Informatik I WS 07/08

Tutorium 24

22.11.07

Bastian Molkenthin

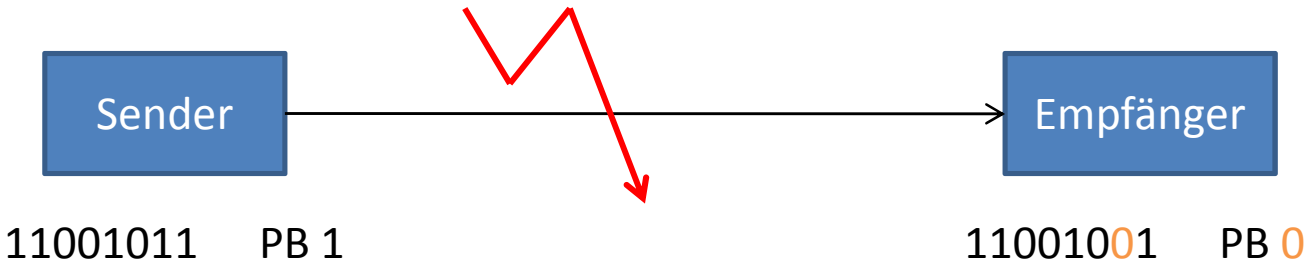
E-Mail: infotut@sunshine2k.de

Web: http://infotut.sunshine2k.de

Universität Karlsruhe (TH)
Forschungsuniversität · gegründet 1825

Review Blatt3

Wenn ein Fehler aufgetreten ist, ist das Paritätsbit 1. **-> FALSCH**



Sender berechnet Paritätsbit und sendet dieses mit den Daten mit.

Empfänger berechnet beim Erhalt von Daten das Paritätsbit und vergleicht dieses mit dem mitgesendeten Paritätsbit.

- Unterscheiden sich diese ist sicherlich ein Fehler aufgetreten.
- Unterscheiden Sie sich nicht, ist wahrscheinlich kein Fehler ausgetreten.

Review Blatt 3

Bezeichner = Kennzeichner { Punkt Kennzeichner }.
Buchstabe = Buchstabe { Buchstabe | Strich | Ziffer }.
Buchstabe = “a“ | “b“ | …

{ … } heißt null- bis n-maliges Vorkommen ⟹ Das Wort a lässt sich ableiten!

Allgemeines :

§ Passt eure Antworten den Fragestellungen an!

- „Nenne…“ , „Zähle auf…“, … ⟹ Strichpunktliste reicht
- „Erläutere…“, „Wie funktioniert ….“, …. ⟹ Ausformulierte Erklärung

§ Achtet auf die Punkte.

- Für 1 Theoriepunkt ist eine halbe Seite Erklärung sicherlich zu viel…
- Für Erklärungsaufgabe mit 3 Punkten reichen 2 kurze Sätze meistens nicht aus…

Programmierkonventionen

Folgende Konventionen gelten für Info1:

- Alle Namen in englischer Sprache!
- Worttrennung durch Großbuchstaben , *z.B. getInfo()*
- Klassennamen beginnen mit Großbuchstaben
- Variablen und Methoden fangen klein an
- Methoden fangen mit einem Verb an.
- Sinnvolle Namen verwenden!
  - außer bei Hilfsvariablen oder
  - Schleifenvariablen (hier *i,j,k* verwenden)

Kommentare

Kommentare sind nützlich, um ein Programm für einen fremden Leser verständlich zu machen.

- Kommentiert können sinnvollerweise z.B.
  - *Methoden*, um über ihre Aufgabe und Funktionsweise zu informieren.
  - *Variablen*, um ihren Sinn und Zweck zu erläutern

- Auch Kommentare müssen auf Englisch sein!

