Informatik II

Übung 2

Giuseppe Accaputo, Felix Friedrich, Patrick Gruntz, Tobias Klenze, Max Rosmannek, David Sidler, Thilo Weghorn

FS 2017

Heutiges Programm

1. Arrays, Strings, Referenzen und Methoden
2. Sampling
3. Programmierübung

Arrayzuweisungen

```
int[] z = new int[5];

for (int i=0; i<z.length; ++i)
    z[i] = i*i;

int[] x = z;

int j = x[2];

x[1] = 99;
```


Bei der Zuweisung von Arrays *wird die Referenz kopiert*, nicht die Daten!

Arrayzuweisungen

```
int[] z = new int[5];

for (int i=0; i<z.length; ++i)
    z[i] = i*i;

int[] x = z;

int j = x[2];

x[1] = 99;
```

Bei der Zuweisung von Arrays *wird die Referenz kopiert*, nicht die Daten!

Arrayzuweisungen

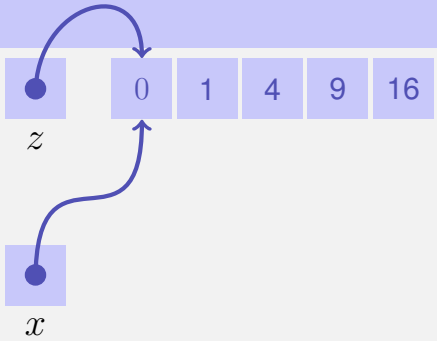
```
int[] z = new int[5];

for (int i=0; i<z.length; ++i)
    z[i] = i*i;

int[] x = z;

int j = x[2];

x[1] = 99;
```



Bei der Zuweisung von Arrays *wird die Referenz kopiert*, nicht die Daten!

Arrayzuweisungen

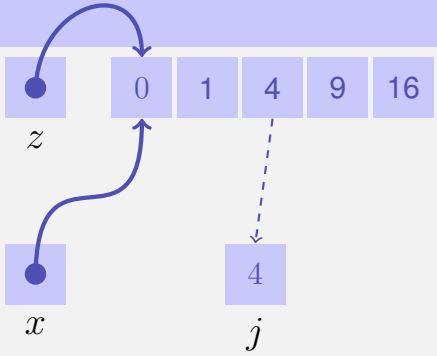
```
int[] z = new int[5];

for (int i=0; i<z.length; ++i)
    z[i] = i*i;

int[] x = z;

int j = x[2];

x[1] = 99;
```



Bei der Zuweisung von Arrays *wird die Referenz kopiert*, nicht die Daten!

Arrayzuweisungen

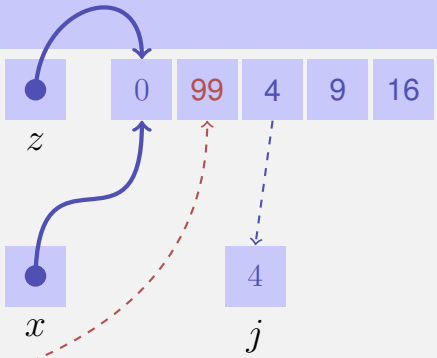
```
int[] z = new int[5];

for (int i=0; i<z.length; ++i)
    z[i] = i*i;

int[] x = z;

int j = x[2];

x[1] = 99;
```



Bei der Zuweisung von Arrays *wird die Referenz kopiert*, nicht die Daten!

Array Vergleiche

Erneut aufgepasst: Arrays sind Referenzen!!

```
double[] x = {1,2,3};
double[] y = x;
double[] z = {1,2,3};

if (y == x) {...} // y==x ist true
if (z == x) {...} // z==x ist false!
```

Für Kenner:

```
if (z.equals(x)) {...}       // z.equals(x) ist auch false !!
if (Arrays.equals(x,z)) {...} // Arrays.equals(x,z) ist true.
```

Vorsicht bei `Arrays.equals` bei mehrdimensionalen Arrays! (Was wird wohl geprüft?)

