Programmiermethodik 1
Programmiertechnik

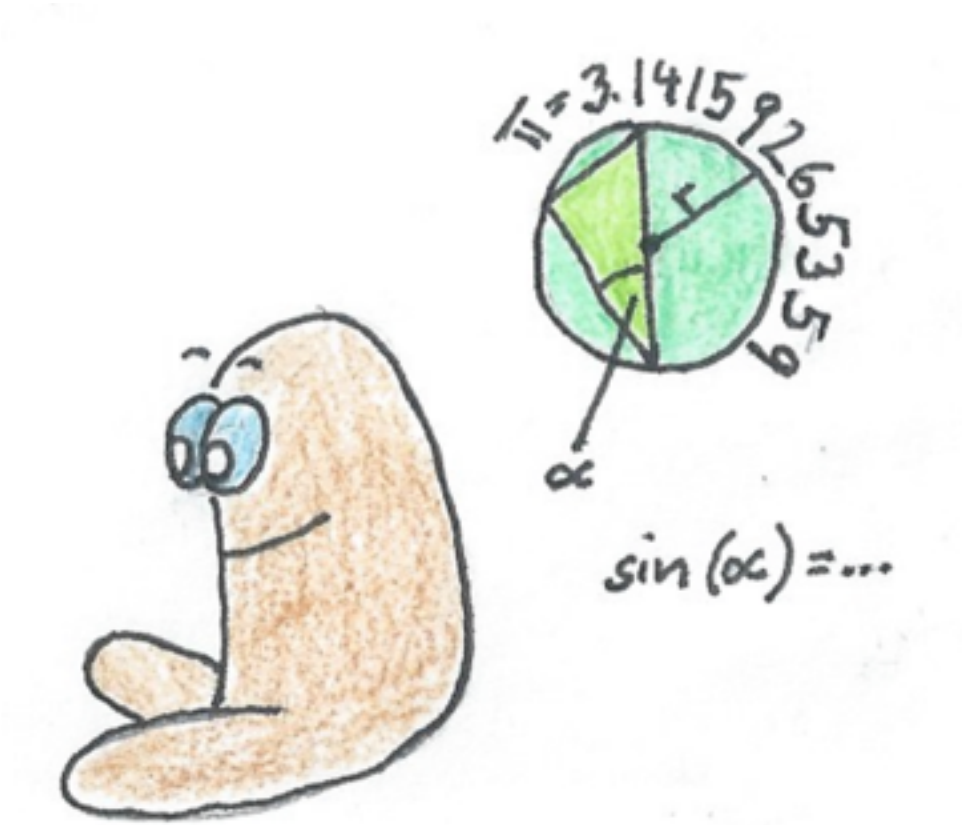
Bibliotheksfunktionen, Binärzahlen und Zeichen

Wiederholung

- Variable Parameteranzahl in Methoden
- Dokumentation
- Code-Formatierung

Ausblick

Worum gehts?



Agenda

- Mathematische Bibliotheksfunktionen
- Binärzahlen
- Zeichen

Mathematische Bibliotheksfunktionen

Mathematische Bibliotheksfunktionen

- Aufruf vorgefertigter Methoden der Klasse Math
- Beispiele:

Mathematische Funktion	Java-Methode
Absolutbetrag	Math.abs(x)
Quadratwurzel	Math.sqrt(x)
natürlicher Logarithmus	Math.log(x)
Logarithmus zur Basis 10	Math.log10(x)
e-Funktion	Math.exp(x)
Sinus	Math.sin(x)
Arcus-Tangens	Math.atan(x)

Mathematische Bibliotheksfunktionen

- Bibliotheksmethoden
  - erwarten bei Verwendung („Aufruf“) Argumente
    - jeweils Ergebnis eines Ausdrucks
  - liefern einen Funktionswert zurück (bei Math i.d.R. vom Typ double)
- Aufruf-Syntax für Methoden der Klasse Math mit einem Argument:
  - Math.<Methodenname>(<Ausdruck>)
  - Beispiel: Sinus von 0.5 (alle Winkel im Bogenmaß)
    - Math.sin(0.5) → 0.479425538604203
- vordefinierte Konstanten
  - Kreiszahl π: Math.PI
  - Eulerzahl e: Math.E

Beispiel

- Sinus von 45° = ¼ π:

```
Math.sin(Math.PI/4) → 0.7071067811865475
```

Variable Anzahl von Argumenten

- mehreren Argumente: Trennung durch Komma
  - Beispiel Potenzfunktion x^y: Math.pow(x,y)
    - Berechnen von 37: Math.pow(3,7) → 2187.0
    - Vorsicht: Math.pow rechnet immer mit double-Werten
      → schwer vorhersagbare Rundungsfehler möglich
  - Beispiel: Minimum zweier Werte
    - Math.min(2,4) → 2
  - Beispiel: Maximum zweier Werte
    - Math.max(2,4) → 4
- Methoden ohne Argument werden mit leeren Klammern aufgerufen
  - Die Random-Funktion liefert eine Zufallszahl aus dem Bereich [0,1) als double-Wert: Math.random()
  - Funktionswert ist nicht vorhersagbar!

