

# Common Language Runtime / C#

## Beispiel 1:

1. Erstellen Sie eine Basisklasse. Die Klasse soll Personen mit Vor- und Nachnamen aufnehmen können.
2. Leiten Sie zwei Klassen für männliche und weibliche Personen von der Basisklasse ab. Implementieren Sie für die neuen Klassen eine Zählvariable mit der Sie die Anzahl der Männer bzw. Frauen erfassen können.
3. Die Ausgabe des Namens bzw. des Geschlechts soll über ein Interface realisiert werden. Erstellen Sie ein entsprechendes Interface und binden Sie es in die beiden neuen Klassen ein. Damit später eine Kontrolle möglich ist, sorgen Sie dafür, dass sich die Ausgaben wirklich unterscheiden.

## Beispiel 2:

Schreiben Sie in C# ein Programm das eine ganze Zufallszahl zwischen 2 und 100 erzeugt und überprüft ob die Zahl eine Primzahl ist. Der Rückgabewert soll ein boolscher Wert sein.

Primzahlen sind Zahlen, die nur durch sich selbst und 1 teilbar sind.

## Beispiel 3:

Erweitern Sie das Programm aus Beispiel 2, sodass die zu überprüfende Zahl von der Command-Line eingelesen wird. Das Ergebnis der Überprüfung soll auf die Console ausgegeben werden.

## Beispiel 4:

Wenn in einer Basisklasse eine Funktion als `virtual` gekennzeichnet ist und in einer abgeleiteten Klasse mit `override` überschrieben wird, wie kann die Basisklasse dann auf die eigene Funktion zugreifen?

→ Beispiel 5 siehe Seite 2

## Beispiel 5:

Gegeben ist folgendes Code-Stück:

```
using System;

public class AgentSmith {
    public static void Main() {
        // Add your code here
    }
}
```

Weiters haben Sie einige Interfaces, Basisklassen und abgeleitete Klassen zur Verfügung (siehe unten). Diese dürfen **NICHT** modifiziert werden! Erweitern Sie nun die **Main()** Methode so, dass durch die Instanzierung der zur Verfügung stehenden Klassen bzw. durch Aufruf der zur Verfügung stehenden Methoden in diesen Klassen folgender Satz entsteht: **Never send a human to do a machine's job.**

Wichtige Tips: Type-Casting kann hilfreich sein!

```
public interface IWord {
    void Print();
}

public interface IWord2 : IWord {
    new void Print();
}

public abstract class Base {
    public Base() { Console.Write(this.GetString()); }
    static Base() { Console.Write("Never "); }
    public virtual void Print() { Console.Write("to "); }
    protected virtual string GetString() { return "llama "; }
    protected static string msg = "send ";
}

public class Derived : Base, IWord {
    static Derived() { Console.Write(Derived.msg); }
    public new virtual void Print() { Console.Write("do "); }
    protected override string GetString() { return "a "; }
}

public sealed class MoreDerived : Derived, IWord {
    public override void Print() { Console.Write("mach"); }
    void IWord.Print() { Console.Write("a "); }
    protected override string GetString() { return "do "; }
}

public sealed class MoreDerived2 : Derived, IWord2 {
    static MoreDerived2() { Console.Write("ine"); }
    public new void Print() { Console.Write("job. "); }
    void IWord2.Print() { Console.Write("job."); }
    protected override string GetString() { return "'s "; }
}

public abstract class Unfinished : Base {
    protected new void Print() { Console.Write("camel "); }
    protected override string GetString() { return "human "; }
}

public class Finished: Unfinished {
}
```