2. AuD Tafelübung T-C3

Simon Ruderich

3. November 2010

Organisatorisches

Bearbeitung der Praxisaufgaben

- Bereits vorhandene Klassen, Methoden, Variablen, . . . dürfen nicht verändert werden!
- Eigene Klassen, Methoden, Variablen, . . . dürfen natürlich hinzugefügt werden.

Tafelübungen

- keine Anwesenheitspflicht–aber natürliche sinnvolle Übung

Codierung

Zahlensysteme

Die Ziffer an der Stelle *i von rechts* im Zahlensystem zur Basis *b* entspricht dem Wert:

$$a_i \cdot b^i$$

Somit gilt für Zahlen zur Basis *b*:

$$w_{(10)} = a_0 \cdot b^0 + a_1 \cdot b^1 + a_2 \cdot b^2 + \ldots + a_i \cdot b^i$$

Beispiel zur Basis 5

$$a_0 \cdot 5^0 + a_1 \cdot 5^1 + a_2 \cdot 5^2 + a_3 \cdot 5^3 + \ldots =$$
$$a_0 \cdot 1 + a_1 \cdot 5 + a_2 \cdot 25 + a_3 \cdot 125 + \ldots$$
$$103_{(5)} = 1 \cdot 5^2 + 0 \cdot 5^1 + 3 \cdot 5^0 = 25 + 0 + 3 = 28_{(10)}$$

Codierung

Umrechnung durch Division

$$97_{(10)} = \ ?_{(7)}$$

$$\begin{aligned} 97 : 7 &= 13\ R\ 6 \\ 13 : 7 &= \ 1\ R\ 6 \\ 1 : 7 &= \ 0\ R\ 1 \end{aligned}$$

$$97_{(10)} = 166_{(7)}$$

Codierung

Wichtige Zahlensysteme

- Binärsystem, zur Basis 2: $13_{(10)} = 1101_{(2)}$
- Oktalsystem, zur Basis 8: $33_{(10)} = 041_{(8)}$
- Hexadezimalsystem, zur Basis 16: $92_{(10)} = 5C_{(16)}$

Hexadezimalsystem

- zusätzliche „Ziffern“ *A*–*F* um 10–15 darstellen zu können

Codierung

Zusammenhang Binär-, Oktal- und Hexadezimalsystem

- Gruppierung im Binärsystem ermöglicht schnelle Umrechnung
- 4 Stellen im Binärsystem $\Rightarrow$ Hexadezimalsystem ($2^4 = 16$)
- 3 Stellen im Binärsystem $\Rightarrow$ Oktalsystem ($2^3 = 8$)

Beispiel

$100111011011_{(2)} = 1001\ 1101\ 1011 = 9DB_{(16)}$
$100111011011_{(2)} = 100\ 111\ 011\ 011 = 04733_{(8)}$

Codierung

Weitere Beispiele

- $77_{(10)} = ?_{(2)} = ?_{(16)}$
- $?_{(10)} = 11\ 1101_{(2)} = ?_{(16)}$
- $?_{(10)} = ?_{(2)} = AA_{(16)}$
- $5213_{(10)} = ?_{(2)} = ?_{(16)}$
- $?_{(10)} = 1100\ 1010\ 1111\ 1110_{(2)} = ?_{(16)}$

Codierung

Weitere Beispiele

- $77_{(10)} = 100\ 1101_{(2)} = 4D_{(16)}$
- $?_{(10)} = 11\ 1101_{(2)} = ?_{(16)}$
- $?_{(10)} = ?_{(2)} = AA_{(16)}$
- $5213_{(10)} = ?_{(2)} = ?_{(16)}$
- $?_{(10)} = 1100\ 1010\ 1111\ 1110_{(2)} = ?_{(16)}$

Codierung

Weitere Beispiele

- $77_{(10)} = 100\ 1101_{(2)} = 4D_{(16)}$
- $61_{(10)} = 11\ 1101_{(2)} = 3D_{(16)}$
- $?_{(10)} = ?_{(2)} = AA_{(16)}$
- $5213_{(10)} = ?_{(2)} = ?_{(16)}$
- $?_{(10)} = 1100\ 1010\ 1111\ 1110_{(2)} = ?_{(16)}$

