

Vom Überladen und Gesehen werden!

Übergabepunkt z

Florian Sihler ◦ KW 50



Erstellen Sie eine `Circle`-Klasse mit privaten Instanzvariablen für Radius, Flächeninhalt und Umfang. Definieren Sie einen öffentlichen Konstruktor der auf Basis eines Radius die Instanzvariablen initialisiert. Erstellen Sie nun zusätzlich die Methoden:

- *Getter*-Methoden für die Instanzvariablen.
- Eine Methode um die Kreiseigenschaften auf der Kommandozeile auszugeben.
- Eine statische Methode, die für einen Kreisradius den Flächeninhalt zurückgibt.
- Eine statische Methode, die für einen Kreisradius den Umfang zurückgibt.

Bonus: Wäre es möglich zusätzliche Konstruktoren zu definieren, welche entweder nur den Flächeninhalt oder nur den Umfang als Parameter übernehmen?

Präsenzaufgabe - Lösung

- Das wichtigste Beiseite: `Circle.java`
- Wir gehen in die Analyse:

Erstellen Sie eine `Circle`-Klasse mit privaten Instanzvariablen für Radius, Flächeninhalt und Umfang.

Definieren Sie einen öffentlichen Konstruktor der auf Basis eines Radius die Instanzvariablen initialisiert.

```
public class Circle {  
    private double radius;  
    private double area;  
    private double circumference;  
  
    public Circle(double radius) {  
        this.radius = radius;  
        this.area = Math.PI * radius * radius;  
        this.circumference = 2 * Math.PI * radius;  
    }  
}
```

Anstelle von `radius * radius` geht auch `Math.pow(radius,2)` für radius^2 . Die Formeln sollten natürlich bekannt sein 😊.



- Nun gilt es zu sondieren...

```
public class Circle {  
    private double radius;  
    private double area;  
    private double circumference;  
    ...  
    public Circle(double radius) { ... }  
  
    public double getRadius() { return radius; }  
    public double getArea() { return area; }  
    public double getCircumference() { return circumference; }  
}
```

■ Ein printchen Liebe, ein printchen Freude...

```
public class Circle {  
    private double radius;  
    private double area;  
    private double circumference;  
  
    public Circle(double radius) { ... }  
    ...  
  
    public void printProperties() {  
        System.out.println("Radius:␣" + this.radius + "cm");  
        System.out.println("Fläche:␣" + this.area + "cm^2");  
        System.out.println("Umfang:␣" + this.circumference + "cm");  
    }  
}
```

■ Berechne Gefechte:...

```
public class Circle {  
    private double radius;  
    private double area;  
    private double circumference;  
  
    public Circle(double radius) { ... }  
    ...  
  
    public static double computeArea(double radius) {  
        return Math.PI * radius * radius;  
    }  
    public static double computeCircumference(double radius) {  
        return 2 * Math.PI * radius;  
    }  
}
```

- Die statischen Methoden können wir rückwirkend verwenden...

```
public class Circle {  
    private double radius;  
    private double area;  
    private double circumference;  
  
    public Circle(double radius) {  
        this.radius = radius;  
        this.area = Circle.computeArea(radius);  
        this.circumference = Circle.computeCircumference(radius);  
    }  
    ...  
  
    public static double computeArea(double radius) { ... }  
    public static double computeCircumference(double radius) { ... }  
}
```

- Da wir nur einen Parameter übergeben, hilft uns Overload™ leider nicht!

```
public class Circle {  
    ...  
    public Circle(double radius) { ... }  
  
    public Circle(double area) { ... }  
    public Circle(double circumference) { ... }  
    ...  
}
```

Alle drei Konstruktoren würden durch `Circle(double)` identifiziert werden. Das erlaubt Java allerdings **nicht**.

