LU05b - Java Collections

Arten von Collections

Die Klassen des Java Collection Framework in `java.util` stellen vier Familien von abstrakten Datentypen und Funktionen zur Verfügung:

- **Listen** (abstrakte Klasse `List`): geordnete Datenstrukturen auf die man wie auf Felder mit einem numerischen Index zugreifen kann. Im Unterschied zu Feldern (Array) erlauben sie das listentypische Einfügen an einer beliebigen Stelle
- **Mengen** (abstrakte Klasse `Set`): Eine Implementierung der mathematischen Menge. Objekte können nur einmal in einer Menge vorkommen. Man kann prüfen ob bestimmte Objekte in einer Menge enthalten sind. Eine Reihenfolge oder Ordnung in der Menge ist nicht relevant.
- **Verzeichnisse** (abstrakte Klasse `Map`): Verzeichnisse können als verallgemeinerte Felder angesehen werden. Felder erlauben einen direkten zugriff mit einem numerischen Index. Verzeichnisse erlauben die Wahl einer beliebigen Zugriffskriteriums. Man kann zum Beispiel Personen nach ihrem Nachnamen verwalten und eine Person mit Hilfe eines gegebenen Nachnamens aurufen.
- **Warteschlangen** (abstrakte Klasse `Queue`): Warteschlangen sind Listen die nach dem FIFO Prinzip (First In , First Out) aufgebaut sind. Sie verfügen über keinen wahlfreien Zugriff

Im Modul 319 beschränken wir uns auf Listen und deren Verwendung.

Listen - ArrayList

Die Klasse ArrayList ist ein größenveränderliches Array, das im Paket `java.util` zu finden ist.

Der Unterschied zwischen einem Array und einer ArrayList in Java besteht darin, dass die Größe eines Arrays nicht geändert werden kann (wenn Sie Elemente zu einem Array hinzufügen oder daraus entfernen möchten, müssen Sie ein neues Array erstellen). Bei einer ArrayList hingegen können jederzeit Elemente hinzugefügt oder entfernt werden. Auch die Syntax ist etwas anders:

Beispiel

Erstellen Sie ein ArrayList-Objekt mit dem Namen cars, das Strings speichern soll:

```
import java.util.ArrayList; // importiere die ArrayList-
Klasse
ArrayList<String> cars = new ArrayList<>(); // Erstellen
eines ArrayList-Objekts
```

Elemente hinzufügen

Die Klasse ArrayList hat viele nützliche Methoden. Um zum Beispiel Elemente zur ArrayList hinzuzufügen, verwenden Sie die Methode `add()`:

```
import java.util.ArrayList; // Erste Zeile im Java-File

public void run() {
  ArrayList<String> cars = new ArrayList<>();
  cars.add("Volvo");
  cars.add("BMW");
  cars.add("Ford");
  cars.add("Mazda");
  System.out.println(cars);
}
```

Zugriff auf ein Element

Um auf ein Element in der ArrayList zuzugreifen, verwenden Sie die Methode `get()` und verweisen auf die Indexnummer:

```
cars.get(0);
```

Beachten Sie den Index `0` bei `cars.get(0)`, Java beginnt beim Zählen immer bei der Position 0.

Ein Element ändern

Um ein Element zu ändern, verwenden Sie die Methode `set()` und beziehen sich auf die Indexnummer:

```
cars.set(0, "Opel");
```

Ein Element entfernen

Um ein Element zu entfernen, verwenden Sie die Methode `remove()` und verweisen auf die Indexnummer:

```
cars.remove(0);
```

Um alle Elemente in der ArrayList zu entfernen, verwenden Sie die Methode `clear()`:

```
cars.clear();
```

ArrayList Größe

Um herauszufinden, wie viele Elemente eine ArrayList hat, verwenden Sie die Methode `size()`:

```
cars.size();
```

