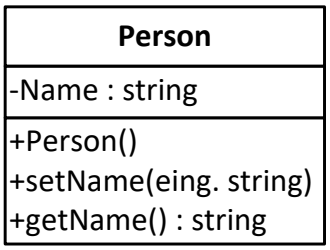
Typisierte Listen zur Verwaltung von Objekten

Modelliert wurde folgendes, vereinfachtes Klassendiagramm zur Erfassung von Personen:



Implementierung der Klasse Person:

```
class Person
{
        private string Name;

        // parametrisierter Konstruktor
        public Person(string name)
        {
            this.Name = name;
        }

        // Ev. kann Name neu gesetzt werden
        public void setName(string name)
        {
            this.Name = name;
        }

        public string getName()
        {
           return this.Name;
        }
}
```

Problemstellung:

Wie können mehrere Personen flexibel und effizient in einem Programm verwaltet werden?

Ein Array ist statisch, die Größe ist vorgegeben und kann nicht mehr geändert werden. Die Verwaltung, d.h. Hinzufügen oder Löschen von Elementen muss programmiert werden.

Gesucht ist eine Datenstruktur, die sich dynamisch zur Laufzeit an die Anzahl der Elemente anpasst, und die die Funktionalität zum Hinzufügen oder Löschen von Elementen schon mitbringt. Optimal wäre eine Liste, der man den Typ der aufzunehmenden Elemente oder Objekte mitgeben kann, so dass der Compiler fehlerhafte Zuweisung schon erkennen kann.

Typisierte Listen zur Verwaltung von Objekten

Diese Funktionalität stellt die typisierte Liste zur Verfügung. Die Deklaration für das Beispiel lautet:

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

Die Typisierung, d.h. welcher Datentyp die Liste aufnehmen kann, wird in den Klammern <> eingetragen. Nach Instanziierung einer solchen Liste ist die Liste zunächst leer. Über die Methode Add() können Objekte eines Typs hinzugefügt werden. Die Eigenschaft Count liefert die Anzahl der Objekte in der Liste. Über die Methode RemoveAt() können Objekte an einer Position gelöscht werden.

Codebeispiel: Beim Drücken des Buttons wird jedes Mal ein neues Objekt Person angelegt und in der Personen-Liste gespeichert.

```
// die Liste erzeugen, Liste ist zu Beginn leer
// diese Zeile steht in der Form1-Klasse ganz oben
List<Person> PersonenListe = new List<Person>();

private void BtnNeu_Click(object sender, EventArgs e)
{
   // eine Person anlegen, wenn Name nicht leer, Name aus Textbox tbName
   if (tbName.Text != "")
   {
      // ein Objekt anlegen
      Person einePerson = new Person(tbName.Text);

      // und dieses Objekt in die Liste ablegen
      PersonenListe.Add(einePerson);

      // anschl. Textbox leeren
      tbName.Text = "";
   }
}
```

Codebeispiel zum Durchlaufen der Liste (for-Schleife und foreach-schleife):

```
for (int i = 0; i < PersonenListe.Count; i++)
{
   // Name der Person holen und in die Listbox eintragen
   listBox.Items.Add(PersonenListe[i].getName());
}
```

oder

```
foreach (Person Element in PersonenListe)
{
   // Name der Person holen und in die Listbox eintragen
   listBox.Items.Add(Element.getName());
}
```