Herzlich Willkommen!

```
public class Hello {
    public static void main (String[] args) {
        System.out.println("Hello World!");
    }
}
```

```
$ javac Hello.java
$ java Hello
Hello World!
```

```
import java.util.Random;
public class UnbekannteZahl {
    private int zahl;
    public UnbekannteZahl (int grenze) {
        zahl = (new Random()).nextInt() % grenze;
        if (zahl < 0) {
            zahl = zahl + grenze;
        }
    }
    public boolean gleich (int vergleichszahl) {
        return (zahl == vergleichszahl);
    }
    public boolean kleiner (int vergleichszahl) {
        return (zahl < vergleichszahl);
    }
}
```

Zahlenraten für Glückspilze

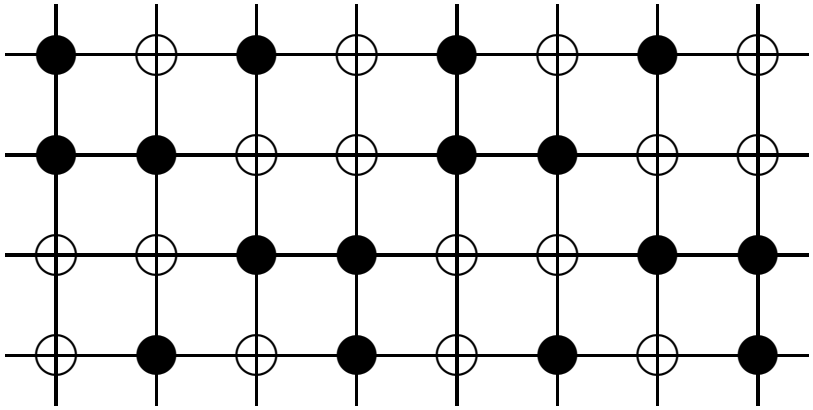
```
$ javac Zahlenraten.java UnbekannteZahl.java
$ java Zahlenraten
Errate eine Zahl zwischen 0 und 99:
50
Die gesuchte Zahl ist kleiner.
Versuche es nocheinmal (Ende mit ^D):
25
Die gesuchte Zahl ist größer.
Versuche es nocheinmal (Ende mit ^D):
36
Die gesuchte Zahl ist größer.
Versuche es nocheinmal (Ende mit ^D):
43
Gratulation! Zahl erraten!
```

Programmkonstruktion

Es geht um mehr als nur Programmieren:

- Zusammenhänge im gesamten Umfeld: Maschinen, Programme, Softwareentwickler, Anwender
- grundlegende Konzepte und Aussagen der Informatik
- Techniken, Werkzeuge und Vorgehensweisen
- handwerkliches Geschick und abstraktes Denkvermögen

Vom Webstuhl ins digitale Zeitalter



10101010	AA	170
11001100	CC	204
00110011	33	51
01010101	55	85

- analog = kontinuierlicher Wert = physikalische Größe, aber Fehler bei Verarbeitung und Speicherung unvermeidlich
- digital = diskreter Wert = Digitalisierungsfehler bei Messung, aber fehlerfrei verarbeit- und speicherbar

Binäre Worte – Darstellungsformen

binär / oktal / hexadezimal / dezimal / Zweierkomplement

0000 = 00 = 0 = 00 = 0	1000 = 10 = 8 = 08 = −8
0001 = 01 = 1 = 01 = 1	1001 = 11 = 9 = 09 = −7
0010 = 02 = 2 = 02 = 2	1010 = 12 = A = 10 = −6
0011 = 03 = 3 = 03 = 3	1011 = 13 = B = 11 = −5
0100 = 04 = 4 = 04 = 4	1100 = 14 = C = 12 = −4
0101 = 05 = 5 = 05 = 5	1101 = 15 = D = 13 = −3
0110 = 06 = 6 = 06 = 6	1110 = 16 = E = 14 = −2
0111 = 07 = 7 = 07 = 7	1111 = 17 = F = 15 = −1

- ein n-Bit-Wort kann 2^n Werte darstellen
- Dezimalzahlen: $0 \ldots 2^n - 1$ oder $-2^{n-1} \ldots 2^{n-1} - 1$

Bytes

- früher: 1 Byte = Wortlänge für Darstellung von Zeichen
- heute: 1 Byte = 8 Bit (auch als Teil eines Wortes)
- noch immer: Größe in Byte $\approx$ Textlänge in Zeichen

$$
\begin{aligned}
1\ \text{kB} &= 2^{10}\ \text{Byte} = &\quad 1.024\ \text{Byte} &\approx 10^{3}\ \text{Byte}\\
1\ \text{MB} &= 2^{20}\ \text{Byte} = &\quad 1.048.576\ \text{Byte} &\approx 10^{6}\ \text{Byte}\\
1\ \text{GB} &= 2^{30}\ \text{Byte} = &\quad 1.073.741.824\ \text{Byte} &\approx 10^{9}\ \text{Byte}\\
1\ \text{TB} &= 2^{40}\ \text{Byte} = &\quad 1.099.511.627.776\ \text{Byte} &\approx 10^{12}\ \text{Byte}
\end{aligned}
$$

