Tutorium 9

16. Dezember 2009

Outline

Mehrdimensionale Arrays

Beispiel: Punktetabelle

- Punkte von Student 3 in Blatt 2 = 6
- Punkte von Student 1 in Blatt 3 = 3
- Punkte von Student 0 in Blatt 1 = 8

Student	Blatt			
	0	1	2	3
0	9	8	7	4
1	7	9	9	3
2	2	4	7	8
3	4	4	6	7
4	1	9	2	8

Mehrdimensionale Arrays

Kompakte Schreibweise:

```
punkte[zeile][spalte]
```

1. Erst Zeilen- dann Spaltenindex
2. Zeilenindizes: $0, \ldots, N-1$
3. Spaltenindizes: $0, \ldots, M-1$

- `punkte[3][2] = 6`
- `punkte[1][3] = 3`
- `punkte[0][1] = 8`

Zeile	Spalte			
	0	1	2	3
0	9	8	7	4
1	7	9	9	3
2	2	4	7	8
3	4	4	6	7
4	1	9	2	8

Aufgabe

1. Implementiere Klasse `Matrix`
2. Attribute:
   1. Matrix mit Elementen vom Typ `int`
   2. Anzahl der Zeilen
   3. Anzahl der Spalten
3. Konstruktor zur Initialisierung der Attribute
4. Methoden
   1. Setzen eines Matrixelements
   2. Zurückliefern eines Matrixelements
   3. Maximum aller Matrixelemente
   4. Anzahl der Matrixelemente mit bestimmten Wert
   5. Stringrepräsentation der Matrix

Call-by-value/Call-by-reference

- Call by value:
  - Bei Aufruf einer Methode wird Wert als Argument übergeben
    $\Rightarrow$ Argument ist primitiver Datentyp
    $\Rightarrow$ hier: `int, double, boolean`
- Call by reference:
  - Bei Aufruf einer Methode wird Referenz auf Objekt als Argument übergeben
    $\Rightarrow$ Argument ist Referenzdatentyp (auch Arrays)

$\Rightarrow$ `DemoCall.java`

Interfaces

- Mehrfachvererbung in Java nicht möglich:
- Ausweg: Interfaces

```
1  class Auto {
2      void fahren() {...}
3  }
4  class Boot {
5      void fahren() {...}
6  }
7  // VERBOTEN in Java:
8  class Amphibienfahrzeug extends Auto, Boot {
9      void fahren() {...}
10     ...
11 }
```

Interfaces

```
1  public interface InterfaceName {
2      // Methodenkoepfe ohne Implementation
3      public typ_1 name_1(Parameterliste_1);
4      ...
5      public typ_k name_k(Parameterliste_k);
6  }
7  public class KlassenName implements InterfaceName {
8      // Attribute
9      // Konstruktoren & Methoden
10     // Implementiert zusaetzlich ALLE Methoden des Interfaces
11     public typ_1 name_1(Parameterliste_1) {...}
12     ...
13     public typ_k name_k(Parameterliste_k) {...}
14 }
```

Interfaces als Referenzdatentyp

Was man so alles machen kann:

```
1 public interface I {...}
2 public class A {...}
3 public class B implements I {...}
```

```
1 // Mehrfachvererbung
2 public class C extends A implements I {...}
3 // Interfaces verwendbar zur Typisierung
4 I b = new B(...);
5 I c1 = new C(...);
6 // Superklassen verwendbar zur Typisierung
7 A c2 = new C(...);
```

Interfaces

Aufgabe: Tarifrechner für Strom

- interface Tarif
- class EPrimo
- class Nuon
- class Vattenfall
- class Tarifrechner
- class DemoStrom