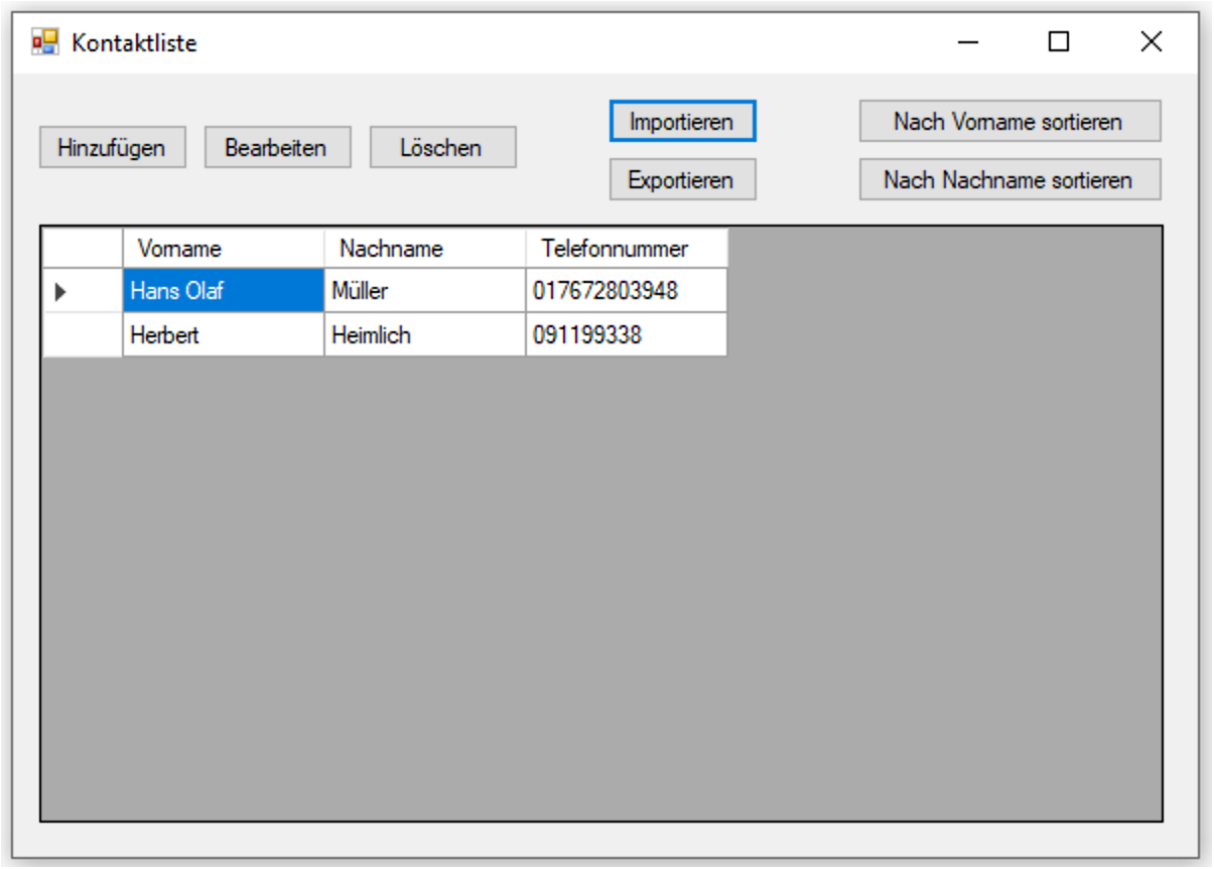
Aufgabe – Kontaktdatei

Schreiben Sie ein kleines Programm, dass Ihnen die Verwaltung Ihrer Kontakte gestattet. Es soll folgende Funktionalitäten enthalten:

- Kontakte aus einer Datei einlesen
- Kontakte in einem DataGridView anzeigen
- Neuen Kontakt hinzufügen
- Kontakt löschen
- Kontakt ändern
- Kontaktliste in Datei speichern
- Kontakte nach Vornamen sortieren (Bonus)
- Kontakte nach Nachnamen sortieren (Bonus)



Konkret gehen Sie dabei folgendermaßen vor:

1. **Benötigte Bibliotheken mit using einbinden**

Binden Sie in Ihrer Form folgende Bibliotheken über using mit ein:

```
using System.IO;
using System.Linq;
```

2. Klasse Kontakt erstellen

Erstellen Sie eine Klasse „Kontakt“.
Diese besitzt die drei Eigenschaften

- Vorname (string)
- Nachname (string)
- Telefonnummer (string)

Hinweis: Nutzen Sie das Kürzel *prop* um Eigenschaften über Codeausschnitte einzufügen.

