Programmierkurs

Vorlesung 04

M.Sc. Laslo Hunhold

Department Mathematik/Informatik
Abteilung Informatik
Universität zu Köln

30. November 2020

Letzte Vorlesung

- Vertiefung Kontrollstrukturen (`break`, `continue`)
- Operationen auf Variablen (Assoziativität, Rang/Präzedenz)
- Wahrheitstabelle BOOLEscher Operatoren
- Unvollständige Auswertung bei `&&` und `||`
- Arrays
- Lokale Variablen (Gültigkeitsbereich, Lebensdauer)

Mehrdimensionale Arrays

- Nicht nur eine, sondern beliebige Dimensionen möglich
- Deklaration: Anzahl der `[]` gibt Dimension an; z.B. ein zweidimensionaler Array:

```
int[][] arr;
```

- Initialisierung: Analog, mit verschachtelten Klammern bei Literalen:

```
arr = new int[2][5];
```

erzeugt einen 2x5-Array,

```
arr = new int[][]{ { 5, 6, 2, 3, 5 }, { 2, 1, 5, 2, 4 }};
```

erzeugt auch einen 2x5-Array und füllt diesen mit Werten

- Zugriff: Analog; z.B. `arr[1]` ist ein Zeiger (Speicheradresse) vom Typ `int[5]` und `arr[1][2]` ist vom Typ `int`
- Speicherstruktur am Beispiel von `int[2][5]`: 1 Array `arr` von 2 Zeigern (`arr[0]` und `arr[1]`), die auf 2 Arrays von jeweils 5 `int` zeigen

Beispiele

Deklaration und Initialisierung

```
int[][][] arr;
arr = new int[2][3][19];
```

```
double[][][][][] darr = null;
```

Doppelschleife

```
double[][] arr = new double[4][15];
for (int i = 0; i < arr.length; i++) {
        for (int j = 0; j < arr[i].length; j++) {
                arr[i][j] = 0.0;
        }
}
```

Gemischte Länge

```
/* Erzeuge nur Array aus 2 Zeigern (int[]) */
int[][] arr = new int[2][];

arr[0] = new int[10];
arr[1] = new int[4];
```

Klassen

Motivation

- Betrachte einen Bruch (z.B. $\frac{1}{3}$)
- Festgelegt durch Zähler 1 und Nenner 3
- Wie im Computer speichern? Fließkommazahl `0.333...` nicht exakt

```
int zaehler = 1;
int nenner = 3;
```

- Was ist bei mehreren Brüchen?
- Zähler und Nenner stehen in Beziehung zueinander und sind alleine bedeutungslos

Lösung

- Klassen binden Daten und Operationen auf Daten; Strukturierung
- Klassen sind komplexe Datentypen (da variable Länge)

```
class Bruch {
        int zaehler;
        int nenner;
}
```

Programmstruktur

- Jede Klasse in einer eigenen `.java`-Datei
- Verwandte Klassen werden in „Packages“ gruppiert
- Es gibt auch ein „Default“-Package bei Nichtverwendung des `package`-Ausdrucks
- `main`-Methode als optionaler (einziger!) Einstiegspunkt in einem Package $\rightarrow$ ausführbares Programm (später)
- Namenskonvention: Package-Bezeichner werden klein geschrieben
- Klassennamen müssen sich innerhalb eines Packages unterscheiden („Namensraum“)

Klassendefinition

- Klasse kann Variablen („Attribute“), Methoden (später) und andere Klassen (später) enthalten (z.B. Variablen `zaehler` und `nenner`)
- Beschreibt Aufbau eines komplexen Datentyps (z.B. `Bruch`)

```
1  ... class Klasse ... {
2          ... Variable;
3          ...
4
5          ... methode(...) {
6                  ...
7          }
8          ...
9
10         ... class Innenklasse ... {
11                 ...
12         }
13         ...
14 }
```

Klasse.java

Deklaration

- Wollen jetzt die Klasse `Bruch` als Datencontainer benutzen, um $\frac{1}{3}$ zu speichern
- Klassenname wird als Typname verwendet

```
Bruch b;
```

- Deklariert Variable mit der Bezeichnung `b` vom Typ `Bruch`
- Klassen sind komplexe Datentypen
- Wir können noch nicht mit `b` arbeiten, da es, wie bei einer Array-Deklaration, noch nicht „existiert“; `b` ist wieder nur ein Zeiger
- `b` kann auch auf `null`, den Nullzeiger, gesetzt werden

Instanziierung

- Analog zum Array
- Erzeuge Instanz einer Klasse („Objekt“)

```
Bruch b;
b = new Bruch();
```