Bit-Operationen

- Logische Bit-Operationen:

AND:	$0\,0 \to 0$	$0\,1 \to 0$	$1\,0 \to 0$	$1\,1 \to 1$
OR:	$0\,0 \to 0$	$0\,1 \to 1$	$1\,0 \to 1$	$1\,1 \to 1$
XOR:	$0\,0 \to 0$	$0\,1 \to 1$	$1\,0 \to 1$	$1\,1 \to 0$
NOT:	$0 \to 1$	$1 \to 0$		

- Bit-Multiplikation entspricht UND:

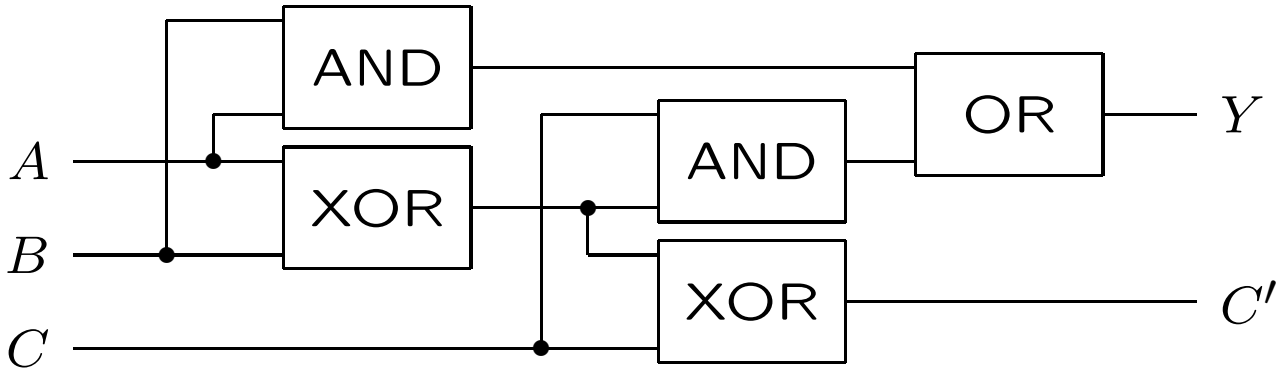
$$0 \cdot 0 = 0 \qquad 0 \cdot 1 = 0 \qquad 1 \cdot 0 = 0 \qquad 1 \cdot 1 = 1$$

- Bit-Addition durch XOR, Übertrag durch AND:

$$0 + 0 = 00 \qquad 0 + 1 = 01 \qquad 1 + 0 = 01 \qquad 1 + 1 = 10$$

Logische Schaltung

Beispiel: Volladdierer



- Auch komplexe Schaltungen über einfache Gatter realisiert
- Gatterlaufzeiten ca. 0,1 ns bis über 100 ns
- Quelle und Ziel ist Speicher, Berechnung im fixen Takt

VHDL-Programm für Hardware

```
library IEEE;                      -- aus der Bibliothek IEEE:
use IEEE.std_logic_1164.all;       -- Form der Bit-Darstellung

entity ANDGATE is                  -- ein ANDGATE hat
 port (IN1 : in std_logic;         -- zwei Eingaenge (IN1, IN2)
       IN2 : in std_logic;         -- und einen Ausgang (OUT1)
       OUT1: out std_logic);       -- als einfache Bits
end ANDGATE;

architecture RTL of ANDGATE is
begin
 OUT1 <= IN1 and IN2;              -- Berechnung des Ergebnisses
end RTL;
```

Quick and Dirty Java Guide (1)

- Klasse von Objekten: `public class UnbekannteZahl {...}`
- Variable als Datenbehälter: `private int zahl;`
- Was beim Erzeugen eines Objekts passiert:

```
public UnbekannteZahl (int grenze) {
    zahl = (new Random()).nextInt() % grenze;
    if (zahl < 0) {
        zahl = zahl + grenze;
    }
}
```

Quick and Dirty Java Guide (2)

- Zuweisung: `zahl = zahl + grenze;`
  `int versuch = sc.nextInt();`
- Eine Nachricht schicken: `zuErraten.gleich(versuch)`
- Wie Nachrichten beantwortet werden:

```
public boolean gleich (int vergleichszahl) {
    return (zahl == vergleichszahl);
}
```

- Bedingte Anweisung: `if(sc.hasNextInt()){...} else{...}`
- Schleife: `while (sc.hasNext()) {...}`

Quick and Dirty Java Guide (3)

- Literal: `"ein String"` (Zeichenkette), `99` (`int`-Zahl)
- Ausgeben einer Zeichenkette:

```
System.out.println("Gratulation! Zahl erraten!");
```

- Einlesen einer Zeichenkette und einer Zahl über `Scanner`:

```
Scanner sc = new Scanner(System.in);
if (sc.hasNext()) { String s = sc.next(); ... }
if (sc.hasNextInt()) { int i = sc.nextInt(); ... }
```

- Wie die Klasse ausführbar wird:

```
public static void main (String[] args) {...}
```