Schleife durch eine ArrayList

Durchlaufen Sie die Elemente einer ArrayList mit einer `for-Schleife` und geben Sie mit der `size()`-Methode an, wie oft die Schleife laufen soll:

```
import java.util.ArrayList; // Erste Zeile im Java-File

public void run() {
    ArrayList<String> cars = new ArrayList<>();
    cars.add("Volvo");
    cars.add("BMW");
    cars.add("Ford");
    cars.add("Mazda");
    for (int i = 0; i < cars.size(); i++) {
      System.out.println(cars.get(i));
    }
}
```

Sie können eine ArrayList auch mit der `for-each-Schleife` durchlaufen:

```
import java.util.ArrayList; // Erste Zeile im Java-File

public void run() {
    ArrayList<String> cars = new ArrayList<>();
    cars.add("Volvo");
    cars.add("BMW");
    cars.add("Ford");
    cars.add("Mazda");
    for (String car : cars) {
      System.out.println(car);
    }
}
```

Andere Typen

Die Elemente in einer ArrayList sind eigentlich Objekte. In den obigen Beispielen haben wir Elemente (Objekte) vom Typ „String“ erstellt. Denken Sie daran, dass ein String in Java ein Objekt ist (kein primitiver Typ). Um andere Typen, wie z.B. int, zu verwenden, müssen Sie eine entsprechende Wrapper-Klasse angeben: Integer. Für andere primitive Typen, verwenden Sie: Boolean für boolean, Character für char, Double für double, usw:

Beispiel: Erstellen Sie eine ArrayList zum Speichern von Zahlen (fügen Sie Elemente vom Typ Integer hinzu):

```
import java.util.ArrayList; // Erste Zeile im Java-File

public void run() {
    ArrayList<Integer> myNumbers = new ArrayList<>();
    myNumbers.add(10);
    myNumbers.add(15);
    myNumbers.add(20);
    myNumbers.add(25);
    for (int number : myNumbers) {  // Java Autoboxing macht
hier aus einem "Integer" selbstständig ein "int"
      System.out.println(number);
    }
}
```

Beispiel: Erstellen Sie eine ArrayList zum Speichern von Personen (eigener Datentyp):

```
import java.util.ArrayList; // Erste Zeile im Java-File
```

```
public void run() {
    ArrayList<Person> personen= new ArrayList<>();
    Person person1 = new Person();
    person1.vorname = "Peter";
    Person person2 = new Person();
    person2.vorname = "Paul";
    personen.add(person1);
    personen.add(person2);

    for (Person person : personen) {
      System.out.println(person.vorname);
    }
}
```

Sortieren einer ArrayList

Eine weitere nützliche Klasse im `java.util`-Paket ist die Klasse Collections, die die Methode `sort()` enthält, um Listen alphabetisch oder numerisch zu sortieren:

```
import java.util.ArrayList;
import java.util.Collections; // Erste zwei Zeilen im Java-
File

public void run() {
    ArrayList<String> cars = new ArrayList<>();
    cars.add("Volvo");
    cars.add("BMW");
    cars.add("Ford");
    cars.add("Mazda");

    Collections.sort(cars);  // Sort cars

    for (String i : cars) {
      System.out.println(i);
    }
}
```

Sortieren einer ArrayList von Ganzzahlen:

```
import java.util.ArrayList;
import java.util.Collections; // Erste zwei Zeilen im Java-
File
```

```java
public void run() {
    ArrayList<Integer> myNumbers = new ArrayList<>();
    myNumbers.add(33);
    myNumbers.add(15);
    myNumbers.add(20);
    myNumbers.add(34);
    myNumbers.add(8);
    myNumbers.add(12);

    Collections.sort(myNumbers);  // Sort myNumbers

    for (int i : myNumbers) {
      System.out.println(i);
    }
}
```

© Kevin Maurizi, Philipp Gressly

From:
https://wiki.bzz.ch/ - **BZZ - Modulwiki**

Permanent link:
https://wiki.bzz.ch/modul/archiv/m319/learningunits/lu05/lu05b-collections

Last update: **2024/03/28 14:07**