Objektorientierte Programmierung OOP

Objektorientierte Programmierung

- Merkmale:
  - Unterteilung des Codes in funktionelle Einheiten (Klassen)
  - Erstellung (Instanziierung) unabhängiger Objekte aus den Klassen
  - Referenzierung abhängiger Objekte innerhalb von Variablen

- Vorteile:
  - Code wiederverwendbar
  - Bessere Strukturierung und Lesbarkeit
  - erhöhte Sicherheit durch Zugriffsmodifizierer

Namespace und Klassen

Namespace

- dient zur Organisation von Klassen und Methoden
- können geschachtelt werden
- über using-Anweisungen vereinfachter Zugriff auf Code

Klasse

- beschreiben die Struktur von Objekten
- von einer Klasse können mehrere Objekte erzeugt werden
- bestehen aus
  - Feldern(Membervariablen)
  - Eigenschaften (Properties)
  - Funktionen (Methoden)
  - Konstruktor/Destruktor

Namespace und Klassen

```
namespace MeinNamespace
{
    class MeineKlasse
    {
        private int feld;

        public int Eigenschaft { get; set; }

        public void Methode()
        {
        }

        public MeineKlasse()
        {
            feld = 0;
        }

        ~MeineKlasse()
        {
        }
    }
}
```

Felder (Fields)

- Felder sind Variablen eines beliebigen Typs innerhalb einer Klasse
- die Felder sollten generell nur privat und nicht von extern verfügbar sein
- der Zugriff auf Felder sollte über „Get“- und „Set“-Methoden erfolgen
- „speichern“ in der Regel die Daten auf die zugegriffen werden soll

```
public class Person
{
    private string vorname;

    public void SetVorname(string vorname)
    {
        this.vorname = vorname;
    }

    public string GetVorname()
    {
        return this.vorname;
    }
}
```

Eigenschaften (Properties)

- vereinfachen das Anlegen der „Get“- und „Set“-Methoden

```
private string vorname;

public void SetVorname(string vorname)
{
    this.vorname = vorname;
}

public string GetVorname()
{
    return this.vorname;
}
```

```
public string Vorname { get; set; }
```

Eigenschaften (Properties)

```
//öffentliches Lesen und Schreiben erlauben
public string Vorname { get; set; }

//Variable kann nur noch intern gesetzt werden
public string Nachname { get; private set; }

//Variable kann von Außen nur überschrieben werden
public string Geheim { private get; set; }
```

```
private int alter;
0 Verweise
public int Alter
{
    get
    {
        return alter;
    }
    set
    {
        if (value > 0)
        {
            alter = value;
        }
    }
}
```

Standardwerte

- Standardwert einem Feld zuweisen

```
class MeineKlasse
{
    private string meinFeld = "Ich habe einen Standardwert!";
    public string MeineEigenschaft
    {
        get { return meinFeld; }
        set { meinFeld = value; }
    }
}
```

Standardwerte

- Standardwert einer Auto-Eigenschaft zuweisen

```
class MeineKlasse
{
    string MeineEigenschaft { get; set; } = "Ich habe einen Standardwert!";

    public MeineKlasse()
    {
    }
}
```

- Standardwert bei Objekterstellung zuweisen

```
class MeineKlasse
{
    string MeineEigenschaft { get; set; }

    public MeineKlasse()
    {
        MeineEigenschaft = "Ich habe einen Standardwert!";
    }
}
```

Konstruktor

- legt den Startzustand des Objektes nach der Initialisierung fest
- eine Klasse kann mehrere Konstruktoren haben (unterschiedliche Parameter)
- wenn kein Konstruktor festgelegt wurde, wird automatisch ein parameterloser Standard-Konstruktor erzeugt (bspw. „Person()“)

```
public Person(string vorname, string nachname)
{
    this.Vorname = vorname;
    this.Nachname = nachname;
}

public Person(string vorname, string nachname, DateTime geburtstag) : this(vorname, nachname)
{
    this.Geburtstag = geburtstag;
}
```

Objekte (Instanzen)

- ein Objekt ist ein Speicherblock, der nach dem Entwurf einer Klasse aufgebaut wird

- es kann mehrere Objekte einer Klasse geben

- Objekte werden mit dem Schlüsselwort „new“ und einer Funktion des Klassennamens (Konstruktor) erzeugt

```
MeineKlasse meinObjekt = new MeineKlasse();
```