- Unnütze Kommentare vermeiden, wie z.B.

```
x = x +1;          // increment x by one
```

- Einzeiliger Kommentar:

```
……;        // it does something clever
```

- Mehrzeiliger Kommentar:

```
……;        /* I am a comment which
               lasts for 2 lines*/
```

Datentypen

In jeder Programmiersprache gibt es Platzhalter, welche verschiedene Werte speichern können (Variablen, Konstanten)

Den Variablen muss meistens (so auch in Java) schon vor der Wertzuweisung ein Datentyp zugewiesen werden (*Deklaration*)

Beispiele hierfür sind

- natürliche Zahl: **int**, **long**, …
- Buchstabe: **char**
- Zeichenkette: **String**
- …

Nicht auf jeden Datentyp können Operationen gleichermaßen angewendet werden

Was ergibt 5 * „Hallo“? ⇒ In der Regel einen Compilerfehler

Datentypen (2)

Folgende Zahlen-Datentypen sollen näher betrachtet werden:

Ganzzahlige Datentypen: *byte, short, int, long*

Fließkommazahlen: *float, double*

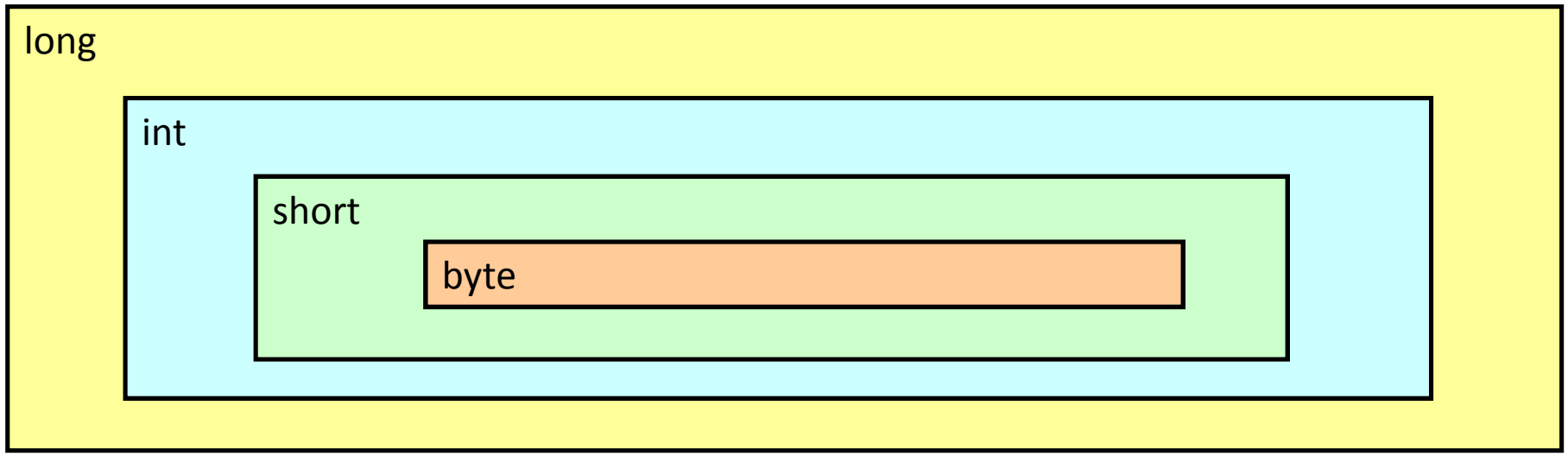
Die Datentypen der Operanden bestimmen in Java den Datentyp vom Ergebnis!

Arithmetische Operationen in Java:

Addition	a + b	Modulo	a % b
Subtraktion	a - b	Erhöhen um 1	a++
Division	a / b	Verringern um 1	a--
Multiplikation	a * b	Einige weitere	-, +=, -=, /=, *=, …

Datentypen (3)

- Wertebereich der Datentypen für ganze Zahlen:
  - byte 8-Bit-Zahl $-2^7 \dots 2^7-1$ -128 … 127
  - short 16-Bit-Zahl $-2^{15} \dots 2^{15}-1$ -32768 … 32767
  - int 32-Bit-Zahl $-2^{31} \dots 2^{31}-1$ -2.147.483.648 … 2.147.483.647
  - long 64-Bit-Zahl $-2^{63} \dots 2^{63}-1$ wirklich sehr groß!



