Arbeitsblatt 11. Klasse Person und Dateiaufgabe "Speisekarte"

Alle Aufgaben bauen aufeinander auf, d.h. Sie sollten nur ein einziges Projekt für die gesamte Übung anlegen.

Aufgabe 11.1. Klasse Person

Erstellen Sie eine Klasse "Person" mit privaten Datenfeldern für Personen-Nummer, Vorname, Nachname und Email sowie eine weitere Klasse Program mit der Main-Funktion. Übernehmen Sie den Code und ergänzen Sie.

```
class Person
{
    int persnr;
    string vorname;
    string nachname;
    string email;
}

class Program
{
    static void Main()
    {
        Person person = new Person(12, "Anne", "Meier", "ameier@thn.de");
        Console.WriteLine(person.GetName());
        Console.WriteLine(person);
        person.SetEmail("ameier@th-nuernberg.de"); // Ändern der Email
        Console.WriteLine(person);
    }
}
```

Die Klasse "Person" soll folgende Methoden haben:

a) Parameter Konstruktor: Definieren Sie einen Konstruktor für die Erzeugung von Personen-Objekten über Personalnummer, Vorname, Nachname und Email.

   Stellen Sie im Konstruktor sicher, dass eine Person immer einen Namen und einen Vornamen hat. Konkret sollen Name und Vorname jeweils mindestens zwei Zeichen lang sein. Werfen Sie im Fehlerfall eine Exception. Testen Sie Ihre Implementierung durch einen geeigneten Testfall.

b) Getter-Funktion GetName(), welche den Namen in der Form "Nachname, Vorname" liefert.

c) Getter- und Setter-Funktionen GetEmail() und SetEmail() für das Setzen und Ändern der Email-Adresse.

d) Sorgen Sie durch Implementierung einer ToString()-Methode dafür, dass mit dem Aufruf Console.WriteLine(person) ein Personen-Objekt in folgender Form ausgegeben wird (Hinweis: override):

   12: Meier, Anne; ameier@thn.de

Aufgabe 11.2. Dateien: Speisekarte und Rechnung (Klausur WiSe2017/18)

Für das Kassensystem in einem chinesischen Restaurant ist eine Funktion zur Erstellung der Rechnung zu entwickeln. Grundlage ist eine elektronische Datei "Speisekarte.txt" (im Git), in der die Speisen mit ihren Preisen aufgeführt sind.

Rechts ist ein beispielhafter Inhalt einer solchen Datei zu sehen. In jeder Zeile steht ein Gericht mit folgenden, durch Semikolon getrennten Informationen:

- Nummer des Gerichts, z.B. "H1"
- Name des Gerichts
- Preis

```
S1;Peking Suppe;3,00
S2;Miso Suppe;2,90
V1;Frühlingsrolle;3,10
V2;Krabbenchips (Krupuk);2,80
V3;Gemischter Salat;3,30
H1;Gebratene Nudeln mit Gemüse;8,50
H2;Nasi-Goreng;10,80
H3;Hühnerfleisch Chop-Suey;9,50
H4;Schweinefleisch süß-sauer;10,00
H5;Rindfleisch Gong-Bao;11,50
H6;Ente mit Gemüse;12,50
D1;Gebackene Banane mit Honig;3,00
D2;Frisches Obst;2,50
```

Eine Bestellung wird in dem Kassensystem als String-Feld mit einer Liste von Gerichts-Nummern verwaltet, z.B.

```
string[] bestellung = { "S2", "V1", "H6", "V1", "H4", "H6", "D2" };
```

Die Nummern dürfen wiederholt vorkommen, wenn bspw. mehrere Personen das gleiche Gericht bestellt haben.

Ihre Aufgabe ist es, eine Funktion "Rechnung" zu schreiben, welche ein solches String-Array für die Bestellung als Parameter bekommt und eine Rechnung als Konsolenausgabe wie folgt erzeugt (Formatierung der Nachkommastellen bei den Preisen ist egal):

```
1x Miso Suppe: 2,90
2x Frühlingsrolle: 6,20
1x Schweinefleisch süß-sauer: 10,00
2x Ente mit Gemüse: 25,00
1x Frisches Obst: 2,50
Summe: 46,60
```

Gehen Sie dabei am besten wie folgt vor:

Für jede Zeile in der Datei Speisekarte.txt

Ermittle, wie oft das entsprechende Gericht in der Bestellung vorhanden ist

Wenn in Bestellung vorhanden, gib Anzahl, Name des Gerichts und Zwischensumme aus

Gib am Ende die Gesamtsumme aus

Für diese Aufgabe dürfen Sie die Funktionen der Klassen `StreamReader` bzw. `FileStream` sowie `string.Split()` verwenden (`string[] string.Split(char Separatorzeichen)`).

Sie können davon ausgehen, dass die Datei "Speisekarte.txt" im aktuellen Verzeichnis liegt und alle Einträge gültig sind, d.h. entsprechende Tests sind nicht erforderlich.

Schreiben Sie eine Main-Funktion, in der Sie die Rechnung zu obiger Bestellung ausgeben lassen.

Aufgabe 11.3. Klasse Person: Überprüfung der Email-Adresse (ähnl. Klausur 2023)

Erweitern Sie das Projekt aus Aufgabe 11.1.

Sie dürfen für die geforderte Methode Char.IsLetter verwenden, ansonsten aber **keine** Funktionen der Standard-Bibliothek (insbesondere nicht String.Contains).

a) Erstellen Sie eine Methode EmailGueltig() mit Rückgabe-Typ Bool innerhalb der Klasse Person, die eine einfache Überprüfung einer Email-Adresse vornimmt. Für einen ersten Test stellen Sie nur sicher, dass die Email-Adresse mindestens die Länge 3 hat.

   Achtung: Die Methode sollte static sein. Warum?

b) Verwenden Sie die Methode EmailGueltig, um im Konstruktor und dem Setter "SetEmail()" sicherzustellen, dass nur gültige Email-Adressen zugewiesen werden können. Werfen Sie dafür ggf. eine Exception. Testen Sie Ihre Implementierung durch einen geeigneten Testfall.

c) Erweitern Sie die Methode EmailGueltig und prüfen Sie, dass die übergebene Email-Adresse

   - kein Leerzeichen enthält,
   - mit einem Buchstaben (groß oder klein ohne Umlaute) beginnt sowie
   - ein @-Zeichen und danach einen Punkt für die Domäne enthält.

   Beispiele:

   - "ameier@thn.de" ist gültig
   - "ab.xyz@web" ist ungültig – kein Punkt nach @-Zeichen
   - "@123.de" ist ungültig – kein Buchstabe oder Ziffer am Anfang
   - "thn.de" ist ungültig – kein @-Zeichen
   - "darth vader@empire.com" ist ungültig, enthält Leerzeichen

   Wenn Sie wollen, können Sie Ihre Methode mit den Emails aus diesem Array testen:

```
string[] emails = { "ameier@thn.de", "ab.xyz@web", "@123.de",
                    "thn.de", "darth vader@empire.com" };
```