VERERBUNG / POLYMORPHISMUS

Lernziele:

Wir üben hier

1. das Überladen von Konstruktoren
2. das Überschreiben der ToString()-Methode
3. das Überschreiben der Equals()-Methode

Anmerkungen:

ToString():

Folgende Vorbemerkungen sind nützlich, um unten stehende Aufgaben lösen zu können! Betrachten wir eine Ableitungsfolge in der Form (PKW ist von Auto abgeleitet, Limousine von PKW):

```
Auto <- PKW <- Limousine
```

Betrachten wir ein Feld von Autos, das PKWs und Limousinen enthält. Wenn wir die Methode `ToString()` nur in der Klasse Auto überschreiben, so wird diese Methode auch von PKW und Limousine verwendet.

Wollen wir die Methode `ToString()` auch in der Klasse PKW überscheiben, und wollen wir, dass dann die `ToString()`-Methode nur dann von PKW aufgerufen wird, wenn es explizit gewünscht wird, so muss

1. der PKW aus dem Auto-Feld auf PKW gecastet werden und
2. die Methode `ToString()` nicht mit *override*, sondern mit *new* implementiert werden!!!

Bitte testet das anhand dieses Beispiels, bevor ihr die Aufgabe zu lösen versucht!

Equals():

Die Methode Equals() vergleicht das aktuelle Objekt mit dem Objekt in der Parameterliste. Es ist wichtig, dass man als Parameter Objekte vom Typ des aktuellen Objektes übergibt. Die Syntax lautet folgendermaßen:

```
public override bool Equals(object obj)
```

Beispiel: Wir wollen PKWs auf ihre PS-Zahlen hin vergleichen, also

```
PKW pkw1 = new PKW(120);
PKW pkw2 = new PKW(140);
pkw1.Equals(pkw2);
```

und die Methode `Equals()`:

```
public override bool Equals(object obj)
{
    PKW pkw = (PKW)obj;
    if (pkw.m_PS == this.m_PS) return true;
    else return false;
}
```

Aufgabenstellung

Implementiere eine Console-Anwendung, das Personal einer IT-Firma verwaltet. Folgendes Personal ist in einer XML-Datei enthalten (komprimiert), Abb. 1. Hat ein Attribut den Wert -1, so bedeutet das, dass der Wert noch nicht existiert und ist daher gleich 0 zu setzen.

```
<?xml version="1.0"?>
<Firma>
  - <ManagerPers>
      - <Manager Vorname="John">
            Short
            <PersDaten Zulage="120"
                PersNr="3123">2500</PersDaten>
        </Manager>
    </ManagerPers>
  - <AngestelltePers>
      - <Angestellter Vorname="Hans">
            Huber
            <PersDaten PersNr="-1">2000</PersDaten>
        </Angestellter>
      + <Angestellter Vorname="Gert">
    </AngestelltePers>
  - <ArbeiterPers>
      - <Arbeiter Vorname="Ida">
            Hofer
            <PersDaten PersNr="1234">15.40</PersDaten>
        </Arbeiter>
      + <Arbeiter Vorname="Inge">
      + <Arbeiter Vorname="Domi">
    </ArbeiterPers>
</Firma>
```

Abb. 1: XML-Datei

Diese sind in einem Personenfeld m_Person[] der Klasse Firma zu verwalten. Das Personenfeld besteht aus den Klassen Manager, Angestellter, Arbeiter. Finde eine passende Ableitungshierarchie zu diesen Klassen, die sich aus der XML-Struktur ergeben.

1. Erstelle die Klassen für obige Personen (Konstruktoren, Membervariablen, etc.). Welche Membervariablen verwalten die Klassen?
   - Vergleiche deine Lösung anschließend mit der vorgegebenen Klassenbeschreibung. Ändere, wenn nötig deine Klassen entsprechend den Klassen in der Klassenbeschreibung.
   - Implementiere die Methoden nach der Klassenbeschreibung.
2. Erstelle für obige Personen eine Personen-Liste m_Person in der Klasse Firma. Die Reihenfolge der Personen im Feld ist willkürlich!!! Dazu ist

User-Stories

Folgende User-Stories sind an diesem Personenfeld in der Klasse Firma durchzuführen:

1. Gib alle Arbeiter aus (Vorname, Nachname). Firma: OnGetAlleArbeiter(): string[]:: Verwende die ToString()-Methode.

   ALLE ARBEITER (VN, NN)
   Ida Hofer
   Inge Riger
   Domi Nöger
2. Gib alle Angestellten MIT Manager aus (Vorname, Nachname). Firma: OnGetAlleAngestellten(): string[]:: Verwende die ToString()-Methode.

   ALLE ANGESTELLTEN (VN, NN) - MIT MANAGER
   Hans Huber
   John Short

Gert Meier

3. Gib alle Angestellten OHNE Manager aus (Vorname, Nachname). Firma: OnGetNurAngestellte(): string[]:: Verwende die ToString()-Methode.

   ALLE ANGESTELLTEN (VN, NN) - OHNE MANAGER
   Hans Huber
   Gert Meier

4. Gib alle Manager aus (Vorname, Nachname). Firma: OnGetManagers(): List<string>:: Verwende die ToString()-Methode.

   ALLE MANAGER (VN, NN)
   John Short

5. Gib alle Angestellten aus (PersNr, Vorname, Nachname). Firma: OnGetAngeTotal(): List<string>:: Verwende die ToString()-Methode.

   ALLE ANGESTELLTEN (PERSNR, VN, NN)
   0 Hans Huber
   3123 John Short
   2345 Gert Meier

6. Gib alle Manager aus (PersNr, Vorname, Nachname). Firma: OnGetManagersMitPersNr(): List<string>:: Verwende die ToString()-Methode.