Datentypen (4)

Das Ergebnis arithmetischer Operationen mit ganzen Zahlen ist **immer** *int*, außer einer der Operanden ist vom Typ *long* oder es wird *explizit gecastet* (das gilt auch für die Division!):

int + int	⇒	int
byte - byte	⇒	int
long - int	⇒	long
int / byte	⇒	int
byte * short	⇒	int

Das bedeutet für konkrete Zahlen z.B. : 10 / 3 = 3

Der Nachkommateil wird **abgeschnitten**, es wird nicht gerundet!

Datentypen (5)

Typkonversion (= Casting)

Ist nötig um einem „kleineren“ Datentypen einen „größeren“ zuzuweisen.

Beispiel:

```
long x = 256;
int i = 42;

x = i;          // ohne Typenkonversion möglich
i = x;          // liefert einen Fehler
i = (int) x;    // Typenkonversion behebt Fehler
```

Achtung: Wenn man explizit castet, kann es zu unerwarteten Auswirkungen kommen: Wenn die Zahl nicht in den Typ passt, so wird ”modulo“ gerechnet.

```
(byte)(64*2)          // -128
(byte)(-129)          // 127
(1000000000*10)       // 1410065408
```

Datentypen (Beispiel)

Welche Typen erhalten wir nach der Auswertung? Welchen Wert haben sie? Korrigiere gegebenenfalls mittels Typkonversion.

Gegeben sind:

```
long l = 2;
int i = 5;
byte c = 3;
```

Zuweisungen (unabhängig voneinander) :

i = l/c;	(0; Typ long, Casting nötig: i = (int)(l/c);!)
i = i*c;	(15; Typ int)
c = c+1;	(4; Typ int, Casting nötig: c = (byte)(c+1);!)
c = c++;	(3; Typ byte)

Datentypen (Beispiel 2)

Welchen Wert ergeben folgende Ausdrücke und von welchem Typ sind sie?

Gegeben sind:

```
long a = 3;
int b = 4;
short c = 5;
byte d = 6;
```

Ausdrücke: (unabhängig voneinander)

d / b * a	(3; Typ long)
c + b * (d+1)	(33; Typ int)
d/(c-1) * b/2	(2; Typ int)
d % b	(2; Typ int)
-d % b	(-2; Typ int)
-d / a	(-2; Typ long)
c++ % d	(5; Typ int)

Datentypen – Kleines Verwirrspiel

Welche Werte nehmen die Variablen a,b,c nach den jeweiligen Befehlen an? (Anweisungen sind diesmal abhängig voneinander).

```
int a = 0;

int b = 1;

int c = 2;
```

a=b++*c;	**a = 1 * 2 = 2; b = 1 + 1 = 2**
c%=b;	**c = 2 % 2 = 0**
b+=++(a)+(c--);	**b = b + (a + 1) + c = 2 + (2 + 1) + 0 = 5; c = 0 – 1 = -1** **a = a + 1 = 3**

Grundgerüst einer Java Anwendung:

```
public class Classname {
        public static void main(String[] args) {
                // Anweisungen etc.
        } // end main
} // end class
```

Classname durch sinnvollen Namen ersetzen!

- Quellcodedatei muss *Classname*.java heißen!
- Methode main ist eine spezielle Methode. Sie ist der Einsprungspunkt in ein Programm, also „dort fängt das Programm an“.
- Sie wird von anderen Programmteilen benutzt und muss daher *public* und *static* sein.

Befehle:

- Eine Handlungsanweisung wird Befehl genannt.
- Mit ihm kann man Werte zuweisen, Variablen zuweisen, Methoden definieren und vieles mehr.
- Um einen Befehl von einem anderen zu trennen, wird dieser mit einem ´;´ beendet

```
int test = 0;

byte zahl = 5;

test = test + zahl;

test++;
```

If-then-else

- *If-then* und *if-then-else* Kontrollstrukturen erlauben, Programmteile nur unter bestimmtem Bedingungen auszuführen
- Können verschachtelt sein.

```
if (Bedingung)
   Anweisung X
```

```
if (Bedingung)
   Anweisung X
else
   Anweisung Y
```

