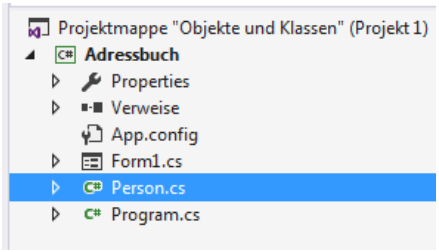
Objekte und Klassen (Teil 2)

Im Teil 2 dieses Themas sollen in einem Projekt Klassen implementiert und mittels Programmcode davon Objekte instanziiert werden.

Implementation einer Klasse

Um in Visual Studio einem Projekt eine Klasse hinzuzufügen, gibt es verschiedene Wege. Ein Weg führt über den Projekt-Explorer. Klicken Sie mit der rechten Maustaste auf den Projekttitel und wählen Sie „Hinzufügen-Neues Element“ und dann den Menüpunkt „Klasse...“.

Im folgenden Fenster geben Sie den Namen der Klasse ein; z.B. **`Person`**. Im Projekt-Explorer erscheint nun ein neuer Eintrag: **`Person.cs`**. Klassen werden innerhalb des Projektordners als eigene Datei mit dem Namen der Klasse und der Erweiterung **`cs`** gespeichert.



Wenn Sie nun den neuen Eintrag per Doppelklick auswählen, erscheint diese Datei zur Bearbeitung. Standardmäßig erhält sie den gleichen Namensbereich wie alle anderen Projektdateien.

Hier kann nun die Implementation der Klasse durchgeführt werden. Üblicherweise werden zu Beginn die privaten Attribute der Klasse festgelegt. Im Anschluss daran erfolgt die Implementation der Konstruktoren bzw. der Methoden.

Nebenstehend ist die verkürzte Implementation der Klasse Person dargestellt. Bei C# gilt eine Besonderheit hinsichtlich des Standard-Konstruktors. Ist dieser **nicht** angegeben, wird er **automatisch** zur Programmlaufzeit erzeugt.

Werden explizit Konstruktoren (auch mit Parametern) angegeben, sind auch **nur** diese benutzbar; es gibt dann **keinen** Standard-Konstruktor mehr!

```
6 Verweise
class Person
{
    private string firstname;
    private string lastname;
    private DateTime birthday;

    0 Verweise
    public string Firstname
    {
        get { return firstname; }
        set { firstname = value; }
    }

    1 Verweis
    public string LastName
    {
        get { return lastname; }
        set { lastname = value; }
    }

    0 Verweise
    public DateTime Birthday
    {
        get { return birthday; }
        set { birthday = value; }
    }

    1 Verweis
    public Person(string s, string n, DateTime b)
    {
        firstname = s;
        lastname = n;
        birthday = b;
    }
}
```

Erstellen eines Objektes

Im Programmcode wird ein Objekt vom Typ der Klasse **`Person`** nun durch einen Konstruktoraufruf, verbunden mit dem Operator **`new`**, erstellt.

Beispiel

```
Person p = new Person("Hans", "Kurz", new DateTime(2017,5,17);
```

Links steht **`Person p`**; es wird eine Objektvariable namens **`p`** vom Typ **`Person`** deklariert. Solch eine Anweisung kann auch allein im Programmcode, gefolgt von einem Semikolon, stehen. Dann wäre diese Objektvariable allerdings noch nicht initialisiert.

In dem obigen Fall wird die Objektvariable jedoch auch initialisiert. Dies macht der Operator **`new`**. Hinter diesem wird nun eine verfügbare Konstruktormethode (bei **`Person`** gibt es nur eine) genutzt. In diesem Fall erwartet die Konstruktormethode drei Parameter: Einen Vornamen, einen Nachnamen, beide vom Typ **`string`**, sowie ein **`DateTime`**-Objekt. Dieses wird als anonymes

Objekt mittels **new** und Konstruktormethode erstellt. Anonym bedeutet, dass dieses Objekt keiner Objektvariablen zugeordnet und damit nicht unmittelbar zugreifbar ist. Ein Zugriff ist in diesem Fall lediglich über das Objekt **p** möglich.

Um diese **List<T>** nutzen zu können, muss der Namespace **System.Collections.Generic** mittels **using**-Anweisung eingebunden sein.

Verwaltung vieler Objekte

Müssen mehrere Objekte vom gleichen Typ erstellt werden, könnte man natürlich für jedes Objekt eine Objektvariable deklarieren (**p1**, **p2**, **p3** …) und diese dann mittels **new** initialisieren. Damit wäre aber bereits zur Entwicklungszeit die Anzahl der Objekte festzulegen und zur Laufzeit nicht änderbar. Auch wäre der Zugriff programmtechnisch sehr aufwändig.