Implementieren Sie den Standardkonstruktor (Codeausschnitt: *ctor*).
Die Klasse verfügt weiterhin über einen Konstruktor, der ihre drei Eigenschaften initialisiert.

Überschreiben Sie die ToString()-Methode so, dass sie eine Eingabe der Form „Hans Müller – 0175634987“ erzeugt.

Die Klasse hat die Sichtbarkeit „public“. Das ist wichtig, da später Daten vom Typ dieser Klasse zwischen den Formularen ausgetauscht werden.

3. Verwaltung der Kontakte

Erstellen Sie eine globale Eigenschaft List<Kontakt>, die die Kontakte aufnehmen kann. Legen Sie eine weitere globale Eigenschaft vom Typ BindingSource an. Diese stellt die Datenquelle für ihr DataGridView dar:

```
BindingSource BindingSource = new BindingSource();
List<Kontakt> Kontaktliste = new List<Kontakt>();
```

Im FormLoad()-Ereignis legen Sie folgende Datenbindungen an:

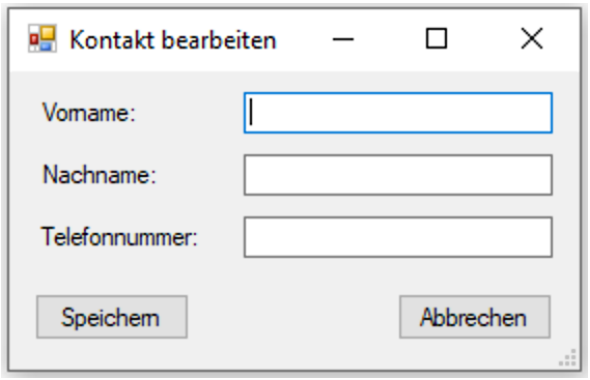
```
private void Form1_Load(object sender, EventArgs e)
{
        BindingSource.DataSource = Kontaktliste;
        grdKontakte.DataSource = BindingSource;
}
```

4. Neuen Kontakt hinzufügen

Über den Button „Kontakt hinzufügen“ rufen Sie ein Eingabeformular zum Hinzufügen eines neuen Kontakts auf. Dieses Eingabeformular besitzt eine Eigenschaft „KontaktInBearbeitung“ mit der Sichtbarkeit public vom Typ „Kontakt“. Über diese Eigenschaft können Sie den Datenaustausch zwischen den Formularen vornehmen.

In diesem Eingabeformular kann der Benutzer nun einen neuen Kontakt anlegen.

Über den Button „Speichern“ wird geprüft, ob alle Eingabefelder gültige Werte beinhalten. Das bedeutet, sie dürfen nicht leer sein. Anschließend wird der neue Kontakt hinzugefügt.



5. Kontakt löschen

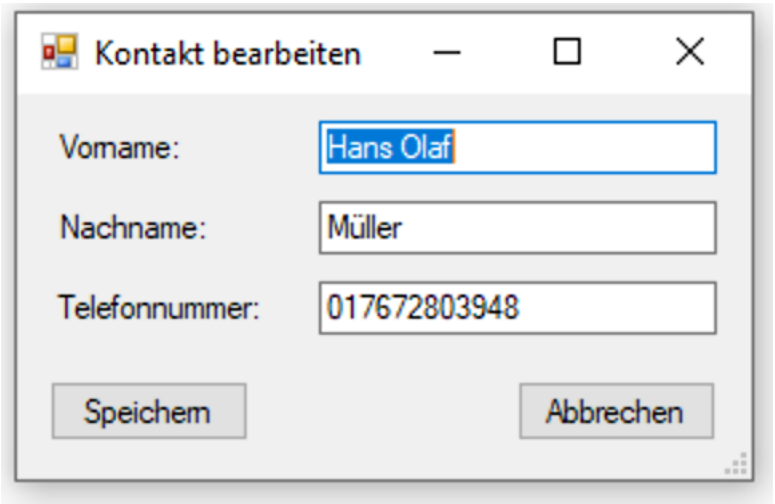
Über den Button „Löschen“ löschen Sie den ausgewählten Kontakt aus ihrer Kontaktliste.