Übungsblatt 7 - Aufgabe 1a)

- Man nennt mich Jörg, den Zeichenkettenspalter ([MethodOverloading.java](#)):

```
public static int[] separateDigits(String number) {  
    int numDigits = number.length();  
    int[] digits = new int[numDigits];  
  
    for(int i = 0; i < numDigits; i++)  
        digits[i] = number.charAt(i) - '0';  
  
    return digits;  
}
```

Alternativ ginge auch `digits[i] = Integer.parseInt(number.substring(i, i + 1))`



Übungsblatt 7 - Aufgabe 1b)

- Theoretisch können wir für die Verarbeitung die alte Methode benutzen:

```
public static int[] separateDigits(int number) {  
    return separateDigits(Integer.toString(number));  
}
```

- Das widerspricht aber der Aufgabenbeschreibung! Deswegen klammern wir uns an den ursprünglichen Wunsch...

Übungsblatt 7 - Aufgabe 1b)

- Deswegen implementieren wir das Ganze so:

```
public static int[] separateDigits(int number) {  
    int numDigits = (int) (Math.log10(number) + 1);  
    int[] digits = new int[numDigits];  
  
    for (int i = numDigits - 1; i >= 0; i--) {  
        digits[i] = number % 10;  
        number /= 10;  
    }  
    return digits;  
}
```

Betrachte $n = 476 = 4 \cdot 10^2 + 7 \cdot 10^1 + 6 \cdot 10^0$.

$n/10 = 4 \cdot 10^{2-1} + 7 \cdot 10^{1-1} + 6 \cdot 10^{0-1} = 47.6$

$n \bmod 10 = 4 \cdot 10^2 + 7 \cdot 10^1 + 6 \cdot 10^0 = 6$

$\log_{10} n = \log_{10}(4 \cdot 10^2 + 7 \cdot 10^1 + 6 \cdot 10^0) \approx_{\max} \log_{10}(4 \cdot 10^2) \approx 2.6$

Kleinste Ziffer: $z = n \bmod 10$, Zahl ohne kleinste Ziffer: $z = \lfloor \frac{n}{10} \rfloor$. Anzahl

Ziffern: $z = \lfloor \log_{10} |n| + 1 \rfloor$ (beachte: 0). Was ist mit 100? Mit 1000?



Übungsblatt 7 - Aufgabe 1c)

- Es ist **nicht** möglich, die zweite Methode mit `public static byte[] separateDigits(String number)` zu erzeugen.
- Warum?
- Die Signatur `separateDigits(String)` enthält (in Java) *nicht* den Rückgabotyp. Daher, sind für Java beide Methoden nicht zu unterscheiden. Dies wird vom Java-Standard verboten.

Method-Signature:

Name und Parametertypen

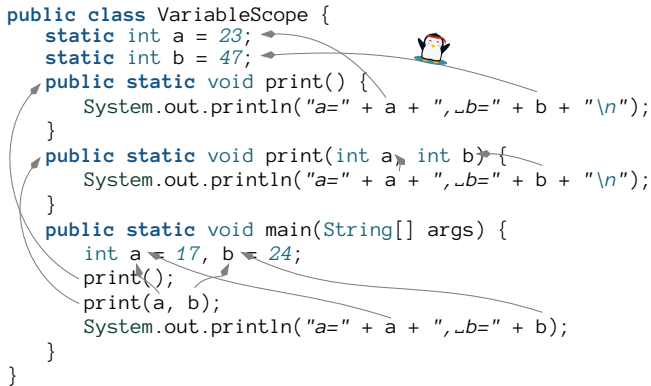
Overloading / Überladung:

Gleicher Name, andere Signatur

Übungsblatt 7 - Aufgabe 2

- Die Ausgabe eines Programms verstehen:

```
public class VariableScope {  
    static int a = 23;  
    static int b = 47;  
    public static void print() {  
        System.out.println("a=" + a + ", b=" + b + "\n");  
    }  
    public static void print(int a, int b) {  
        System.out.println("a=" + a + ", b=" + b + "\n");  
    }  
    public static void main(String[] args) {  
        int a = 17, b = 24;  
        print();  
        print(a, b);  
        System.out.println("a=" + a + ", b=" + b);  
    }  
}
```

A diagram illustrating variable shadowing. A penguin icon is positioned above the code. Arrows point from the penguin to the static variables 'a' and 'b' in the class scope. Other arrows point from the local variables 'a' and 'b' in the 'main' method to the 'print()' and 'print(a, b)' calls, indicating that the local variables are used when they are in scope.

