BBZW-Sursee;FMZ; Bernhard Tim

[Firmenname]

M404

Lernjournal

[C# 2](#_Toc69907204)

[Arrays in Konsolenprogramm 3](#_Toc69907205)

[Array erstellen 3](#_Toc69907206)

[Array bearbeiten 3](#_Toc69907207)

[Array-Daten auslesen 3](#_Toc69907208)

[For-Schleife 4](#_Toc69907209)

[Foreach-Schleife 4](#_Toc69907210)

[Random Klasse 4](#_Toc69907211)

[Try and catch 5](#_Toc69907212)

[Enumeratoren 5](#_Toc69907213)

[Klassen 6](#_Toc69907214)

[Instanz 6](#_Toc69907215)

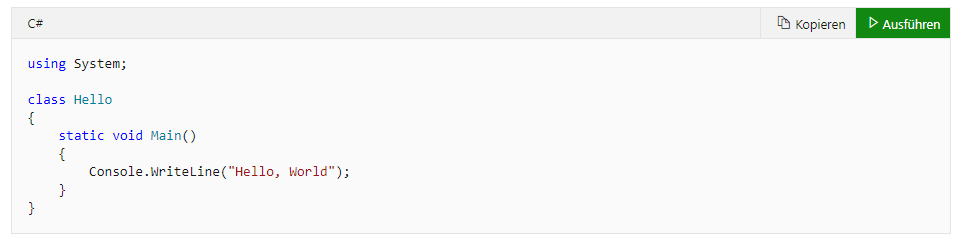
[Objekte 6](#_Toc69907216)

# C# Allgemein

C# (Aussprache „C Sharp“) ist eine moderne, objektorientierte und typsichere Programmiersprache. C# ermöglicht Entwicklern das Erstellen zahlreicher sicherer und robuster Anwendungen, die im .NET-Ökosystem ausgeführt werden. C# hat seine Wurzeln in der C-Sprachenfamilie. C# ist eine objektorientierte, komponentenorientierte Programmiersprache.

C#-Programme werden auf Grundlage von .NET ausgeführt, ein virtuelles Ausführungssystem namens Common Language Runtime (CLR) sowie Klassenbibliotheken. Die CLR ist die Implementierung der Common Language Infrastructure (CLI) von Microsoft, ein internationaler Standard. Die CLI ist die Grundlage für das Erstellen von Ausführungs- und Entwicklungsumgebungen, in denen Sprachen und Bibliotheken nahtlos zusammenarbeiten. C# ist eine objektorientierte, komponentenorientierte Programmiersprache.

.NET enthält auch umfangreiche Bibliotheken. Diese Bibliotheken unterstützen viele verschieden Workloads. Diese sind in Namespaces organisiert, die eine Vielzahl hilfreicher Funktionen bereitstellen: von der Dateieingabe und -ausgabe über die Zeichenfolgenbearbeitung bis hin zur XML-Analyse und zu Webanwendungsframeworks und Windows Forms-Steuerelementen.



Das Programm „Hello, World“ wird mit einer using-Richtlinie gestartet, die auf den System-Namespace verweist. Namespaces bieten eine hierarchische Möglichkeit zum Organisieren von C#-Programmen und -Bibliotheken. Namespaces enthalten Typen und andere Namespaces. Beispiel: Der System-Namespace enthält eine Reihe von Typen, wie etwa die Console-Klasse, auf die im Programm verwiesen wird, und eine Reihe anderer Namespaces, wie etwa IO und Collections. Eine using-Richtlinie, die auf einen bestimmten Namespace verweist, ermöglicht die nicht qualifizierte Nutzung der Typen, die Member dieses Namespace sind. Aufgrund der using-Direktive kann das Programm Console.WriteLine als Abkürzung für System.Console.WriteLine verwenden.

# Arrays in Konsolenprogramm

## Array erstellen



* List<Datentyp> Name = new List<Datentyp>()
  + List
    - Grün weil eine Klasse
  + <Datentyp>
    - Int, float, double …
  + Name
    - Name des Arrays
  + New
    - Ruft ein neues Objekt auf in der Klasse
    - Ruft einen Konstruktor(Methode) auf

## Array bearbeiten



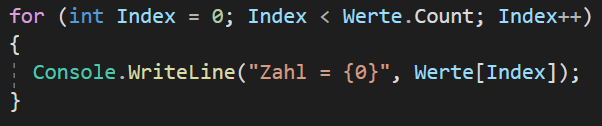
* Name.Add(Wert)
  + Name
    - Name des Arrays
  + .Add
    - Methode
  + (Wert)
    - Wert der ins Array hinzugefügt werden soll