Hinweis: Nach dem Löschvorgang müssen Sie ggfs. die Datenquelle ihrer BindingSource neu an die Kontaktliste binden – das kann notwendig sein, falls Sie zuvor Filter verwendet haben. Verwenden Sie dazu folgende Anweisungen:

```
BindingSource.DataSource = Kontaktliste;
BindingSource.ResetBindings(true);
```

6. Kontakt bearbeiten

Über den Button „Kontakt bearbeiten“ kann der ausgewählte Kontakt bearbeitet werden. Dabei öffnet sich das gleiche Eingabeformular wie bei „Kontakt hinzufügen“. Dort kann der ausgewählte Kontakt bearbeitet werden. Es finden die gleichen Überprüfungen wie unter „Kontakt hinzufügen“ statt (es kann derselbe Code verwendet werden). Bei einem Klick auf speichern wird das Eingabeformular geschlossen und der ausgewählte Kontakt wird aktualisiert. Klickt der Benutzer auf „Abbrechen“, so wird das Formular geschlossen und der Kontakt bleibt unverändert.

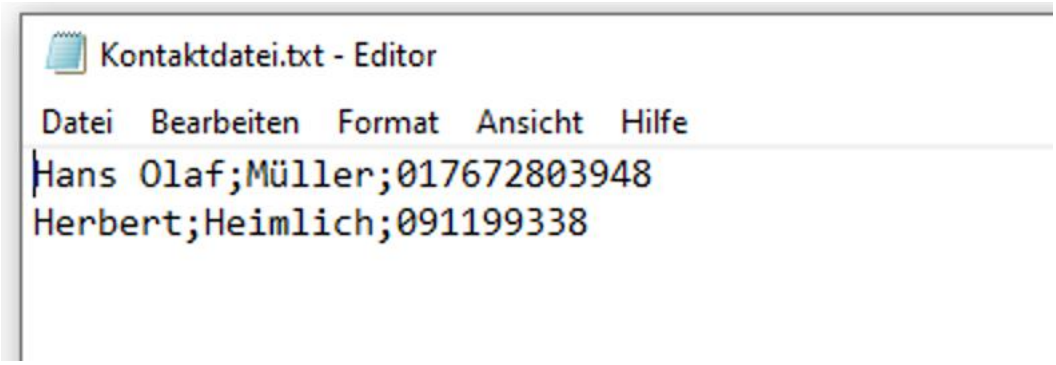


7. Kontakt importieren

Über „Kontakt importieren“ können Kontakte aus einer Textdatei importiert werden. Der Inhalt der Datei ist folgendermaßen formatiert:

Vorname;Nachname;Telefonnummer
Vorname;Nachname;Telefonnummer

In jeder Zeile steht also jeweils ein Kontakt und dessen Eigenschaften sind in der gezeigten Reihenfolge angeordnet und durch Semikola voneinander getrennt.



Über den Button „Kontakt importieren“ soll dem Benutzer ein Auswahldialog zum Öffnen von Dateien angezeigt werden. Der Filter soll auf Textdateien (*.txt) gesetzt werden. Wählt der Benutzer eine Datei aus und klickt auf Öffnen, so werden alle Einträge in dieser Datei eingelesen und der Kontaktverwaltung hinzugefügt.

Lesen Sie dazu die Datei Zeile für Zeile ein. Verwenden Sie die Funktion „Split“ aus der Klasse String, um die Zeile in ihre einzelnen Bestandteile zu zerteilen. Erstellen Sie dann ein neues Objekt vom Typ „Kontakt“ und weisen Sie die Eigenschaften hinzu. Dieses Objekt fügen Sie dann der Kontaktliste hinzu.
Wiederholen Sie das für jede Zeile der Datei.

8. Kontakte exportieren

Exportieren Sie alle in der Kontaktliste enthaltenen Kontakte in eine Textdatei. Der Benutzer soll den Speicherort der Datei frei wählen können. Der Filter des Auswahldialogs soll nur Textdateien (*.txt) zulassen.

Der Inhalt der Datei soll wie unter „Kontakt importieren“ gezeigt formatiert werden. Schreiben Sie alle Einträge der Kontaktliste in dieser Formatierung in die Datei.

Bonus: Vergeben Sie als Standarddateinamen: „<Aktuelles Datum und Uhrzeit> - Kontaktliste.txt“

9. Sortieren

Fügen Sie zwei Buttons hinzu, um die Liste nach Vorname und Nachname zu sortieren.

10. Beenden (Bonus)

Fragen Sie den Benutzer beim Beenden, ob er die Kontaktliste speichern möchte („Möchten Sie speichern?“ Ja – Nein -Abbrechen).

Bei Ja zeigen Sie den Dialog zum Kontaktexport an.
Bei Nein beenden Sie die Anwendung.
Bei Abbrechen kehren Sie zur Anwendung zurück und brechen das Schließen ab.