Programmieren in Java

Vorlesung 02: Methoden

Prof. Dr. Peter Thiemann

Albert-Ludwigs-Universität Freiburg, Germany

SS 2017

Inhalt

Verwendung von java.util.Scanner

- vordefinierte Klasse zum Lesen von Eingaben
- Scanner sc = **new** Scanner(System.in);
  erzeugt einen Scanner, der mit der Konsole verbunden ist
- **int** i = sc.nextInt();
  - liest die nächste Integer Zahl von der Eingabe
  - **white space** wird dabei überlesen
  - **white space** umfasst *alle* Formattierungszeichen (space, newline, tab, usw)
- genauso funktionieren nextBoolean(), nextByte(), nextFloat(), nextDouble(), nextLong(), nextShort()
- String s = sc.next()
  liefert einen String

Beispiel java.util.Scanner

```
1  package readthreedoubles;
2  import java.util.Scanner;
3  public class Main {
4     public static void main(String[] arg) {
5        Scanner sc = new Scanner(System.in);
6        double d1 = sc.nextDouble();
7        double d2 = sc.nextDouble();
8        double d3 = sc.nextDouble();
9        System.out.println("d1 = " + d1);
10       System.out.println("d2 = " + d2);
11       System.out.println("d3 = " + d3);
12    }
13 }
```

Beispiel java.util.Scanner

```
1  package readthreedoubles;
2  import java.util.Scanner;
3  public class Main {
4    public static void main(String[] arg) {
5      Scanner sc = new Scanner(System.in);
6      double d1 = sc.nextDouble();
7      double d2 = sc.nextDouble();
8      double d3 = sc.nextDouble();
9      System.out.println("d1 = " + d1);
10     System.out.println("d2 = " + d2);
11     System.out.println("d3 = " + d3);
12   }
13 }
```

Lokale Variable vs Felder

Definiere Variable **lokal** innerhalb von Methoden!

Vgl. sc, d1, d2, d3

Außerhalb werden **Felder** definiert (später)

Problem mit Scanner

- Scanner übernimmt die Spracheinstellung des Rechners
- Konvention der Aufgaben: Sprache = US english
- Probleme beim Einlesen von Gleitkommazahlen mit Spracheinstellung "deutsch"
- Abhilfe: **import** java.util.Locale;

```
1  // ersetze
2  Scanner sc = new Scanner(System in);
3  // durch
4  Scanner sc = new Scanner(System.in).useLocale(new Locale("en", "US"));
```

Conditionals / Aufgabe int-to-string

P. Thiemann, L. Fennell Sommersemester 2017

Programmieren in Java

`http://proglang.informatik.uni-freiburg.de/teaching/java/2017/`

int-to-string

Integer in String umwandeln

Woche 02 Aufgabe 1/7

Herausgabe: 2017-04-25
Abgabe: 2017-05-12

Achtung: beachten Sie unbedingt die allgemeinen Hinweise zur Abgabe auf der Homepage.

Project `int-to-string`
Package `inttostring`
Klassen

`Main`
`public static void main(String[])`

Die Aufgabe besteht darin eine über `stdin` eingegebene Zahl vom Typ `int` einzulesen und diese in einen String umzuwandeln. Falls eine korrekte Integer Zahl n eingegeben wurde soll

`"Found int:` n`"`

auf `stdout` auszugeben werden, in jedem anderen Fall

`"No int"`

.

Lösungsmuster mit java.util.Scanner

```
1  package inttostring;
2  import java.util.Scanner;
3  public class Main {
4    public static void main(String[] args) {
5      // get a scanner
6      if (sc.hasNextInt()) {
7        int n = sc.nextInt();
8        // do s.t. with integer n
9      } else {
10       // no integer available
11     }
12   }
13 }
```

Weitere Lösungen

- sind möglich: mit Integer(), Integer.parseInt(), aber sie erfordern den Umgang mit Exceptions.
- "ganz von Hand" benötigt Schleife: später

Methoden

- Für viele Aufgaben sollen **Methoden** erstellt werden.
- Meist ist der Name vorgegeben, z.B. gallonToLiter()
- Sowie der Typ der Eingaben und der Ausgaben, also
  double gallonsToLiter(**double** gallons)
- Hierfür muss eine neue Methode (neben oder anstelle der main Methode) erstellt werden.

