

```
try {
    // Fehler wird geworfen
    throw new NotImplementedException("Test Exception");
} catch(NotImplementedException ex) {
    // Fehler wird abgefangen und behandelt
    Console.WriteLine(ex.Message);
}
```

EXCEPTION HANDLING

- Jeder try Block muss von einem catch oder finally Block gefolgt sein
- Der finally Block wird immer ausgeführt und ist ideal zum "Aufräumen"
- Es können unbegrenzt viele catch Blöcke genutzt werden, um auf spezielle Exceptions zu reagieren
- Im Falle einer Exception wird der try Block verlassen und zum passenden catch Block gesprungen
- Info: Wenn in der aktuellen Methode kein entsprechender catch-Block vorhanden ist, prüft die Common Language Runtime (CLR) die vorhergehenden Methoden in der Aufrufliste, beginnend mit der Methode, die die aktuelle Methode aufgerufen hat. Falls kein passender catch-Block gefunden wird, beendet die CLR den aktuellen Thread.

EXCEPTION HANDLING

- try-catch Blöcke können verschachtelt werden, um eine präzise Fehlerbehandlung zu ermöglichen

```
try {
    try {
        throw new NotImplementedException();
    } catch (NotImplementedException) {
        throw;
    }
} catch(Exception ex) {
    Console.WriteLine(ex.Message);
}
```

EXCEPTION HANDLING

- Innere Try-Catch-Blöcke werden innerhalb äußerer Try-Catch-Blöcke platziert
- Dadurch können Fehler auf unterschiedlichen Ebenen der Anwendungslogik abgefangen und behandelt werden
- Innerhalb eines inneren Try-Catch-Blocks können spezifische Ausnahmen behandelt und dann mit throw nach außen geworfen werden

```
try {
    try {
        throw new NotImplementedException();
    } catch (NotImplementedException) {
        throw;
    }
} catch(Exception ex) {
    Console.WriteLine(ex.Message);
}
```

BEDINGTE AUSNAHMEN

- Das when Schlüsselwort ermöglicht die Definition von Bedingungen für das Abfangen von Ausnahmen in einem catch Block

```
try {
    // Code
} catch(Exception ex) when (DateTime.Now.DayOfWeek == DayOfWeek.Friday)
{
    Console.WriteLine(ex.Message);
}
```