   ALLE MANAGER (PERSNR, VN, NN)
   3123 John Short

7. Gib alle Manager aus (PersNr, Vorname, Nachname, Zulage). Firma: OnGetManagersTotal(): List<string>:: Verwende die ToString()-Methode.

   ALLE MANAGER (PERSNR, VN, NN, ZULAGE)
   3123 John Short, Zulage: 120

8. Gib das Gehalt der Managers aus (VN, NN, Gehalt): Firma: OnGetManagerPay(): String[]::

   GEHALT DES MANAGERS
   Gehalt Managers John Short: 2500

9. Gib das Gehalt aller Mitarbeiter MIT Manager (VN, NN, Gehalt) aus. Firma: OnGetAlleAngeGehalt(): List<string>

   GEHALT DER MITARBEITER MIT MANAGER
   Gehalt Angestellter Hans Huber: 2000
   Gehalt **Manager** John Short: 2500
   Gehalt Angestellter Gert Meier: 2000

10. Gib das Gehalt der Angestellten OHNE Manager (Namen, Gehalt) aus. Firma: OnGetNurAngeGehalt(): List<string>

    GEHALT DER MITARBEITER OHNE MANAGER
    Gehalt Angestellter Hans Huber: 2000
    Gehalt Angestellter Gert Meier: 2000

11. Gib den Lohn aller Verkäufer aus (VN, NN, Lohn). Firma: OnGetLohnAllerArbeiter(): List<string>

    LOHN DER ARBEITER
    Lohn Verkäufer Ida Hofer: 15,4
    Lohn Verkäufer Inge Riger: 16,2
    Lohn Verkäufer Domi Nöger: 15,9

Die folgenden Abfragen beziehen sich auf die `Equals`-Methode von `Object`.

✓ Füge diesem Feld ein Arbeiter-Objekt namens "Ilse Huber" mit PersNr = 1414 OHNE Lohn hinzu. Firma: OnAddArbeiter(fn, ln: string, persNr: int):

1. Gib alle Personen mit Nachnamen "Huber" aus. Firma: OnGetAllePersonen(nachname: string): List<string>

   GIB ALLE PERSONEN MIT NAMEN 'Huber' AUS (VN, NN)

Folgende Person hat Nachname 'Huber': Hans Huber
Folgende Person hat Nachname 'Huber': Ilse Huber

2. Gib alle Personen mit Nachnamen "Topfer" aus. Firma: OnGetAllePersonen(nachname: string): List<string>::

   GIB ALLE PERSONEN MIT NAMEN 'Topfer' AUS (VN, NN)
   Es gibt keine Personen mit Nachname 'Topfer'

3. Gib alle Angestellten mit einem Gehalt > 2000 € aus. Firma: OnGetAngeGehaltGroesser(pay: int): List<string>::

   GIB ALLE ANGESTELLTEN MIT EINEM GEHALT > 2000 (PERSNR, VN, NN)
   3123 John Short

4. Gib alle Angestellten mit einem Gehalt > 3000 € aus. Firma: OnGetAngeGehaltGroesser(pay: int): List<string>

   GIB ALLE ANGESTELLTEN MIT EINEM GEHALT > 3000 (PERSNR, VN, NN)
   Es gibt keinen Angestellten mit > 3000 EURO

5. Gib alle Arbeiter mit einem Lohn < 16 € aus. Firma: OnGetLohnArbeiterKleiner(pay: int): List<string>::

   GIB ALLE ARBEITER MIT EINEM LOHN < 16 (PERSNR, VN, NN)
   1234 Ida Hofer
   0 Domi Nöger
   1414 Ilse Huber

6. Gib alle Verkäufer mit einem Lohn < 14 € aus. Firma: OnGetLohnArbeiterKleiner(pay: int): List<string>::

   GIB ALLE ARBEITER MIT EINEM LOHN < 14 (PERSNR, VN, NN)
   1414 Ilse Huber

7. Prüfe, ob es eine Person mit Namen "Hans Huber" gibt. Firma: OnPruefePerson(fn, ln: string): List<string>:: Gib die Antwort dazu aus:

   Es gibt eine Person namens ‘Hans Huber‘.

Klassenbeschreibung

Es sind die Methoden ToString(), Equals() passend zu überschreiben!

Firma: Verwaltet die Personaldaten in einem Personen-Feld.
Eigenschaften:
- ✓ m_Person[]: Personenfeld.
- ✓ m_Dal: DAL

Methoden: Siehe User-Stories

Person: Basisklasse aller Personen in der Firma.
Eigenschaften:
- ✓ m_FirstName: string:: Vorname
- ✓ m_LastName: string:: Nachname

Methoden:
- ✓ verschiedene Konstruktoren
- ✓ ToString(), Equals(), GetHashCode()

Angestellter: Von Person abgeleitet.
Eigenschaften:
- ✓ m_Pay: float:: Gehalt
- ✓ m_PersNo: int: Personalnummer

Methoden:
- ✓ verschiedene Konstruktoren
- ✓ ToString(), Equals(), GetHashCode()

Arbeiter: Von Person abgeleitet.
Eigenschaften:
- ✓ m_Wage: float:: Lohn
- ✓ m_PersNo: int: Personalnummer

Methoden:
- ✓ verschiedene Konstruktoren
- ✓ ToString(), Equals(), GetHashCode()

Manager: Von Angestellter abgeleitet.
Eigenschaften: m_Zulage: float:: Zulage
Methoden:
- ✓ Ein Konstruktor
- ✓ ToString(), Equals(), GetHashCode()