Boolscher Ausdruck – ergibt true oder false

Beispiel:

```
If (zahl > 0)
   Out.println("Größer Null");
else
   if (zahl == 0)
      Out.println("Null");
   else
      Out.println("Negativ");
```

Switch

Kann anstatt einer Folge von If-Anweisungen verwendet werden.

```
If (year == 1983) ….
else if (year == 1989) ….
else if (year == 2000) ….
else ….
```

```
switch (year) {
    case 1983 : …. break;
    case 1989 : …. break;
    case 2000 : …. break;
    default : …. break;
}
```

- Der letze else-Zweig entspricht dem default-Zweig.
- Die case-Ausdrücke müssen Konstanten sein.
- Die break-Anweisung nach jedem case-Zweig nicht vergessen!
- Die switch-Anweisung ist schneller, verbraucht aber mehr Speicherplatz. Warum?

Befehlsblöcke:

- Schleifen / If-Anweisungen führen jeweils nur einen Befehl aus
- Deshalb kann man analog zur Klammerung in Rechnungen, Befehle zu Befehlsblöcken zusammenfügen
- Um beides zu unterscheiden verwendet man nur { und }.

```
if (zahl > 10)
    zahl = 8;
    zahl *= test;
…
```

Wenn zahl <= 10 wird zahl = 8 nicht ausgeführt aber zahl *= test.

```
if (zahl > 10) {
    zahl = 8;
    zahl *= test;
}
```

Wenn zahl <= 10 wird keiner der beiden Befehle ausgeführt.

Variablen

- Variablen sind Speicherzellen mit veränderlichem Inhalt.
- Die Wahl des Variablentyps schränkt die Art der Daten ein, die man ihn ihr speichern kann.
- So kann ein String eine Zeichenkette fassen, während ein int nur eine Ganzzahl von -2147483648 bis 2147483647 speichert.
- Um eine Variable benutzen zu können, muss sie bekannt gemacht (*deklariert*) und ihr ein Anfangswert zugewiesen (*initialisiert*) werden.

```
long zahl;                          // declaration
zahl = 1;                           // initialization
String text = “Ich bin ein String!“; // declaration + initialization
```

Variablen (2) - Fließkommazahlen

- Fließkommazahlen beinhalten auch ganze Zahlen.
- Fließkommazahlen sind **immer** double (auch das Ergebnis von Operationen)

Konkret heißt das für eine Variable f vom Typ float:

```
float f = 2.78;                    // liefert Fehler
```

Beheben durch Typkonversion:

```
float f = 2.78f;
float f = (float)2.78;
```

Float liegt (ungefähr) im Bereich $+/-10^{+/-38}$

Double liegt (ungefähr) im Bereich $+/-10^{+/-308}$

Ausgabe

Die In-/Out Klassen liefern euch Methoden zur Ein- und Ausgabe von Werten.

Ausgabe:

Folgende Methoden stellt euch die Out-Klasse bereit:

```
Out.print(x);
Out.println(x);
```

Dabei kann x ein *String*, *char*, *byte*, *short*, *int*, *long*, *float*, *double* oder *boolean* sein.

Ein String kann durch Verkettung von Zeichenketten und Variablen erstellt werden

```
int i = 1;
String s ="Der Wert von i ist:" + i;
Out.print(s);
```

Alternativ: `Out.print("Der Wert von i ist:" + i);`

Unterschied zwischen *print()* und *println()* ?

Eingabe

Ziel : Einlesen einer Benutzereingabe

Folgende Methoden stellt euch die In-Klasse bereit:

```
In.readInt();        // liest ein int
In.readLine();       // liest ein String
In.readDouble();     // liest eine Kommazahl vom Typ double
```

Beispiel:

```
int i;
String s;
i = In.readInt();
s = In.readLine();
```

Aufgabe

Schreibe ein Programm, das zwei Integerzahlen a und b einliest und gebe ihre Summe aus!

```
public class Sum {
        public static void main(String args[])
        {
                Out.print(“Enter first summand : ");
                int a = In.readInt();
                Out.print(“Enter second summand: ");
                int b = In.readInt();
                Out.println(“Sum is " + (a+b));
        }
}
```