Strings

String: ein Objekt, welches Zeichenketten speichert.

```
String name = "Informatik";
String hochschule = "ETH";
String vorlesung = name + " an der " + hochschule;
int x = 3;
int y = 5;
String koordinaten = "(" + x + "," + y + ")"; // "(3,5)"
```

Stringvergleiche

Der Vergleich mit '==' vergleicht Referenzen, nicht den Inhalt!

```
Scanner scanner = new Scanner(System.in);
String n1 = scanner.next();
String n2 = scanner.next();
if (n1 == n2)
  System.out.println(n1 + "==" +n2);
else
  System.out.println(n1 + "!=" + n2);
}
```

Stringvergleiche

Der Vergleich mit '==' vergleicht Referenzen, nicht den Inhalt!

```
Scanner scanner = new Scanner(System.in);
String n1 = scanner.next();
String n2 = scanner.next();
if (n1 == n2)
  System.out.println(n1 + "==" +n2);
else
  System.out.println(n1 + "!=" + n2);
}
```

Eingabe: Info Info

Ausgabe: Info != Info

Stringvergleiche

Der Vergleich mit 'equals' vergleicht den Inhalt![1]

```
Scanner scanner = new Scanner(System.in);
String n1 = scanner.next();
String n2 = scanner.next();
if (n1.equals(n2))
  System.out.println(n1 + "==" +n2);
else
  System.out.println(n1 + "!=" + n2);
}
```

[1] Bei Arrays geht das nicht!

Stringvergleiche

Der Vergleich mit 'equals' vergleicht den Inhalt![1]

```
Scanner scanner = new Scanner(System.in);
String n1 = scanner.next();
String n2 = scanner.next();
if (n1.equals(n2))
  System.out.println(n1 + "==" +n2);
else
  System.out.println(n1 + "!=" + n2);
}
```

Eingabe: Info Info

Ausgabe: Info == Info

[1] Bei Arrays geht das nicht!

Methodenaufruf - Pass by Value

- Ein Methodenaufruf ist ein Ausdruck, dessen Wert falls vorhanden der Rückgabewert der Methode ist.
- In Java gilt immer die *Pass by Value* Semantik.

Pass by Value bedeutet: Argumentwerte werden beim Methodenaufruf in die Parameter *kopiert*.

Dies entspricht demselben Prinzip wie die Wertzuweisung an eine Variable.

Methodenaufruf - Pass by Value

```
// Methodenaufruf
int kekse = readInt( "Kekse: " , kinder );
```

Methodenaufruf - Pass by Value

```
  // Methodenaufruf
  int kekse = readInt( "Kekse: " , kinder );
```

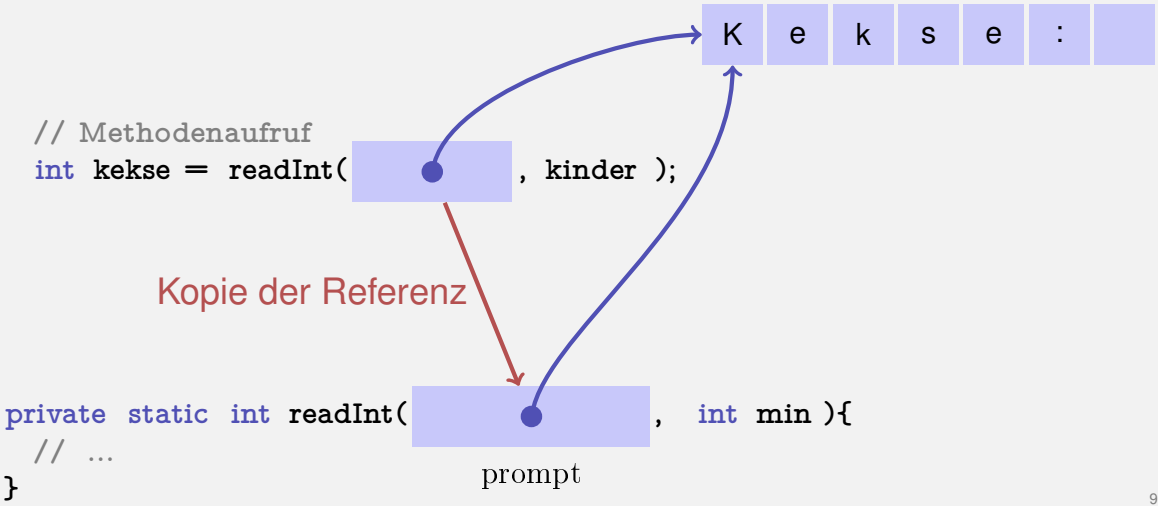
```
private static int readInt( String prompt ,  int min ){
  // ...
}
```

