

Einführung in die Programmierung

by André Karge
Übung - Polymorphie Teil 1

letzte Woche

- Listen
- Stack
- Queue

diese Woche

- Besprechung Übungsblatt 7
- Methoden Überladen
- Generics
- Besprechung Übungsblatt 8

Übungsblatt 7

Übungsblatt 7

Aufgaben 1-2

Code Beispiel

Polymorphie

Überladene Methoden - Insel 2.8.10

Beschreibung

- Polymorphie = Vielseitigkeit
- erstes Beispiel: Überladen von Methoden
- schon gesehen: Methoden können selben Bezeichner haben, wenn Signatur unterschiedlich

```
public class Poly {  
    public static void morph() {  
        System.out.println("morph");  
    }  
    public static void morph(String s) { // selber Name, andere Signatur (String)  
        System.out.println(s);  
    }  
    public static void morph(String s, int i) { // selber Name, andere Signatur (String, int)  
        System.out.println(s + i);  
    }  
    public static void morph(int i) { // selber Name, andere Signatur (int)  
        System.out.println(i + 2);  
    }  
}
```

Überladene Methoden - Insel 2.8.10

- beim call der Methoden kommt es darauf an, welchen Typ die Parameter haben

```
Poly.morph();                                // Eingabe: nichts      - Ausgabe: morph
Poly.morph(1);                                // Eingabe: int        - Ausgabe: 3
Poly.morph("hi");                            // Eingabe: String     - Ausgabe: hi
Poly.morph("the cake is a lie ", 42); // Eingabe: String,int - Ausgabe: the cake is a lie 42
```

Generics

Generics - Insel 7

Beschreibung

- Generics sind Platzhalter im Programmcode
- Stichwort: $\langle Typ \rangle$
- besonders Hilfreich für Container
- Ermöglicht es uns die selbe Implementation eines Containers für unterschiedliche Typen zu verwenden
- Beispiel von letzter Woche: einfach verkettete Liste für integer
- was, wenn wir damit unterschiedliche Komplexe Datentypen speichern wollen?

Generics - Insel 7

```
public class Node<T> { // wir geben an, dass die Klasse Node einen Generischen Typen T verwendet
    private T data; // data ist vom Typ T (unspezifiziert)
    private Node<T> next; // Referenz zum folgenden Element
    public Node(T data) { // Argument ist vom Typ T (unspezifiziert)
        this.data = data;
        this.next = null;
    }
    // ...
}
public class List<T> { // wir geben an, dass die Klasse Node einen Generischen Typen T verwendet
    private Node<T> first;
    private Node<T> last;
    public List() {
        this.first = null;
        this.last = null;
    }
    // ...
}
// ...
List<Integer> myList = new List<Integer>(); // Konkrete Spezifikation des generischen Typ (Wrapper)
```

Generics - Insel 7

Wrapper

- Schon aus der Vorlesung bekannt: wir können keine Primitiven Datentypen als Typparameter verwenden
- Daher: Wrapper
- int → Integer, float → Float, usw.

Wichtig

- Man kann sich bei Generics merken: Man nutzt Platzhalter ($<Typ>$) an bestimmten Stellen im Code, den man für verschiedene Typen nutzen möchte
- Bei der Nutzung beschreibt man diesen Platzhalter mit einem konkreten typen ($List<Integer>$)

Generics - Insel 7

Übungsaufgabe (15 min)

1. Schreiben Sie Ihre einfach verkettete Liste von letzter Woche um, dass sie anstelle von integer einen Generic Datentypen abspeichert
2. Testen sie diese generische Liste mit den Typen Integer, String und Float

Übungsblatt 08

Anmerkung

Der Abgabetermin von Übungsblatt 08 wird vom
17.12.2018 11:00
auf den
07.01.2019 11:00
verschoben!

Fragen?