Codierung

Weitere Beispiele

- $77_{(10)} = 100\ 1101_{(2)} = 4D_{(16)}$
- $61_{(10)} = 11\ 1101_{(2)} = 3D_{(16)}$
- $170_{(10)} = 1010\ 1010_{(2)} = AA_{(16)}$
- $5213_{(10)} = ?_{(2)} = ?_{(16)}$
- $?_{(10)} = 1100\ 1010\ 1111\ 1110_{(2)} = ?_{(16)}$

Codierung

Weitere Beispiele

- $77_{(10)} = 100\ 1101_{(2)} = 4D_{(16)}$
- $61_{(10)} = 11\ 1101_{(2)} = 3D_{(16)}$
- $170_{(10)} = 1010\ 1010_{(2)} = AA_{(16)}$
- $5213_{(10)} = 1\ 0100\ 0101\ 1101_{(2)} = 145D_{(16)}$
- $?_{(10)} = 1100\ 1010\ 1111\ 1110_{(2)} = ?_{(16)}$

Codierung

Weitere Beispiele

- $77_{(10)} = 100\ 1101_{(2)} = 4D_{(16)}$
- $61_{(10)} = 11\ 1101_{(2)} = 3D_{(16)}$
- $170_{(10)} = 1010\ 1010_{(2)} = AA_{(16)}$
- $5213_{(10)} = 1\ 0100\ 0101\ 1101_{(2)} = 145D_{(16)}$
- $51966_{(10)} = 1100\ 1010\ 1111\ 1110_{(2)} = CAFE_{(16)}$

Codierung

Der ASCII-Code

- ASCII = American Standard Code for Information Interchange
- Jedes Zeichen hat einen bestimmten Code
- `$ man ascii`
  zeigt ASCII-Tabelle an

Beispiele

- $41_{(16)}\ 165_{(8)}\ 68_{(10)}$ = AuD
- $101_{(8)}\ 53_{(16)}\ 103_{(8)}\ 73_{(10)}\ 111_{(8)} = ?$

Codierung

Der ASCII-Code

- ASCII = American Standard Code for Information Interchange
- Jedes Zeichen hat einen bestimmten Code
- `$ man ascii`
  zeigt ASCII-Tabelle an

Beispiele

- $41_{(16)}\ 165_{(8)}\ 68_{(10)}$ = AuD
- $101_{(8)}\ 53_{(16)}\ 103_{(8)}\ 73_{(10)}\ 111_{(8)}$ = ASCII

Codierung

Entscheidungsgehalt

Der Entscheidungsgehalt ist die minimale Anzahl an Bits, um ein Element einer Menge eindeutig identifizieren zu können.

$$H = \lceil \log_2 n \rceil = \left\lceil \frac{\lg n}{\lg 2} \right\rceil$$

Beispiele

- Sekunden an einem Tag:
  $H = \lceil \log_2(24 \cdot 3600) \rceil \approx \lceil 16.3987 \rceil = 17$
- Alle angebotenen Übungen (37 Übungen):
  $H = \lceil \log_2 37 \rceil \approx \lceil 5.2094 \rceil = 6$

Strings

Die `String`-Klasse

- Zeichenkette (auch Zahlen, siehe ASCII)
- Groß- und Kleinschreibung wird beachtet:
  `"hallo"` $\neq$ `"HALLO"`
- Vergleich mit `equals()`-Methode
  (Groß- und Kleinschreibung wird beachtet!)
- `equalsIgnoreCase()`-Methode
  (ignoriert Groß- und Kleinschreibung)

Strings und Zahlen

Strings und Zahlen

- `"12345"` ist keine Zahl sondern Zeichenkette!
- zum Rechnen Umwandlung in **int**, **double**, **long**, ... nötig
- `Integer.parseInt()`, `Double.parseDouble()`, `Long.parseLong()` ...

Strings und Zahlen

Beispiel

```
public static void main(String[] args) {
    int a    = Integer.parseInt(args[0]);
    double b = Double.parseDouble(args[1]);
    System.out.println((a * 3) + "; " + (b * 0.3));
}
```

Ausgabe

```
> java Test 4 0.5
12; 0.15
```

Arrays (Felder/Reihungen)

Arrays

- speichern mehrere Elemente des *gleichen* Typs (Liste von Elementen)
- Zugriff über Index (Integer)
- erstes Element hat Index 0
- `array.length` liefert Länge des Arrays
- letzte Position hat Index: `array.length`-1