- Virtuelle Maschine reserviert Speicher für die Instanzvariablen (später) in der Klasse (hier `zaehler` und `nenner`)
- Klassenvariablen (Attribut `static`) (später) werden von allen Instanzen geteilt und schon beim Programmstart reserviert; meist für Konstanten, die mit der Klasse zu tun haben
- Speicheradresse wird in `b` (Zeiger) gespeichert

Zugriff

- Operator .
- Zugriff auf eine Variable oder Methode einer Klasse
- Zwei Operanden: Klasse oder Objekt und Name der Variable oder Methode
- Beispiel:

```
Bruch b = new Bruch();
b.zaehler = 1;
b.nenner = 3;
```

Beispiel Instanzvariablen versus Klassenvariablen

```
1  class Main {
2          public static void main(String[] args) {
3                  Test.i = 0; /* Fehler */
4                  Test.k = 5;
5
6                  Test obj = new Test();
7                  obj.i = 3;
8
9                  System.out.println(obj.i);
10                 System.out.println(obj.k); /* Warnung */
11                 System.out.println(Test.k);
12         }
13 }
```

Main.java

```
1  class Test {
2          int i; /* Instanzvariable */
3          static int k; /* Klassenvariable */
4  }
```

Test.java

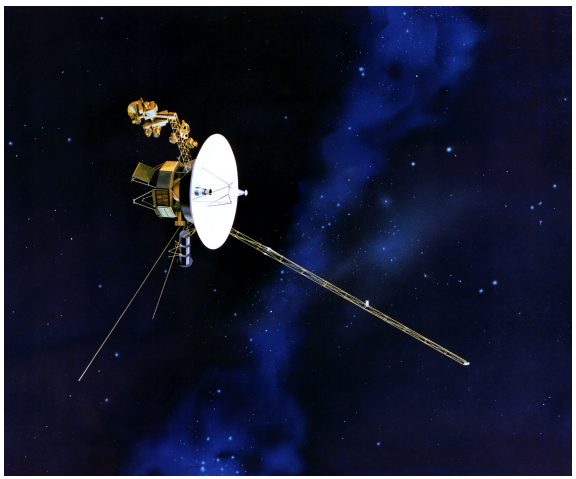
Beispiel Bruch

```
1  package rechner; /* optional, sonst im Default-Package */
2
3  class Main {
4          public static void main(String[] args) {
5                  Bruch a = new Bruch();
6                  a.zaehler = 1;
7                  a.nenner = 3;
8          }
9  }
```

Main.java

```
1  package rechner; /* optional, sonst im Default-Package */
2
3  class Bruch {
4          int zaehler;
5          int nenner;
6  }
```

Bruch.java



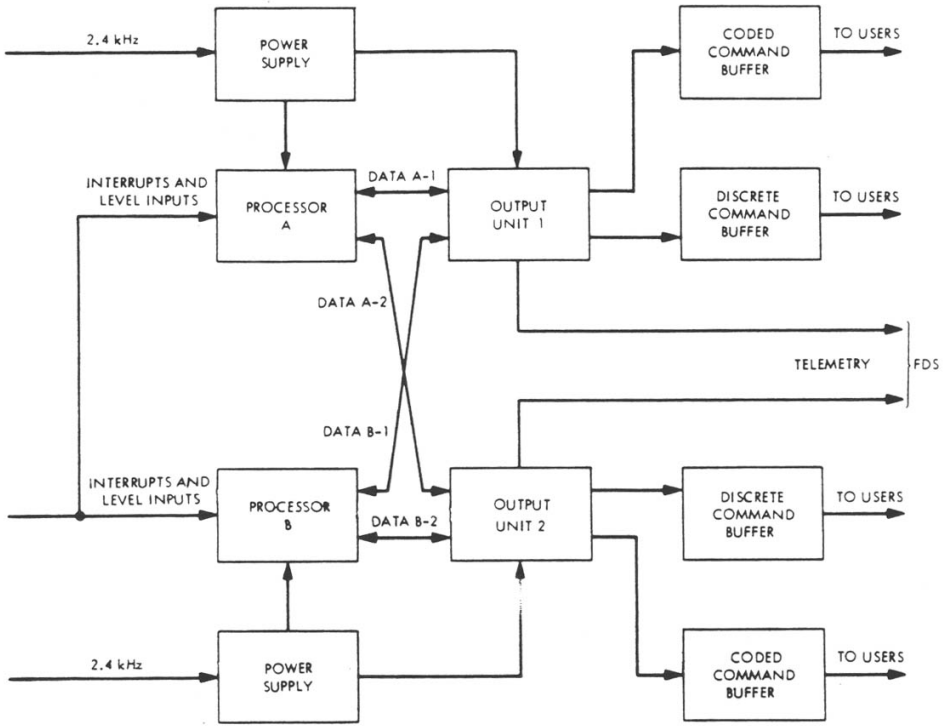


Abbildung: Voyager-Raumsonde (1977) und Diagramm des CCS (Computer Command Subsystem) (Quelle: NASA/JPL, „Viking Orbiter Computer Command Subsystem Hardware“)