## Array-Daten auslesen



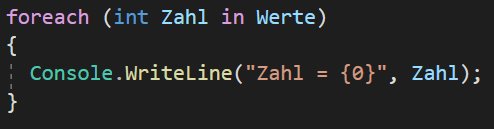
* Console.WriteLine(«Zahl = {0}», Name[Position des Arrays])
  + Console
    - Namespace von Console
  + .WriteLine
    - Methode
  + («Zahl = {0}»,
    - Text der angezeigt wird
    - In geschweifter Klammer ist ein Platzhalter wie in C++ %
  + Name[Position des Arrays]
    - Name und die Zahl angeben vom gewünschten Array

# For-Schleife



* int Index = 0
  + Index startet mit dem Wert 0
* Index < Werte.Count
  + Solange Index kleiner als Werte.Count ist
  + Werte.Count zählt die Werte
* Index++
  + Immer um eines erhöhen

# Foreach-Schleife



# Random Klasse



* Random
  + Klassenname
* Zufall
  + Name definieren
* New
  + Erstellt ein neues Objekt
* Random()

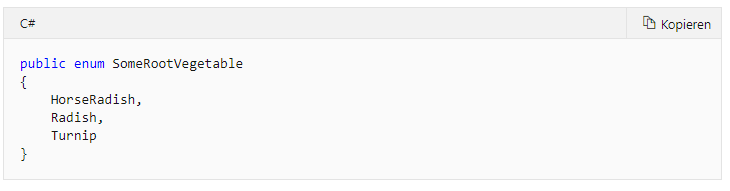
# Redundante Daten bei der Ausgabe vermeiden

# Try and catch

* DivideZeroExeption
  + Durch 0 teilen
* OverflowExeption
  + Zu viele Werte eingegeben
* FormatExeption
  + Ungültiges Format

# Enumeratoren

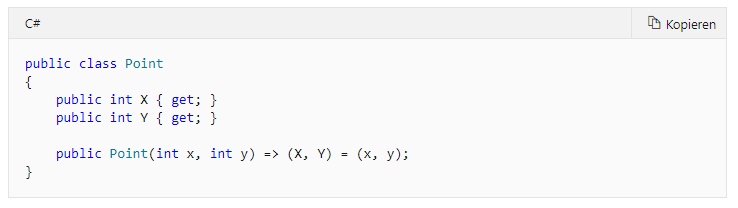
Enumeratoren sind selbstdefinierte Datentypen. Den Variablen diesen neuen Datentypen



# Klassen

Wenn wir den Typ eines Objektes beschreiben, so definieren wir die Eigenschaften (ohne Ausprägung) und die Methoden. Auf diese Weise erhalten wir eine Definition, die für jedes Objekt dieses Typs gilt. Eine Klasse ist also eine abstrakte Beschreibung von einer Gruppe von Objekten die gleichen Eigenschaften und Methoden besitzen.

Klassen sind die grundlegendsten der C#-Typen. Eine Klasse ist eine Datenstruktur, die einen Zustand (Felder) und Aktionen (Methoden und andere Funktionsmember) in einer einzigen Einheit kombiniert. Eine Klasse stellt eine Definition für Instanzen der Klasse bereit, die auch Objekte genannt werden. Klassen unterstützen Vererbung und Polymorphie.



Instanzen von Klassen werden mit dem new-Operator erstellt. Dieser reservierte Speicher für eine neue Instanz, ruft einen Konstruktor zum Initialisieren der Instanz auf und gibt einen Verweis auf die Instanz zurück. Mit den folgenden Anweisungen werden zwei Point-Objekte erstellt und Verweise auf diese Objekte in zwei Variablen gespeichert:



Der von einem Objekt belegte Speicher wird automatisch wieder freigegeben, wenn das Objekt nicht mehr erreichbar ist.

**Besteht aus:**

1. Member
2. Methoden
3. Eigenschaften

## Instanz

Eine Instanz ist eine andere Bezeichnung für ein Objekt.

# Objekte

# Zugriffsmodifizierer

## Public

* + Auf den Typ oder Member kann von jedem Code in der gleichen Assembly oder einer anderen Assembly, die darauf verweist, zugegriffen werden.

## Private

* + Der Zugriff auf den Typ oder Member kann nur über Code innerhalb derselben Klasse oder Struktor erfolgen

## Protected

* + Der Zugriff auf den Typ oder Member kann nur über Code in derselben Klasse erfolgen, die von der Klasse abgeleitet wurde.

## Internal

* + Auf den Typ oder Member kann von jedem Code in der gleichen Assembly zugegriffen werden, jedoch nicht von Code in einer anderen Assembly.