Shadowing / Überschatten:
[vielseitig, z.B.] lokale Variable mit
gleichem Namen wie globale.

a=23, b=47↔

↔

a=17, b=24↔

↔

a=17, b=24↔

↔

- Wir erhalten „a=23, b=47↵↵“ durch den Aufruf von `print()`, welcher die Werte der zwei statischen globalen Variablen ausgibt, da diese nicht **überschattet** werden.
- „a=17, b=24↵↵“ erhalten wir durch `print(int a, int b)` welches durch die Namen der Parameter die globalen Variablen a und b überschattet. Die Parameter sind an die lokalen Variablen in `VariableScope::main` gebunden, wo sie ebenfalls die globalen Variablen **überschatten**.
- Analog erhalten wir „a=17, b=24↵↵“ durch die eben angesprochene **Überschattung** in `VariableScope::main`.

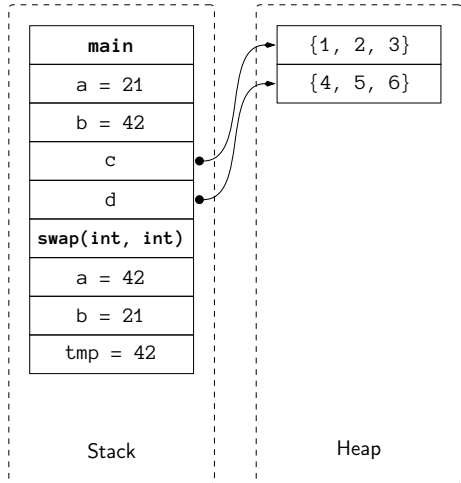
Ich verwende „↵“ um eine neue Zeile zu markieren. Auch dann, wenn ich sie nicht im Text formatiere.



- Die zwei Integer Variablen `a` und `b` haben nach dem Aufruf der `swap` Methode die selben Werte wie davor. Das liegt daran, dass in Java alle Parameter *by value* übergeben werden, somit findet der Tausch der Werte in der `swap` Methode auf einer Kopie der Variablen statt.
- Die Werte der Array Variablen `c` und `d` sind nach dem Aufruf der `swap` Methode jedoch getauscht. Dies liegt daran, dass hier nur die Referenz auf das Array *by value* übergeben wird, welche nicht verändert werden kann. Beim Zugriff auf die Arrays, werden jedoch die originalen Werte und keine Kopien geliefert, und somit können diese verändert werden.

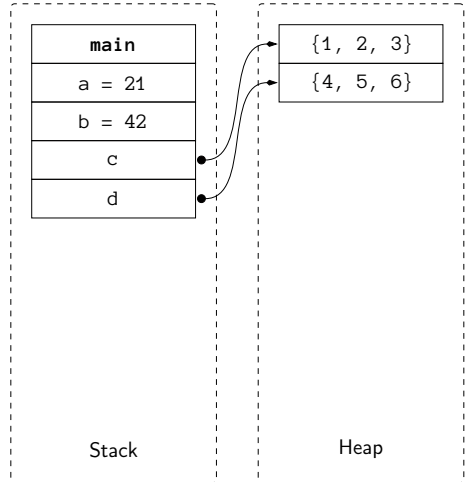
Übungsblatt 7 - Aufgabe 3

```
public class SwapFunction {  
    public static void swap(int a, int b) {  
        int tmp = b; b = a; a = tmp;  
    }  
    public static void swap(int[] a, int[] b) {  
        if(a.length != b.length) return;  
        for(int i = 0; i < a.length; i++) {  
            int tmp = b[i]; b[i] = a[i]; a[i] = tmp;  
        }  
    }  
    public static void main(String[] args) {  
        int a = 21; int b = 42;  
        int[] c = {1, 2, 3};  
        int[] d = {4, 5, 6};  
        swap(a, b); // → swap(int, int)  
        swap(c, d);  
    }  
}
```