Arrays

Beispiel

```
int[] zahlen = new int[5]; // Platz fuer 5 ints
zahlen[0] = 3;
zahlen[3] = 7;
zahlen[5] = 6; // Fehler! Letzer Index ist 4

String[] strings = new String[]{"", "a", "b", "ab"};
System.out.println(strings[0]); // gibt "" aus
System.out.println(strings[2]); // gibt "b" aus
System.out.println(strings[strings.length-1]);
    // gibt "ab" aus
```

Arrays

Beispiel

```
int[] zahlen = new int[5];
zahlen[0] = 3;
zahlen[3] = 7;

for (int i = 0; i < zahlen.length; i++) {
    System.out.print(zahlen[i] + ",");
}
```

Arrays

Beispiel

```
int[] zahlen = new int[5];
zahlen[0] = 3;
zahlen[3] = 7;

for (int i = 0; i < zahlen.length; i++) {
    System.out.print(zahlen[i] + ",");
}
```

Ausgabe

```
3,0,0,7,0,
```

Mehrdimensionale Arrays

- Arrays, die wiederum Arrays enthalten
- „beliebig“ wiederholbar (2D-, 3D-, 4D-, ... Arrays)
- nicht jeder Platz muss besetzt sein!

Beispiel

```
String[][] strings = new String[5][5];
// strings kann maximal 25 Strings speichern

strings[0][2] = "foo";
strings[4][3] = "bar";

strings[2] = null;
```

Mehrdimensionale Arrays

Beispiel

```
String[][] strings = new String[5][5];
strings[0][2] = "foo"; strings[4][3] = "bar";
strings[2] = null;
for (int i = 0; i < strings.length; i++) {
    System.out.print(strings[i] + ",");
}
```

Mehrdimensionale Arrays

Beispiel

```
String[][] strings = new String[5][5];
strings[0][2] = "foo"; strings[4][3] = "bar";
strings[2] = null;
for (int i = 0; i < strings.length; i++) {
    System.out.print(strings[i] + ",");
}
```

Ausgabe

```
[Ljava.lang.String;@1fee6fc,[Ljava.lang.String;
    @1eed786,null,[Ljava.lang.String;@187aeca,
    [Ljava.lang.String;@e48e1b,
```

Mehrdimensionale Arrays

Beispiel

```
String[][] strings = new String[5][5];
strings[0][2] = "foo"; strings[4][3] = "bar";
strings[2] = null;
for (int i = 0; i < strings.length; i++) {
    for (int j = 0; j < strings[i].length; j++) {(*)
        System.out.print(strings[i][j] + ",");
    }
}
```

Mehrdimensionale Arrays

Beispiel

```
String[][] strings = new String[5][5];
strings[0][2] = "foo"; strings[4][3] = "bar";
strings[2] = null;
for (int i = 0; i < strings.length; i++) {
    for (int j = 0; j < strings[i].length; j++) {(*)
        System.out.print(strings[i][j] + ",");
    }
}
```

Ausgabe

```
null,null,foo,null,null,null,null,null,null,null,
Exception in thread "main"
    java.lang.NullPointerException at (*)
```

Mehrdimensionale Arrays

Beispiel

```
String[][] strings = new String[5][5];
strings[0][2] = "foo"; strings[4][3] = "bar";
strings[2] = null;
for (int i = 0; i < strings.length; i++) {
    if (strings[i] == null) {
        System.out.println("skipping");
        continue;
    }
    for (int j = 0; j < strings[i].length; j++) {
        System.out.print(strings[i][j] + ",");
    }
    System.out.print("\n");
}
```

Mehrdimensionale Arrays

Ausgabe

```
null , null , foo , null , null ,
null , null , null , null , null ,
skipping
null , null , null , null , null ,
null , null , null , bar , null ,
```

Mehrdimensionale Arrays

Beispiel

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

System.out.println(zahlen[0][0]); // null

zahlen[4][3] = new int[]{ 3, 8, 0 };
zahlen[4][4] = new int[4];
zahlen[4][4][1] = 42;
zahlen[4][4][3] = 9;

