# **1. Wiederholung**

Du sollst aus dem 3. JG folgende Begriffe kennen und erklären können, sowie über deren praktischen Einsatz Bescheid wissen:

* Fehlerbehandlung (Exceptions)
* static
* Generische Listen
* Dictionaries
* Arbeiten mit strings
* Klassenbibliothek (Assembly)
* C
  + Sprachkonstrukte der Programmiersprache C
  + Zeiger
  + Call-by-value and call-by-reference
* Arduino
  + Grundlegende Programmierung
  + I²C
  + SPI
  + Ethernet - UDP - TCP

# **2 C# - Advanced**

## **2.1 Visual Studio 2022**

Ab .NET 6 generiert die Projektvorlage für neue C#-Konsolen-Apps den folgenden Code in der Datei Program.cs

|  |
| --- |
| // See https://aka.ms/new-console-template for more information Console.WriteLine("Hello, World!"); |

Für .NET5 und frühere Versionen generiert die Konsolen-App-Vorlage den folgenden Code:

|  |
| --- |
| using System; namespace MyApp // Note: actual namespace depends on the project name. {     internal class Program     {         static void Main(string[] args)         {             Console.WriteLine("Hello World!");         }     } } |

Die beiden Formen stellen das gleiche Programm dar.

## **2.2 NuGet Pakete**

NuGet ist ein Mechanismus, durch den Entwickler nützlichen Code erstellen, freigeben und verwenden können. Solch ein Code wird in “Paketen” gebündelt, in denen kompilierter Code (z.B. DLLs) zusammen mit anderen Inhalten enthalten sind, die in Projekten benötigt werden, die diese Pakete nutzen.

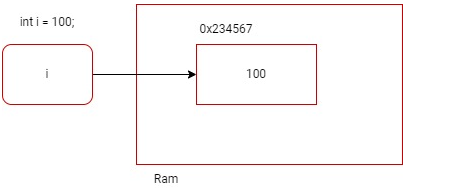
Einfach gesagt: ein NuGet-Paket ist eine einzelne ZIP-Datei mit der Erweiterung .nupkg, die kompilierten Code (DLLs), andere Dateien und ein beschreibendes Manifest enthält, in dem Informationen wie die Versionsnummer des Pakets enthalten sind (sogenannte Meta-Daten),

Entwickler, die Code teilen wollen, erstellen und veröffentlichen Pakete auf einem öffentlichen oder privaten Host. Paketverbraucher erhalten diese Pakete über diese entsprechenden Hosts, fügen diese ihren Projekten hinzu und rufen dann die Funktionalität eines Pakets in ihrem Projektcode auf.

## **2.3 Value types, reference types**

Bei Datentypen kann zwischen value type (Wertetypen) und reference type (Referenztypen) unterschieden werden.

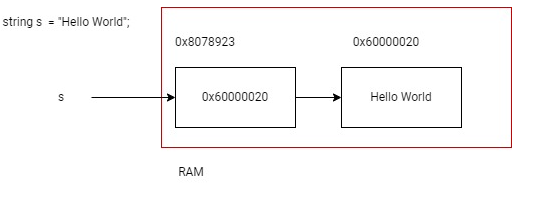
Ein Typ ist dann ein value type, wenn die Stelle im Arbeitsspeicher, die für die Variable vorgesehen ist, auch den Wert enthält.



value types:

* bool
* byte, sbyte, short, int, …
* float, double, decimal
* char
* enums

Bei einem reference type enthält die Stelle, die für die Variable vorgesehen ist, nicht den eigentlichen Wert/Inhalt, sondern “nur” einen Verweis auf die Stelle im Arbeitsspeicher, wo sich die tatsächlichen Daten befinden.



reference types:

* array
* alle Klassen, welche von object abgeleitet sind (somit jede Klasse)
* delegates
* string (Sonderfall: reference type, der sich wie ein value type verhält)

2.4 Nullable datatypes

Einem reference type kann jederzeit der Wert null zugewiesen werden.

Einem value type muss immer ein Wert zugewiesen werden;

Mithilfe des Nullable types kann einem value type der Wert null zugewiesen werden.

Folgende Properties stehen uns zur Verfügung

HasValue

Value

Bei Aufruf der Methode GETVALUEORDEFAULT() wird entweder der Wert (falls vorhanden) oder der Standardwert zurückgeliefert

2.5 Call by Value call by reference

Parameter, die mit ref übergeben werden, sind sogenannte Übergangsparameter, die sowohl in die Methode übergeben werden, als auch eine Änderung der Variablenwerte innerhalb der Methode wieder nach außen geben (call by reference). “Normale” Parameter sind “nur” Eingangsparameter.

Der out-Parameter bekommt beim Methodenaufruf eine Variable übergeben, welche nicht initialisiert werden muss. Innerhalb der Methode zugewiesen wurde