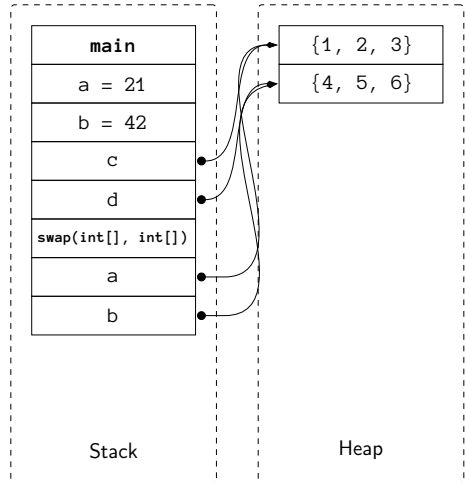
Übungsblatt 7 - Aufgabe 3

```
public class SwapFunction {  
    public static void swap(int a, int b) {  
        int tmp = b; b = a; a = tmp;  
    }  
    public static void swap(int[] a, int[] b) {  
        if(a.length != b.length) return;  
        for(int i = 0; i < a.length; i++) {  
            int tmp = b[i]; b[i] = a[i]; a[i] = tmp;  
        }  
    }  
    public static void main(String[] args) {  
        int a = 21; int b = 42;  
        int[] c = {1, 2, 3};  
        int[] d = {4, 5, 6};  
        swap(a, b);  
        swap(c, d);  
    }  
}
```



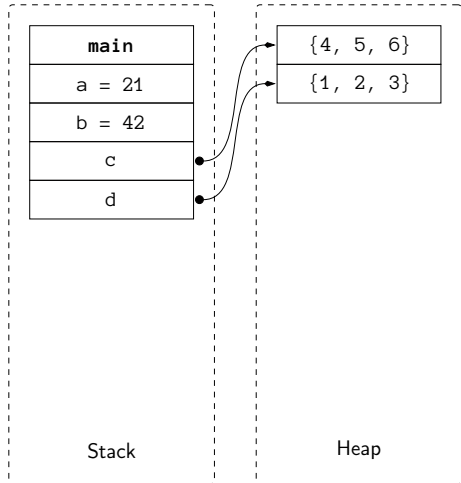
Übungsblatt 7 - Aufgabe 3

```
public class SwapFunction {  
    public static void swap(int a, int b) {  
        int tmp = b; b = a; a = tmp;  
    }  
    • public static void swap(int[] a, int[] b) {  
        if(a.length != b.length) return;  
        for(int i = 0; i < a.length; i++) {  
            int tmp = b[i]; b[i] = a[i]; a[i] = tmp;  
        }  
    }  
    public static void main(String[] args) {  
        int a = 21; int b = 42;  
        int[] c = {1, 2, 3};  
        int[] d = {4, 5, 6};  
        swap(a, b);  
        swap(c, d); // → swap(int[], int[])  
    }  
}
```



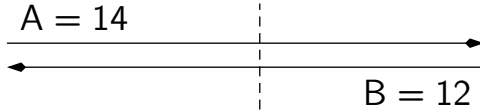
Übungsblatt 7 - Aufgabe 3

```
public class SwapFunction {  
    public static void swap(int a, int b) {  
        int tmp = b; b = a; a = tmp;  
    }  
    public static void swap(int[] a, int[] b) {  
        if(a.length != b.length) return;  
        for(int i = 0; i < a.length; i++) {  
            int tmp = b[i]; b[i] = a[i]; a[i] = tmp;  
        }  
    }  
    public static void main(String[] args) {  
        int a = 21; int b = 42;  
        int[] c = {1, 2, 3};  
        int[] d = {4, 5, 6};  
        swap(a, b);  
        swap(c, d);  
    }  
}
```





Alice



Bob



Eve



Let's talk about december 22nd