In solchen Fällen kommen sogenannte generische[1] Container-Klassen wie z.B. die Klasse **List<T>**. Das **T** steht für Template (Schablone) und ermöglicht in einer Liste Objekte zu speichern, die einem bestimmten Typ, in diesem Fall dem Typ **Person**, entsprechen.

Solch eine Liste kann folgendermaßen erzeugt werden:

```
List<Person> personen = new List<Person>();
```

Diese Zeile ist prinzipiell genauso aufgebaut, wie die Zeile zur Objekterstellung eines **Person**-Objektes. **List<Person>** ist der Datentyp der Objektvariablen mit dem Namen **personen**.

Mit dem **new**-Operator wird nun eine neue und leere Liste durch die Konstruktormethode **List<Person>()** erstellt.

In dieser Liste **personen** können nun **Person**-Objekte aufgenommen werden. Auf diese Objekte kann dann sehr leicht zugegriffen werden.

Das Hinzufügen von Objekten erfolgt mit einer Methode namens **Add()**, wobei diese Methode das hinzuzufügende Objekt als Parameter erwartet: **personen.Add(p);**

Zum Beispiel mit einer **foreach**-Schleife kann nun auf die einzelnen Objekte in der Liste zugegriffen werden:

```
foreach (Person person in personen)
{
    MessageBox.Show(p.Name);
}
```

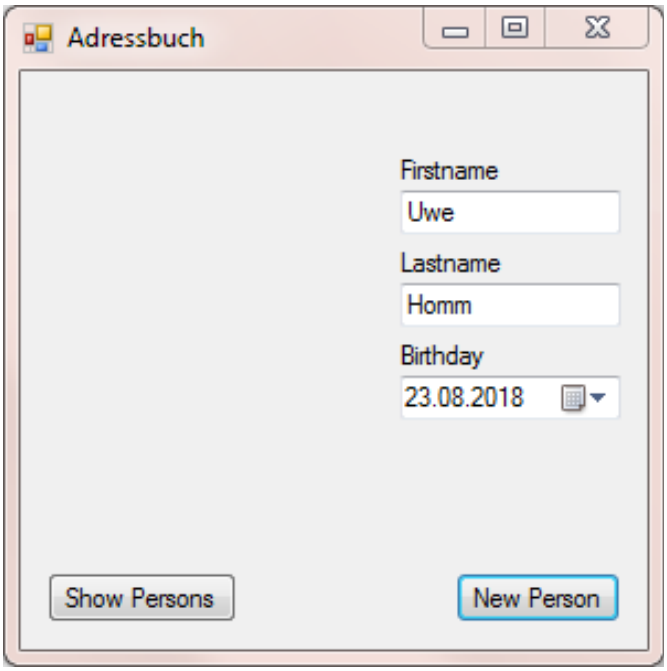
Eine **foreach**-Schleife iteriert alle Objekte eines Containers. Innerhalb der Schleife kann ich dann auf das jeweils aktuelle Objekt lesend zugreifen.

Rechts neben dem in steht der Name des Containers (hier **personen**), links neben dem in wird der Name einer Objektvariablen (hier **person**) sowie deren Datentyp (hier **Person**) angegeben. Beim Iterieren der Liste wird temporär der Objektvariablen **person** das aktuelle Objekt in der Liste zugewiesen.

1 Siehe hierzu **https://de.wikipedia.org/wiki/Generischer_Typ** abgerufen am 17. Mai 2017

Aufgaben

1. Erstellen Sie ein Projekt mit dem Namen **`Adressbuch`**.
2. Fügen Sie dem Formular die Klasse **`Person`** wie oben dargestellt zu.
3. Gestalten Sie das Formular wie dargestellt. Verwenden Sie sinnvolle Bezeichner für die Steuerelemente wie **`textBoxFirstName`**, **`textBoxLastName`**, **`dateTimePickerBirthday`**, **`buttonShowPersons`** und **`buttonNewPerson`**.
4. Tragen sie den dargestellten Programmcode ein und testen Sie das Programm.




```
using System;
using System.Collections.Generic;
using System.Windows.Forms;

namespace Adressbuch
{
    public partial class Form1 : Form
    {
        // Erstelle leere Liste für Person-Objekte
        List<Person> personen = new List<Person>();
        public Form1()
        {
            InitializeComponent();
        }

        private void buttonNewPerson_Click(object sender, EventArgs e)
        {
            // Erstelle Person-Objekt
            Person p = new Person(textBoxFirstname.Text,
                                  textBoxLastName.Text,
                                  dateTimePickerGeburtstag.Value);

            // Füge erstelltes Objekt der Liste hinzu
            personen.Add(p);
        }

        private void buttonShowPersons_Click(object sender, EventArgs e)
        {
            foreach (Person person in personen)
            {
                MessageBox.Show(person.Name);
            }
        }
    }
}
```