Scanner: Syntax

- `Scanner` aus Bibliothek `java.util` importieren
  - `import java.util.Scanner;`
- `Scanner`-Objekt konstruieren mit Angabe der Eingabequelle
  - `Scanner name = new Scanner(System.in);`

Ein neues Objekt (eines nicht-primitiven Typs) muss mit dem Schlüsselwort `new` erstellt werden

Eingabequelle

Scanner: Eine Auswahl von Methoden

Aufruf `Scanner`-Methoden mit Punktnotation («dot notation»)

Method	Description
`nextInt()`	reads an `int` from the user and returns it
`nextDouble()`	reads a `double` from the user …
`next()`	reads a one-word `String` from the user …
`nextLine()`	reads a one-line `String` from the user …

```
Scanner myConsole = new Scanner(System.in);
int alter = myConsole.nextInt();
System.out.println("Ihr Alter: " + alter);
```

Scanner: Beispiel 1

```
import java.util.Scanner;
public class UserInputExample {
  public static void main(String[] args) {
    Scanner myConsole = new Scanner(System.in);
    System.out.print ("Wie alt sind Sie? "); // Prompt
    int alter = myConsole.nextInt();
    int jahre = 65 - alter;
    System.out.println(jahre + " Jahre bis zur Pensionierung");
  }
}
```

```
Wie alt sind Sie? 32
33 Jahre bis zur Pensionierung
```

Scanner: Beispiel 2

```
import java.util.Scanner;
public class UserInputExample {
  public static void main(String[] args) {
    Scanner myConsole = new Scanner(System.in);
    System.out.print("Zwei Zahlen bitte: "); // Prompt
    int a = myConsole.nextInt();
    int b = myConsole.nextInt();
    int product = a * b;
    System.out.println(product);
  }
}
```

mehrere Zahlen einlesen

```
Zwei Zahlen bitte: 6 7
42
```

Beispiel: Falscher Input

```
System.out.print("Wie alt sind Sie? ");
int alter = myConsole.nextInt();
```

```
Wie alt sind Sie? Zweiunddreissig
Exception in thread "main" java.util.InputMismatchException
        at java.util.Scanner.throwFor(Scanner.java:964)
        at java.util.Scanner.next(Scanner.java:1619)
        at java.util.Scanner.nextInt(Scanner.java:2284)
        ...
```

Falscher Input-Typ

Eingabeelement («token»)

Folge von Zeichen, die vom Scanner **am Stück** gelesen werden

- Scanner muss wissen, wo Beschreibung aufhört (und anfängt)
- Separiert durch Zwischenraum («whitespace»)
  - Leerzeichen («space», «blank»), Tabulator («tab»), neue Zeile («new line»)
- Kann kontextabhängig sein
  - `45 18 $2.50 hello world`
  - `$2.50 "hello world" 1.456E12 "45 18"`
- Erwartete Zeichen hängen von Methode ab
  - Z.B. Unterschied zwischen `nextInt()` und `nextDouble()`
- Braucht Beschreibung von legalen Zeichenfolgen (≈ EBNF)
  - Fehlermeldung (zur Laufzeit), falls Token nicht den richtigen Typ/die richtige Form hat

Beispiel: `next()`

```
Scanner console = new Scanner(System.in);
System.out.print("What is your name? ");
String name = console.next();
name = name.toUpperCase();
System.out.println(name + " has " + name.length() + " letters ");
```

Liest ein Wort bis zum Zwischenraum

```
What is your name? Minnie
MINNIE has 6 letters
What is your name? Minnie Mouse
MINNIE has 6 letters
What is your name? "Minnie Mouse"
"MINNIE has 7 letters
```

Beispiel: `nextLine()`

```
Scanner console = new Scanner(System.in);
System.out.print("What is your name? ");
String name = console.nextLine();
name = name.toUpperCase();
System.out.println(name + " has " + name.length() + " letters ");
```

Liest eine Zeile

```
What is your name? Minnie
MINNIE has 6 letters
What is your name? Minnie Mouse
MINNIE MOUSE has 12 letters
What is your name? "Minnie Mouse"
"MINNIE MOUSE" has 14 letters
```

Zufallszahlen («random numbers»)

random {adjective}
zufällig
willkürlich [wahllos]
zufallsbedingt
Zufalls-
dem Zufall überlassen

(aus EN-DE-Wörterbuch)

Kann z.B. anstelle von Benutzereingabe genutzt werden.

(Pseudo-)Zufallszahlengenerator `Random`

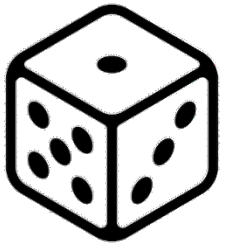
- `Random` aus Bibliothek `java.util` importieren
  - `import java.util.Random;`
- `Random`-Objekt konstruieren
  - `Random name = new Random();`
- Verschiedene Methoden für unterschiedliche Modi

Method	Description
`nextInt()`	returns a ”random integer”
`nextInt(n)`	returns a ”random” integer in the range [0, `n`) in other words: returns a number from {0, …, `n` – 1} at ”random”
`nextDouble()`	returns a “random real” number in the range [0.0, 1.0)

Häufige Szenarien

- Zufällige ganze Zahl aus [`1`, `n`]
  - `1 + rand.nextInt(n)`
- Zufällige ganze Zahl aus [`min`, `max`]
  - `min + rand.nextInt(max - min + 1)`
- Zufällige reelle Zahl aus [`min`, `max`)
  - `min + rand.nextDouble() * (max - min)`
- Zufälliges Objekt aus einer Menge von `n` Objekten
  - Beliebige Nummerierung der Objekte mit [`1`, `n`] oder [`0`, `n-1`]

Beispiel 1: Würfel



```java
import java.util.Random;
public class DiceThrow {
  public static void main(String[] args) {
    Random rand = new Random();
    int face = 1 + rand.nextInt(6);
    System.out.println("You threw a " + face);
  }
}
```

Beispielausgabe: `You threw a 6`

Beispiel 2: Zufällige Note in [1.0, 6.0)

```
import java.util.Random;
public class RandomGrade {
  public static void main(String[] args) {
    Random rand = new Random();
    double r = 5.0 * rand.nextDouble(); // 0 <= r < 5
    double grade = 1.0 + r;
    System.out.println("You got a " + grade);
  }
}
```

Beispielausgabe: `You got a 4.2`

Beispiel 3: Zufällige gerade Zahl zwischen 4 und 12

```
import java.util.Random;
public class RandomGrade {
  public static void main(String[] args) {
    Random rand = new Random();
    int r = 2 * rand.nextInt(5) + 4; // 4, 6, …, 12
    System.out.println(r);
  }
}
```

Beispielausgabe: 6