Beispiel Methode gallonsToLiter()

```
1 class Main {
2   public static final double GALLONS_PER_LITER = 0.264172; // lt Google
3   public static double gallonsToLiter(double gallons) {
4     return gallons / GALLONS_PER_LITER;
5   }
6 }
```

- Zeile 2 definiert eine **Konstante** (erkennbar am **final**)
- Zeile 3–5 definiert Methode gallonsToLiter
- **return** definiert den Rückgabewert

Sinnvoll zum Testen: main Methode

```
1  class Main {
2    public static final double GALLONS_PER_LITER = 0.264172; // It Google
3    public static double gallonsToLiter(double gallons) {
4      return gallons / GALLONS_PER_LITER;
5    }
6    public static void main(String[] args) {
7      Scanner sc = new Scanner(System.in);
8      double gallons = sc.nextDouble();
9      double liters = gallonsToLiter(gallons);
10     System.out.println(gallons + " gallons = " + liters + " liters");
11   }
12 }
```

Exkurs: Dokumentation mit Javadoc

```
1   public class Main {
2     public static final double GALLONS_PER_LITER = 0.264172;
3
4     /**
5      * converts gallons into the equivalent amount in liters.
6      * @param gallons amount to convert.
7      * @return equivalent amount in liters.
8      */
9     public static double gallonsToLiters(double gallons) {
10        return gallons / GALLONS_PER_LITER;
11    }
12  }
```

- @param dokumentiert einen Parameter
- @return dokumentiert den Rückgabewert

Geschachtelte Conditionals

Coffee Temperature

Given the temperature in a cup of coffee, return “too hot” if the temperature exceeds 60 degrees, “just right” if the temperature is between 50 and 60 degrees, and “too cold” if it is below 50.

- Hier müssen mehrere Bedingungen getestet werden
- Conditionals “if” muss geschachtelt werden.

Beispiel Coffee Temperature

```
1  public class Main {
2      public static final double COLD_COFFEE_TEMP = 50;
3      public static final double HOT_COFFEE_TEMP = 60;
4      public static String coffeeTemperature(double temp) {
5          if (temp < COLD_COFFEE_TEMP) {
6              return "too cold";
7          } else if (temp > HOT_COFFEE_TEMP) {
8              return "too hot";
9          } else {
10             return "just right";
11         }
12     }
13 }
```

Mehrwege Conditional

Japanese Numbers

Translate a number in the range 1-999 to Japanese numbers (Kanji).
There are characters for 1-10, 100 (1000, 10000).

- Japanische Zahlen sind sehr einfach
- 11 = (10, 1), 12 = (10,2)
- 20 = (2, 10), 21 = (2, 10, 1), 30 = (3, 10)
- 345 = (3, 100, 4, 10, 5)

Mehrwege Conditional - switch

Abbildung von 1-9 auf die entsprechenden Kanji-Zeichen

```
1   public static char jdigit(int i) {
2       switch (i) {
3           case 1: return '\u4e00';
4           case 2: return '\u4e8c';
5           case 3: return '\u4e09';
6           case 4: return '\u56db';
7           case 5: return '\u4e94';
8           case 6: return '\u516d';
9           case 7: return '\u4e03';
10          case 8: return '\u516b';
11          case 9: return '\u4e5d';
12          default: return ' ';
13      }
14  }
```

Alternative Lösung

```
1  // string with japanese numbers from 1 to 9
2  public static final String j1To9 =
3    " \u4e00\u4e8c\u4e09\u56db\u4e94\u516d\u4e03\u516b\u4e5d";
4  public static char jdigit(int i) {
5    if (i > 0 && i < 10) {
6      // return character at position i, 0-based
7      return j1To9.charAt(i);
8    } else {
9      return ' ';
10   }
11 }
```

Fragen