Informatik II Übung, Woche 10

Giuseppe Accaputo

10. März, 2016

Plan für heute

1. Typumwandlung (*Typecasts*)
2. Ordnerstruktur für Übungen
3. Vorbesprechung Übung 3
4. Nachbesprechung Übung 2 (inkl. Live Programmierung)

Java Typen Übersicht

Typ	Grösse (Bits)	Min. Wert	Max. Wert
`byte`	8	-128	127
`short`	16	-32768	32767
`int`	32	-2^{31}	$2^{31}-1$
`long`	64	-2^{63}	$2^{63}-1$
`float`	32	$-3.4 \cdot 10^{38}$	$3.4 \cdot 10^{38}$
`double`	64	$-1.80 \cdot 10^{308}$	$1.80 \cdot 10^{308}$
`boolean`	n.d.	Entweder `true` oder `false`	
`char`	16	0x0000	0xffff

Implizite Typumwandlung

Eine implizite Typumwandlung findet dann statt, wenn der Zieltyp *grösser* als der Ursprungstyp ist:

```
ursprungstyp a = ...;
zieltyp b = a;
```

- Reihenfolge (aufsteigend in Grösse):

  `byte` → `short` → `int` → `long` → `float` → `double`

- Beispiel:

```
int i = 10213123;
long l = i; // OK, da long > int
byte d = i; // FEHLER, da byte < int
```

Explizite Typumwandlung

Eine explizite Typumwandlung wird dann benötigt, wenn der Zieltyp *kleiner* als der Ursprungstyp ist:

```
ursprungstyp a = ...;
zieltyp b = (zieltyp)a;
```

- Reihenfolge (absteigend in Grösse):

  `double` → `float` → `long` → `int` → `short` → `byte`

- Beispiel:

```
int i = 10213123;
byte d = (byte)i; // OK, da explizit
```

Beispiel: Typumwandlung

Frage: welcher Wert hat die Variable `d` nach der Ausführung des folgenden Codes?

```
double b = 1/2;
```

Beispiel: Typumwandlung

Frage: welcher Wert hat die Variable `d` nach der Ausführung des folgenden Codes?

```
double b = 1/2;
```

Antwort: 0. Die Zahlen $1, 2$ sind vom Typ `int`, d.h. die Nachkommastellen gehen verloren bei der Division.

Lösung: Explizite Typumwandlung nach `double`:

```
double b = (double)1/2;
```

Oder ein bisschen einfacher:

```
double b = 1.0/2;
```

Ordnerstruktur für Übungen

- Pro Übung ein Projekt und pro Aufgabe ein Packet

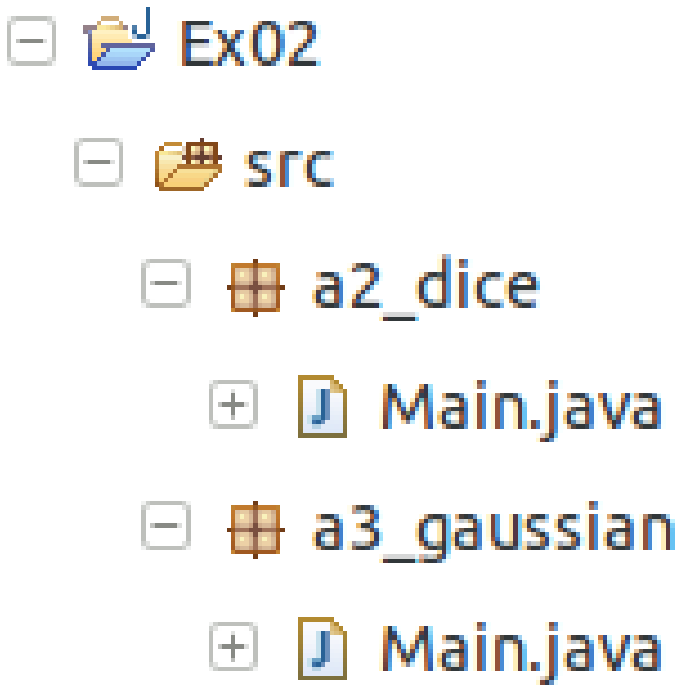

Figure : Projektstruktur für die Übung 2

Übung 3, Aufgabe 1: Random Surfer

- Implementation des Ameisen-Problems aus der Vorlesung (siehe Vorlesungsslides)

1. Matrix einlesen
2. Vektor-Matrix Multiplikation implementieren
3. Simulation implementieren

Übung 3, Aufgabe 1.1: Matrix einlesen

- Eingabeformat:

```
2 3       // Dimension der Matrix
1 2 3     // Matrix
4 5 6
```

