C# Einführung

4. Die Programmiersprache C#

- ist vollwertige objektorientierte Programmiersprache
- geeignet für „fast alle" Anwendungsszenarien
- ähnelt Java

Allgemeine Eigenschaften

- Garbage Collection
  Keine Speicherlecks oder „wilde Pointer"
- Exceptions
  Error Handling von Anfang an bedacht
- Typsicherheit
  - Keine uninitialisierten Variablen
  - Keine unsicheren Casts
- Vielzahl von Bibliotheken erleichtert
  Anwendungsentwicklung sehr stark

C# Einführung

4.1 Detaillierte Eigenschaften

- Namespaces
  - Enthalten Typdefinitionen und Namespaces
- Typdefinitionen
  - Klassen, Strukturen, Interfaces, Delegaten
  - eigene Werttypen sind definierbar
- Elemente von Typen
  - Konstanten, Felder, Methoden, Properties, Indexer, Events, Operatoren, Konstruktoren, Destruktoren
  - Operatoren sind teilweise überladbar
- variable Datentypen: generics
- unmanaged Code: unsafe
- Organisation der Dateien
  - Keine Header-Dateien, Programmcode ist "in-line"
  - Die Reihenfolge der Deklarationen ist ohne Bedeutung

C# Einführung

C# Namenskonventionen

Identifier	Case	Example
Class	Pascal	**AppDomain**
Enum type	Pascal	**ErrorLevel**
Enum values	Pascal	**FatalError**
Event	Pascal	**ValueChange**
Exception class	Pascal	**WebException** suffix **Exception.**
Read-only Static field	Pascal	**RedValue**
Interface	Pascal	**IDisposable** prefix I
Method	Pascal	**ToString**
Namespace	Pascal	**System.Drawing**
Parameter	Camel	**typeName**
Property	Pascal	**BackColor**
Private, protected instance field	Camel	**redValue**
Public instance field	Pascal	**RedValue**

C# Einführung

Vergleich zu Java

- derzeit 2 dominate Entwicklungsumgebungen:
  - .NET und Java
  - beeinflussen sich positiv gegenseitig
- Problem der zerbrechlichen Basisklasse in C# nicht vorhanden; C# explizite Deklaration virtual erforderlich
- eigene Werttypen (structs) in C# definierbar
- C# Sprachen-interoperabel, Java nicht
- Java läuft auf „allen" Betriebssystemen
  C# hat Nähe zu Windows, Mono (Linux-Plattform)
- Java u. a. enthält keine:
  - Properties
  - Indexer
  - rechteckige Arrays
  - ref, out, params Parameterübergabe

C# Einführung

Vergleich zu C++

- C++ hat keine Garbage Collection
- C++ ermöglicht Zeigerarithmetik C# nicht
- C++ hat Mehrfachvererbung, C# nicht
- C++ bietet reine Implementierungsvererbung C# nicht
- C# exe enthält Metadaten, C++ exe nicht
- C++ sehr gut geeignet für systemnahe speicherkritische Anwendungen
- C# (Java) sehr gut geeignet für nicht systemnahe Anwendungen
- Prognose: C++ wird in vielen Anwendungsbereichen durch C# (Java) ersetzt werden, Zahl der C++ (unmanaged) Entwickler wird deutlich abnehmen
- .Net C++: managed C++ (mit GC)

C# Einführung

Aspeke für verteilte Komponenten

Wir werden nur folgende Aspekte von C#, die für den Einsatz als verteilte Komponenten im .Net Framework erforderlich sind und nicht direkt aus Java Kenntnissen erschlossen werden können, behandeln

- Properties
- Referenz- und Werttypen
- Vererbung
- Delegaten
- Attribute
- Serialisierung
- Reflection

Alle anderen Detaills sollten mit einem Buch eigenständig erarbeitet werden!

C# Einführung

Properties 1

- Eigenschaften sind spezielle Methoden, die den Zugriff auf Datenelemente ermöglichen, wobei der Anschein des direkten Zugriffs auf ein Feld entsteht
- tatsächlich erfolgt der Zugriff über Methoden

```
Bsp.: int p=11;          public int P{
                                  get{return p;  }
                                  set{p=value;)}
      Aufruf: Instanz.P = 7;
```

- mindestens eine dieser Methoden (get oder set) muss implementiert werden
- Vorteil: die Implementierung (z.B. Plausibilitäts-prüfungen)des dahinter verborgenen Elementes könnte geändert werden, ohne dass der Zugreifer hiervon beeinflusst wird

C# Einführung

Properties 2

- aus Sicht des Clients ist es eine Membervariable
- Definiton einer Eigenschaft entspricht Definition eines Datenelementes mit anschließendem Block für get und/oder set
- get und set haben keine expliziten Parameter
- set hat impliziten Parameter value
- get/set werden vom Compiler inline expandiert
- sie lassen sich nicht als Argumente an Methoden übergeben
- können static sein, abstract sein und vererbt werden
- als Namenskonvention wird der Bezeichner für die Property gleich dem Bezeichner des Datenelementes mit grossem Anfangsbuchstaben (Datenelement klein) benannt
- ab C# 3.0 automatische Properties:`string Name{get;set;}`