Geschachtelte Aufrufe

- Argumente für Bibliotheksmethoden sind beliebige Ausdrücke
- Funktionswert einer Bibliotheksmethode kann in Ausdrücken wie der Wert einer Variablen oder eines Literals verwendet werden
- geschachtelte Aufrufe sind daher zulässig
- Auswertereihenfolge: von innen nach außen

- Beispiel

```
Math.sin(Math.pow(2.2, 3.4));
```

Anwendungstipps

- Math. kann weggelassen werden, wenn die Methoden der Klasse Math dem Compiler bekannt sind

```
import static java.lang.Math.*; // Sourcecode-Anfang
```

- Java-API-Dokumentation der Klasse Math
  - *http://download.oracle.com/javase/8/docs/api/java/lang/Math.html*

Übung: Max3Bib

- Erstellen Sie ein Programm Max3Bib, das von 3 übergebenen Integer-Werten den größten Wert ermittelt und ausgibt!
    - Verwenden Sie für Ihren Algorithmus eine Bibliotheksmethode.

- Anforderungsanalyse
    - Eingabe
        - der Benutzer gibt 3 ganzzahlige Werte ein
    - Ausgabe
        - der größte der drei Werte wird ausgegeben

Binärzahlen

Bit

- genau zwei Zustände
- Darstellung auf verschiedene Weise
    - an/aus
    - richtig/falsch
    - magnetisiert/nicht magnetisiert
    - Spannung 5 Volt/0 Volt (Darstellung im Prozessor)
    - 1/0 (mathematische, binäre Darstellung)
- alle Daten (Zahlen, Texte, Bilder, ..) und Prozessor-Befehle können als Bitfolge dargestellt werden

Beispiel

- Umrechnung der Bitfolge 10101 in eine Dezimalzahl (21)

1	**0**	**1**	**0**	**1**
1 x 24	0 x 23	1 x 22	0 x 21	1 x 20
16	0	4	0	1

Logische Verknüpfungen

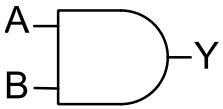
- Bits werden durch elektronische Schaltelemente verknüpft
- Schaltelemente realisieren die logischen Schaltfunktionen

Y=NOT (A)

A	Y
0	1
1	0

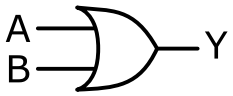
Y=AND (A,B)

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1



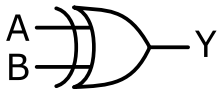
Y=OR (A,B)

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1



Y=XOR (A,B)

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0



Übung: Logische Operationen

- Welches Ergebnis liefern diese Verknüpfungen?
  - a) 0 AND 1
  - b) 0 AND 0
  - c) 1 AND 1
  - d) 0 OR 1
  - e) 0 XOR 1
  - f) 1 OR 1
  - g) 1 XOR 1
  - h) 0 NOT 1
  - i) NOT 0

Datentyp byte

- einfacher Typ byte repräsentiert
    - 8-Bit-Zahlen (mit Vorzeichen, Wertebereich -128 bis 127) oder
    - 8 Datenbits (ohne Vorzeichen, Wertebereich 0 bis 255)
- vordefiniert wie long (64 Bit), int (32 Bit), short (16 Bit)
- kompatibel zu allen anderen numerischen Datentypen
- dient oft der bitweisen Verarbeitung von Daten mit
    - Bit-Operatoren (logische Operatoren für Operanden vom Typ byte)
    - Shift-Operatoren (Verschiebe-Operatoren)
- Anwendungsbeispiel:
    - Codieren / Decodieren (Verschlüsseln / Entschlüsseln) von Dateien
- Vorsicht bzgl. Lesbarkeit!
    - Nur anwenden, wenn nötig!