Methodenaufruf - Pass by Value


```
// Methodenaufruf
int kekse = readInt(          , kinder );
```

```
private static int readInt( String prompt ,  int min ){
  // ...
}
```

Methodenaufruf - Pass by Value



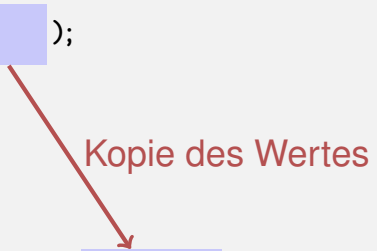
Methodenaufruf - Pass by Value

```
  // Methodenaufruf
  int kekse = readInt( "Kekse: " , 5 );
```

```
private static int readInt( String prompt ,  int min ){
  // ...
}
```

Methodenaufruf - Pass by Value

```
// Methodenaufruf
int kekse = readInt( "Kekse: " , 5 );
```

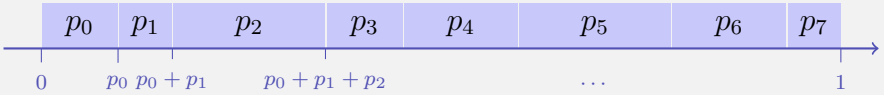



```
private static int readInt( String prompt , 5 ){
  // ...
}
```

min

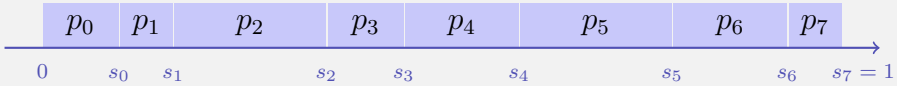
Unfair würfeln

Gegeben: Wahrscheinlichkeitsvektor $p = (p_0, \ldots, p_{n-1})$ mit $\sum_{i=0}^{n-1} p_i = 1$ und $p_i \geq 0$ $(0 \leq i < n)$.



Gesucht: `Sample(p)` soll ein j $(0 \leq j < n)$ zurückgeben mit Wahrscheinlichkeit p_j.

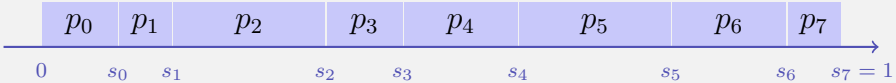
Unfair würfeln




```
static int Sample(double[] p){
  double u = Math.random();
  if (u<p[0]) return 0;
  if (u<p[0]+p[1]) return 1;
  if (u<p[0]+p[1]+p[2]) return 2;
  if (u<p[0]+p[1]+p[2]+p[3]) return 3;
  ...
}
```

Zu umständlich: wir brauchen eine Schleife!

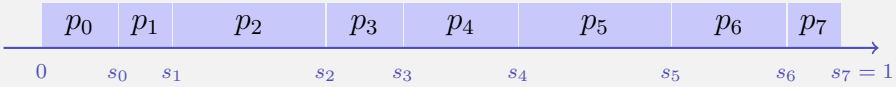
Ein kleiner Trick




```
static int Sample(double[] p){
  double u = Math.random();
  if (u<p[0]) return 0;
  u -= p[0];
  if (u<p[1]) return 1;
  u -= p[1];
  if (u<p[2]) return 2;
  ...
}
```

So müssen wir Summe der p_i nicht mitrechnen.

In eine Schleife




```
static int Sample(double[] p){
  double u = Math.random();
  for (int k = 0; k < p.length−1; ++k){
    if (u<p[k]) return k;
    u −= p[k];
  }
  return p.length−1;
}
```

Quizfrage

Angenommen, die Methode f wird aufgerufen mit einer $k \times k$-Matrix P ($k > 0$) und $n > 0$.
Wie oft wird der Vergleich

$$u < p[k]$$

im schlechtesten Fall ausgeführt? (in Θ Schreibweise!)
Wie lässt sich das verbessern?

```
static int Sample(double[] p){
  double u = Math.random();
  for (int k = 0; k < p.length−1; ++k){
    if (u<p[k]) return k;
    u −= p[k];
  }
  return p.length−1;
}

static void f(double[][] P, int n){
int value=0;
for (int i = 1; i<n; ++i){
       value = Sample(P[value]);
}
```

Fragen oder Anregungen?