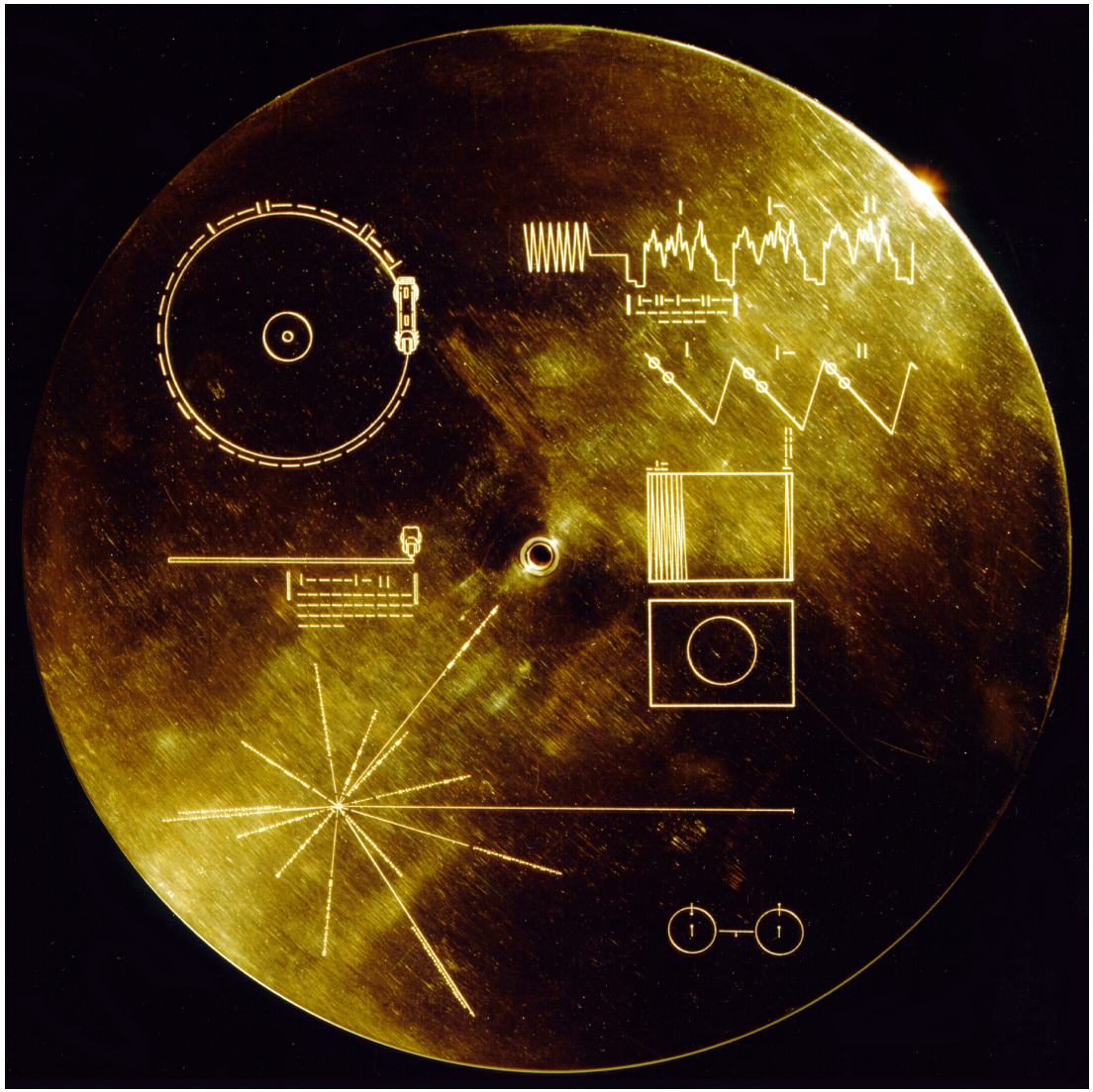


Abbildung: Voyager Golden Record „The Sounds of Earth“ mit Dekodierungsanleitung (Quelle: NASA/JPL)