Bit-Operatoren für byte-Operanden

- Wahrheitstabelle für Bit-Verknüpfungen

Operator	Name	deutsch	Funktion
&	AND	Bitweises Und	Bei a & b wird jedes Bit der beiden Bytes einzeln Und-verknüpft
|	OR	Bitweises inklusives Oder	Bei a | b wird jedes Bit einzeln Oder-verknüpft
^	XOR	Bitweises exklusives Oder	Bei a ^ b wird jedes Bit einzeln Xor-verknüpft (kein "a hoch b")
~	NOT	Komplement	Invertiert jedes Bit

Bit 1	Bit 2	~Bit 1	Bit 1 & Bit 2	Bit 1 | Bit 2	Bit 1 ^ Bit 2
0	0	1	0	0	0
0	1	1	0	1	1
1	0	0	0	1	1
1	1	0	1	1	0

Beispiele für Bit-Operationen

```
byte a = (byte)0b11110000;
byte b = (byte)0b00001111;
byte and = (byte) (a & b);
System.out.println("and: " + and);  // 0
byte or = (byte) (a | b);
System.out.println("or: " + or);    // -1
byte c = (byte)0b10000001;
System.out.println("mit Vorzeichen: " + c); /* implizite
                                Konvertierung zu int = -127 */
```

Shift-Operatoren

- sogenannte Verschiebe-Operatoren
- Shift-Operatoren verschieben die Bits eines Datenworts nach rechts oder links

Operator	Bezeichnung	Funktion
<<	Links-Shift	- alle Bits von b um n Positionen nach links geschoben - rechts frei werdenden Bitstellen: mit 0 aufgefüllt
>>	Rechts-Shift mit Vorzeichen	- alle Bits von b um n Positionen nach rechts geschoben - je nach Vorzeichenbit wird links mit 0 oder 1 aufgefüllt - Vorzeichenbit bleibt also unberührt
>>>	Rechts-Shift ohne Vorzeichen	- alle Bits von b um n Positionen nach rechts geschoben - links frei werdenden Bitstellen werden mit 0 aufgefüllt - Vorzeichenbit wird mitgeschoben

Shift-Anwendungen

- Division und Multiplikation durch Shift
  - Verschiebung um 1 Stelle nach links → Multiplikation mit 2
  - Beispiel: 00000010 << 1 → 00000100
    2 4
  - Allgemein:
    - Verschiebung um n Stellen nach links → Multiplikation mit 2n
    - Verschiebung um n Stellen nach rechts → Division durch 2n
- Beispiel: Bit setzen

```
int n = ...;
int pos = ...;
/* Setzen des n-ten Bits (von rechts) */
n = n | (1 << pos);
```

Übung: Multiplikation

- Schreiben Sie eine Methode

  `schnelleMultiplikation(int zahl, int faktor),`

  die die Zahl `zahl` mit dem Faktor `faktor` multipliziert. Verwenden Sie Bit-Shifting. Nehmen Sie an, dass sich `faktor` als 2^m darstellen läßt (m müssen Sie dann natürlich bestimmen).
- Beispiel: `m.schnelleMultiplikation(3, 4) → 12`

Zeichen (char)

Datentyp char

- einfacher Typ char (engl. character) repräsentiert einzelne Zeichen
- vordefiniert wie int, double, boolean
- char-Literale werden in einzelne Hochkommas gesetzt
- Beispiele:

Literal	Bedeutung
'a'	der kleine Buchstabe „a“
'5'	die Ziffer „5“ (nicht der int-Wert 5)
'%'	das Prozent-Zeichen
' '	das Leerzeichen

Operationen mit char-Variablen

- Deklaration und Zuweisung einer Variablen:

```
char letter;
letter = 'a';
```

- Vergleich von Zeichen auf Gleichheit und Ungleichheit:

```
char five = '5';
if ( five == 'V' ){          // false
      ...
}
```

- Größenvergleich für kleine oder große Buchstaben gemäß Alphabet:

```
char upperA = 'A';
if ( upperA < 'B' ){         // true
    …
} else if ( upperA > 'B'){ // false
              …
}
```

Zeichencodes

Zeichen	Code (Dez)
'a'	97
'5'	53
'%'	37
' '	32

- Code = eindeutiger int-Wert für jedes Zeichen
- implizite Typkonversion char→int liefert Code
- Beispiele:

```
int i = 'a';          // i == 97
i = 5*'5' + i;        // i == 5·53 + 97 = 362
int x = 'a' + 'b'    // x == 97 + 98 = 195, not 'c'!
```

- keine implizite Typkonversion int → char,
- explizite Typkonversion (Typecast) aber zulässig

```
char letter;
letter = (char)97;            // letter == 'a'
letter = (char)(letter + 1);        // letter == 'b'
```

Übung: Zeichen

- Schreiben Sie ein Programm, das das Zeichen an der Stelle index im Alphabet ausgibt. index wird vom Anwender eingegeben
  - Hinweis: der Buchstabe 'a' hat den Code 97
  - Beispiele:

```
index = 0 → Ausgabe 'a'
index = 10 → Ausgabe 'k'
```

Zeichensätze

- Zeichensatz = Zuordnungstabelle Zeichen ↔ Code
- verschiedene Zeichensätze, einige wenige haben sich durchgesetzt