Schleifen

Eine Schleife erfüllt den Zweck bestimmte Operationen häufig hintereinander ausführen, bis eine Abbruchbedingung erreicht ist

In Java gibt es die drei wichtigen Schleifentypen:

- *For-Schleife*
- *While-Schleife*
- *Do-While-Schleife*

Meistens lassen sich die verschiedenen Schleifen ineinander überführen, aber in der Regel drängt sich ein bestimmter Typ auf

For - Schleife

- Für den Fall, dass etwas mit einer bestimmten Anzahl von Wiederholungen ausgeführt werden soll.
- Im Schleifenkopf wird ein Zähler integriert

Syntax:

```
for (Initialisierung; Bedingung; Inkrementierung) { /*Code*/ }
```

Beispiel:

```
int sum = 0;
for (int i = 0; i < 10; i++) {
        sum += i;
}
```

Berechnet 0+1+2+3+4+5+6+7+8+9 = 45

While - Schleife

- Etwas so oft ausführen, bis eine/mehrere Bedingung(en) erfüllt sind
- Kopfgesteuert

Syntax:

```
while (Bedingung){ /*Code*/ }
```

Beispiel:

```
while (tired == false || z < 10) {
        // Diverse Anweisungen
        z--;
}
```

Do-While - Schleife

- Etwas so oft ausführen, bis eine/mehrere Bedingung(en) erfüllt sind
- Wird mindestens einmal durchlaufen
- Fußgesteuert

Syntax:

```
do {
/*Code*/
} while (Bedingung);
```

Beispiel:

```
do {
        // Diverse Anweisungen
        z--;
} while (tired == false || z < 10);
```

Methoden

- Methoden sind benannte Anweisungsfolgen
- Schaffen wiederverwendbaren Code
- Können Parameter erhalten und/oder einen Wert zurückgeben

- Man unterscheidet zwischen Funktionen und Prozeduren

  Wo ist der Unterschied?

  ⇒ Funktionen liefern immer einen *Rückgabewert*, Prozeduren nicht!

- Welches Schlüsselwort wird in Java für Prozeduren verwendet?

  ⇒ **void**

Methoden (2)

(Unvollständige) Deklaration von Methoden (Syntax):

[public | private] static datentyp name (param1, param2, …)

Beispiele:

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

public static int addOne (int x) {
        return x + 1;
}
```

Aufgabe

Schreibe ein Programm welches einen Radius einliest (double) und den Flächeninhalt des Kreises berechnet. Die Berechnung soll in einer Funktion stattfinden.

```
public class CircleArea {

        public static double calcCircleArea(double radius) {
                return (radius * radius * 3.14159);
        }

        public static void main(String[] args)  {
                double radius, result;

                Out.print("Enter radius: ");
                radius = In.readDouble();
                result = calcCircleArea(radius);
                Out.println("Area of circle is : " + result);
        }
}
```

Aufgabe

Schreibe ein Programm welches solange eine (Int-)Zahl einliest bis eine Null eingegeben wird. Für jede Zahl soll ausgegeben werden, ob sie gerade/ungerade ist sowie eine Zufallszahl zwischen 0 und der eingegebenen Zahl.

Hinweis:

public static double **random**()

Returns a double value with a positive sign, greater than or equal to 0.0 and less than 1.0.
Returned values are chosen pseudorandomly with (approximately) uniform distribution from that range.

Von: http://java.sun.com/javase/6/docs/api/java/lang/Math.html#random()

Aufgabe - Lösung

```
public class ReadNumbers {

        public static void main(String[] args) {

                Out.print("Enter number : ");
                int number = In.readInt();
                while (number != 0)
                {
                        // if number is divisable by 2, it's even
                        if (number % 2 == 0)
                                Out.println("-> Even.");
                        else
                                Out.println("-> Uneven.");

                        Out.println("Random number between 0 and " + number +
                                        " : " + (Math.random() * number));
                        Out.print("Enter number : ");
                        number = In.readInt();
                }
        }
}
```

Fragen ???

Viel Spaß mit dem Übungsblatt!