- Der `Scanner` splittet den Konsoleninput standardmässig nach Leerzeichen (inkl. neue Zeilen) in Tokens auf.
- Beispiel: Matrix-Eingabe von oben in die Konsole kann mittels `Scanner` standardmässig in 8 Tokens aufgesplittet werden.
  - Pro Token ein `scanner.next...()`
- `Scanner` Dokumentation: https://docs.oracle.com/javase/8/docs/api/java/util/Scanner.html
- Eclipse Demo.: Einlesen der vorgegebenen Beispiele im Code

Übung 3, Aufgabe 1.3: Simulation

- Siehe Slide 35 der 3. Vorlesung:

```
int[] count = new int[3];
double[][] P=   { {0.1, 0.2, 0.7},
                  {0.2, 0.2, 0.6},
                  {0.5, 0.5, 0} };
int loc = 0; // 0=A, 1=B, 2=C

for (int i = 0; i<steps; ++i){
    ++count[loc];
    loc = UnfairDice(P[loc]);
}
```

- `simulateLine` wird verwendet, um den nächsten Zielort des Random Surfers auszuwählen.

Übung 3, Aufgabe 2: Hashfunktionen I

- Eine Hashfunktion ist eine Abbildung, die eine grosse Eingabemenge (z.B. einen String) auf eine kleinere Zielmenge (die Hashwerte) abbildet.
- Da die Eingabemenge meistens viel grösser ist als die Zielmenge kann es zu Kollisionen kommen.

Übung 3, Aufgabe 2: Hashfunktionen II

- Die Hashfunktion aus der Aufgabe sieht wie folgt aus:

$$\texttt{sum} = \sum_{\texttt{i=0}}^{\texttt{s.length - 1}} \left(\texttt{s[i]} \cdot b^{(i+1)} \right) \quad , \tag{1}$$

  wobei `s` unser Eingabestring, `s[i]` das $i$te Zeichen des Strings und b eine Primzahl ist.
- Ein `String` ist ein Array bestehend aus Zeichen, i.e., `char` Einträgen.
- Das $i$te Zeichen eines `Strings` kann mittels der `charAt()`-Methode ausgewählt werden.
- *Verwandle* das $i$te Zeichen in einen `int` um die Multiplikation in der Gleichung 1 auszuführen.

Übung 2, Aufgabe 2: Dice I

- Linear congruential generator (LCG): generiert Pseudozufallszahlen $L \in \{0, \ldots, m-1\}$ mit $\mathbb{P}[L = x] = 1/m$ für alle $x \in \{0, \ldots, m-1\}$ mittels

$$X_{n+1} = (aX_n + c) \mod m \tag{2}$$

- Im Code ist der LCG gegeben als

```
static final long m = 2147483647;
static final long b = 12345;
static final long a = 1103515245;
...
u = (u * a + b) % m;
```

Übung 2, Aufgabe 2: Dice II

- Verwende nun LCG um eine gleichverteilte Zufallszahl $U \in [0, 1)$ mit $\mathbb{P}[U \in [l, r)] = r - l$ zu generieren:

$$L \in \{0, 1, \ldots, m-1\} \quad \longrightarrow \quad U \in [0, 1) \quad . \tag{3}$$

- Gleichung 3 entspricht der Transformation

$$L \to L/m \quad . \tag{4}$$

- `Uniform()`-Methode implementiert die Transformation:

```
public static double Uniform(){
u = (u * a + b) % m;
return(double)u/m;
}
```

Übung 2, Aufgabe 2: Dice III

Lösung der *fair Dice* Aufgabe:

```
static int Dice(){
    double u = Uniform();
    if (u < 1.0/6) return 1;
    else if (u < 1.0/3) return 2;
    else if (u < 1.0/2) return 3;
    else if (u < 2.0/3) return 4;
    else if (u < 5.0/6) return 5;
    else return 6;
}
```

Übung 2, Aufgabe 2: Dice III (elegante Lösung)

- Beobachtung: Intervalle sind gleich gross

$W = 1$	$W = 2$	$W = 3$	$W = 4$	$W = 5$	$W = 6$
$\mathbb{P} = 1/6$	$\mathbb{P} = 1/6$	$\mathbb{P} = 1/6$	$\mathbb{P} = 1/6$	$\mathbb{P} = 1/6$	$\mathbb{P} = 1/6$
$[0, \frac{1}{6})$	$[\frac{1}{6}, \frac{2}{6})$	$[\frac{2}{6}, \frac{3}{6})$	$[\frac{3}{6}, \frac{4}{6})$	$[\frac{4}{6}, \frac{5}{6})$	$[\frac{5}{6}, 1)$

- Bilde das Intervall von $[0, 1)$ nach $[1, 7)$ ab und Runde ab:

```
static int Dice(){
    double u = Uniform();
    return (int)(u*6 + 1);
}
```

Übung 2, Aufgabe 3: Gaussian

Live-Programmierung