Zeichensatz	Codes	Bemerkung
ASCII	0−127	Auf US-amerikanische Anwendungen ausgerichtet, keine deutschen Umlaute oder Sonderzeichen europäischer Sprachen
ISO Latin-1 (ISO-8859-1)	0−255	0-127: identisch mit ASCII 128−255: überwiegend Schriftzeichen aus westeuropäischen Sprachen
Unicode	0−100000+	0−127: identisch mit ASCII 0−255: identisch mit Latin-1 256+: Zeichen der meisten Weltsprachen

- Java benutzt Unicode (2 Byte pro Zeichen: UTF-16-Darstellung)

Unicode-Escapesequenzen

- Darstellung von Textzeichen als Escape-Sequenz:

```
\uXXXX  = vierstelliger, hexadezimaler Unicode eines einzelnen Zeichens
ggf. mit führenden Nullen
```

- Beispiel: Code von „€“ = 836410 = 20AC16

```
char euro = '\u20AC';
System.out.println(euro);  // gibt "€" aus
```

- Escape-Sequenzen werden vom Compiler zu Beginn der Übersetzung ausgewertet
  - ⇒ gelten im gesamten Quelltext, nicht nur in Literalen
- äquivalent zum obigen Beispiel:

```
ch\u0061r euro = '\u20AC';
System.out.println(\u20AC);
```

Ersatzdarstellungen für Steuerzeichen

- Steuerzeichen = Zeichen mit Codes 0–31 mit Sonderfunktion, oft zur Steuerung externer Geräte
- meistens nur noch historische Bedeutung
- in Java gibt es Ersatzdarstellungen für einige Steuerzeichen

Code (Dez)	Ersatz-darstellung	Bezeichnung	Bedeutung
0	\0	0-Byte	nur Nullen
8	\b	Backspace	voriges Zeichen löschen
9	\t	Tabulator	zur nächsten Spaltenposition
10	\n	Newline	neue Zeile
12	\f	Form Feed	neues Blatt (Drucker)
13	\r	Carriage Return	Schreibposition zurück an den Anfang derselben Zeile (nicht in allen OS!)

Ersatzdarstellungen für Begrenzer/Entwerter

- Begrenzer " und ' sowie der Entwerter \ müssen auch als "normales" Zeichen ohne spezielle Funktion darstellbar sein
  - Ersatzdarstellungen nötig!

Code (Dez)	Ersatz-darstellung	Bezeichnung	Bedeutung
34	\"	Anführungs-zeichen	Begrenzer für Text (Zeichenketten-Literale / Strings)
39	\'	Hochkomma	Begrenzer für einzelne Zeichen (char-Literale)
92	\\	Backslash	Entwertet das folgende Zeichen

- Beispiel:

```
System.out.println("\"a\\b\""); ? "a\b"
```

Zeilenwechsel

- verschiedene Betriebssysteme codieren Zeilenwechsel in Textdateien mit Zeichen unterschiedlicher Codes:
  - Unix, Linux, MacOS X: 10 (ein Zeichen: \n)
  - Windows: 13 10 (zwei Zeichen: \r\n)
  - MacOS bis Version 9: 13 (ein Zeichen: \r)
- innerhalb eines Javaprogramms erscheint ein Zeilenwechsel immer als "\n"
- Konversion von der/in die Betriebssystem-spezifische Codierung leistet die Java Virtual Machine

Bibliotheksklasse Character

- Beispiele (Methoden arbeiten im Unicode)
- `Character.isLetter('ß') → true`
- `Character.toLowerCase('Ä') → 'ä'`

boolean isLetter(char ch)	ist ch ein Buchstabe (groß oder klein)?
boolean isDigit(char ch)	ist ch eine Ziffer?
boolean isLowerCase(char ch)	ist ch ein Großbuchstabe?
boolean isUpperCase(char ch)	ist ch ein kleiner Buchstabe?
char toLowerCase(char ch)	kleiner Buchstabe zu ch, wenn er existiert; ch ansonsten
char toUpperCase(char ch)	großer Buchstabe zu ch, wenn er existiert; ch ansonsten

Primitive Datentypen

Typname	Länge (Byte)	Wertebereich	Defaultwert
`boolean`	1	`true`, `false`	`false`
`char`	2	Alle Unicode-Zeichen	\u0000
`byte`	1	$-2^7 \ldots 2^7-1$	0
`short`	2	$-2^{15} \ldots 2^{15}-1$	0
`int`	4	$-2^{31} \ldots 2^{31}-1$	0
`long`	8	$-2^{63} \ldots 2^{63}-1$	0
`float`	4	$+/-3.40282347 * 10^{38}$	0.0
`double`	8	$+/-1.7976931348623157 * 10^{308}$	0.0

Zusammenfassung

- Mathematische Bibliotheksfunktionen
- Binärzahlen
- Zeichen