OOP - Collections

Hochschule Flensburg
Sommersemester 2017

arnold.willemer@hs-flensburg.de

OOP-Collections: Array

- Daten vermehren sich wie Kaninchen. Und sie sehen alle gleich aus.
- Aufnahme mehrerer Objekte durch ein Array:

```
Bruch[] brueche = new Bruch[7];
```

- Problem: Starre Festlegung der Arraylänge

OOP Collections

- Eine ArrayList kann durch add() gefüttert werden.

```
import java.util.ArrayList;
...
ArrayList brueche = new ArrayList();
for (int i=0; i<7; i++)
{
	brueche.add(new Bruch(i,2));
}
for (Object bruch : brueche)
{
	System.out.println(bruch.toString());
}
```

- Eine ArrayList kann alle Objekte aufnehmen, da alle Objekte Object erweitern.

OOP Collection: Typsicherheit

- Katze einfüllen, Hund herausholen

```
ArrayList tiere = new ArrayList();
for (int i=0; i<3; i++ )
{
      tiere.add(new Katze());
}
Katze katze = (Katze) tiere.get(0);
Hund hund = (Hund) tiere.get(1); // Das gibt Ärger!!!
```

- Merke: Casting ist nicht nur im Fernsehen gruselig!

OOP Collection: Typ-Parameter

- ArrayList mit einem Typ-Parameter versehen:

```
ArrayList<Katze> tiere = new ArrayList<>();
for (int i=0; i<3; i++ )
{
	tiere.add(new Katze());
}
Katze katze = tiere.get(0);
Hund hund = (Hund) tiere.get(1); // Fehler
```

- Hunde können nicht mehr eingefügt oder entnommen werden.

OOP Collection: Interfaces

- Collection ist das Haupt-Interface
  - add(Object)
  - remove(Object)
  - Iterator()
  - isEmpty()

OOP Collection: Interfaces

- List beherrscht den Zugriff über einen Index
  - object = get(index) und set(index, object)

    element = list.get(i); // element = array[i];

    list.set(i, element); // array[i] = element;
  - add(index, object)
  - remove(index)
  - indexOf(object)

OOP Collection: Implementierung

- Collection und List sind Interfaces.
- ArrayList ist eine Implementierung, deren Elemente wie in einem Array nebeneinander liegen.
  - Effizient bei Zugriffen über den Index
  - Nicht optimal bei Einfügungen und Löschungen
- LinkedList ist eine Implementierung als verkettete Liste
  - Nicht optimal bei Zugriff über den Index
  - Effizient bei Einfügungen und Löschungen
- Das Interface gibt die Methoden vor, die Implementierung realisiert EINE Lösung.

OOP Collection: Iterator

- Durchlaufen einer Collection

```
List<Bruch> liste = new LinkedList<>();
for (int i=0; i<3; i++)
{
	liste.add(new Bruch(1, i+1));
}
Iterator<Bruch> iterator = liste.iterator();
while (iterator.hasNext())
{
	Bruch bruch = iterator.next();
	System.out.println(bruch);
}
```

- previous() statt next() möglich

OOP Collection: Sortieren

- Collections hat statische Methode sort()

```
List<Bruch> liste = new ArrayList<>();
// ...
Collections.sort(liste);
```

- sort() erwartet die Implementierung von Comparable

```
public class Bruch implements Comparable<Bruch>
{
	// ...
	@Override
	public int compareTo(Bruch o)
	{
		return … ;
	}
}
```

OOP Collection: FIFO-Schlange

- FIFO durch das Interface Queue definiert.
- Warteschlange bei der Ausgabe von iPhones
- Implementierung durch LinkedList, nicht ArrayList!

```
LinkedList<Bruch> queue = new LinkedList<Bruch>();
queue.add(new Bruch(1,2));
queue.add(new Bruch(1,4));
System.out.println(queue.remove());
System.out.println(queue.remove());
System.out.println(queue.remove());
```

OOP Collection: Stack

- Stapel werden durch das Interface Dequeue definiert und durch LinkedList implementiert, aber nicht ArrayList.

```
LinkedList<Bruch> stack = new LinkedList<Bruch>();
stack.push(new Bruch(1,2));
stack.push(new Bruch(1,4));
while ( ! stack.isEmpty() )
{
	System.out.println(stack.pop());
}
```

OOP Collection: Assoziationen

- Maps verwalten Daten über einen Schlüssel. Landkreise können über ihr Autokennzeichen identifiziert werden.

```
Map<String, Kreis> kennzeichen = new HashMap<String, Kreis>();
kennzeichen.put("HH", new Kreis("Hamburg"));
kennzeichen.put("FL", new Kreis("Flensburg"));
kennzeichen.put("SL", new Kreis("Schleswig"));
Kreis kreis = kennzeichen.get("SL");
System.out.println(kreis.name);
```

- Auslesen einer kompletten Map:

```
for (String key : kennzeichen.keySet()) {
	System.out.println(key+": "+kennzeichen.get(key).name);
}
```