// Exception, da Array nur 3 lang
int a = zahlen[4][3][3];
// funktioniert, da Array 4 lang
int b = zahlen[4][4][3];

zahlen[2] = null;
```

`Math`-Klasse

Mathematische Berechnungen

- $\cos(x) =$ `Math.cos(x)`
- $\sqrt{x} =$ `Math.sqrt(x)`
- $x^y =$ `Math.pow(x, y)`
- $\ln(x) = \log_e(x) =$ `Math.log(x)`
- $\lg(x) = \log_{10}(x) =$ `Math.log10(x)`

Zufallszahlen

- `Math.random()` erzeugt zufälligen `double`-Wert zwischen 0.0 (einschließlich) und 1.0 (ausschließlich)

Math-Klasse

Funktionsberechnung

$$f(x) = \sqrt{x^2 + \cos(x)}$$

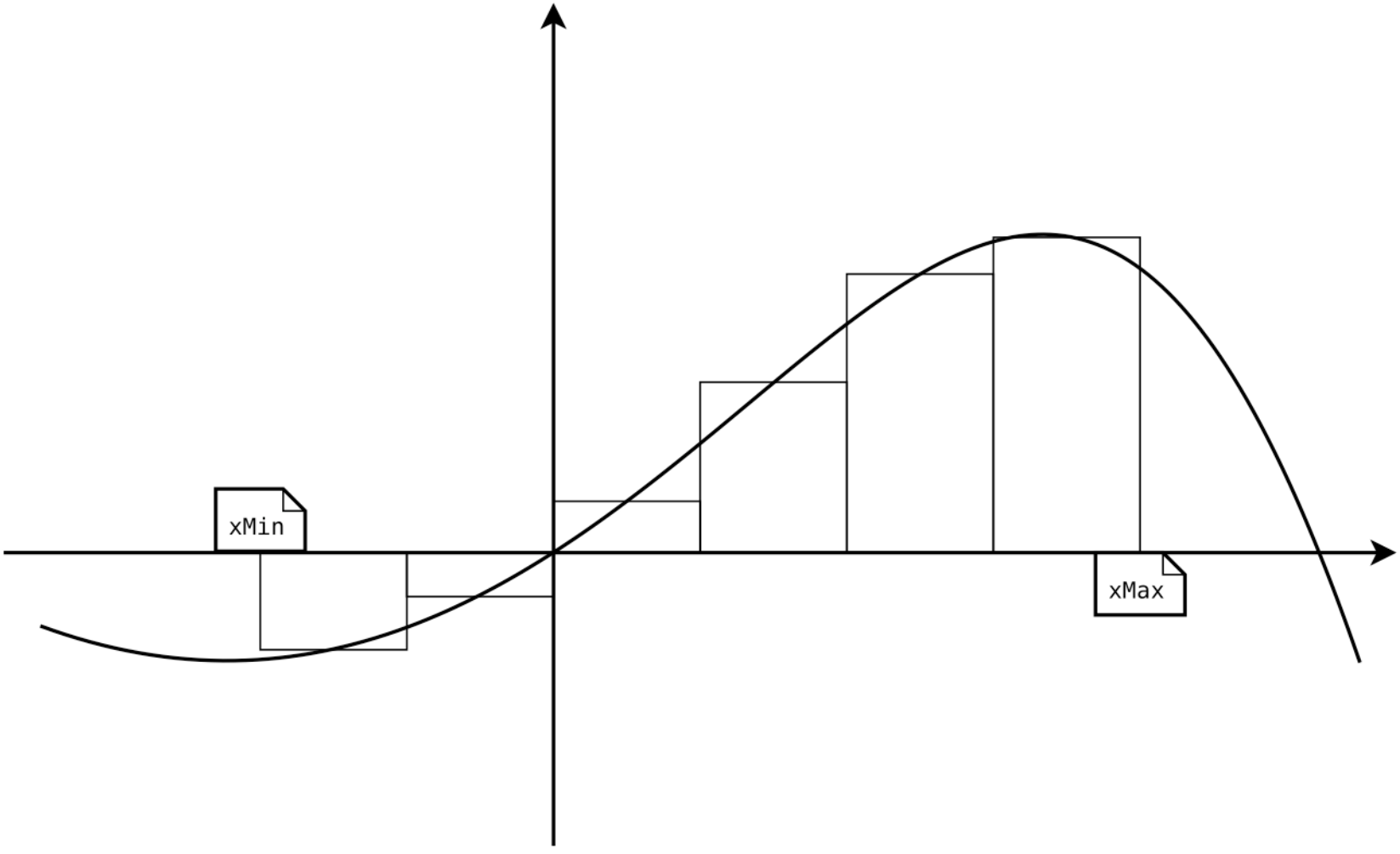
```
double r = Math.sqrt(Math.pow(x, 2) + Math.cos(x));
System.out.println("f(" + x + ") = " + r);
```

```
> f(2) = 1.893106749090726...
```

Zufallszahlen generieren

```
double random = Math.random();
```

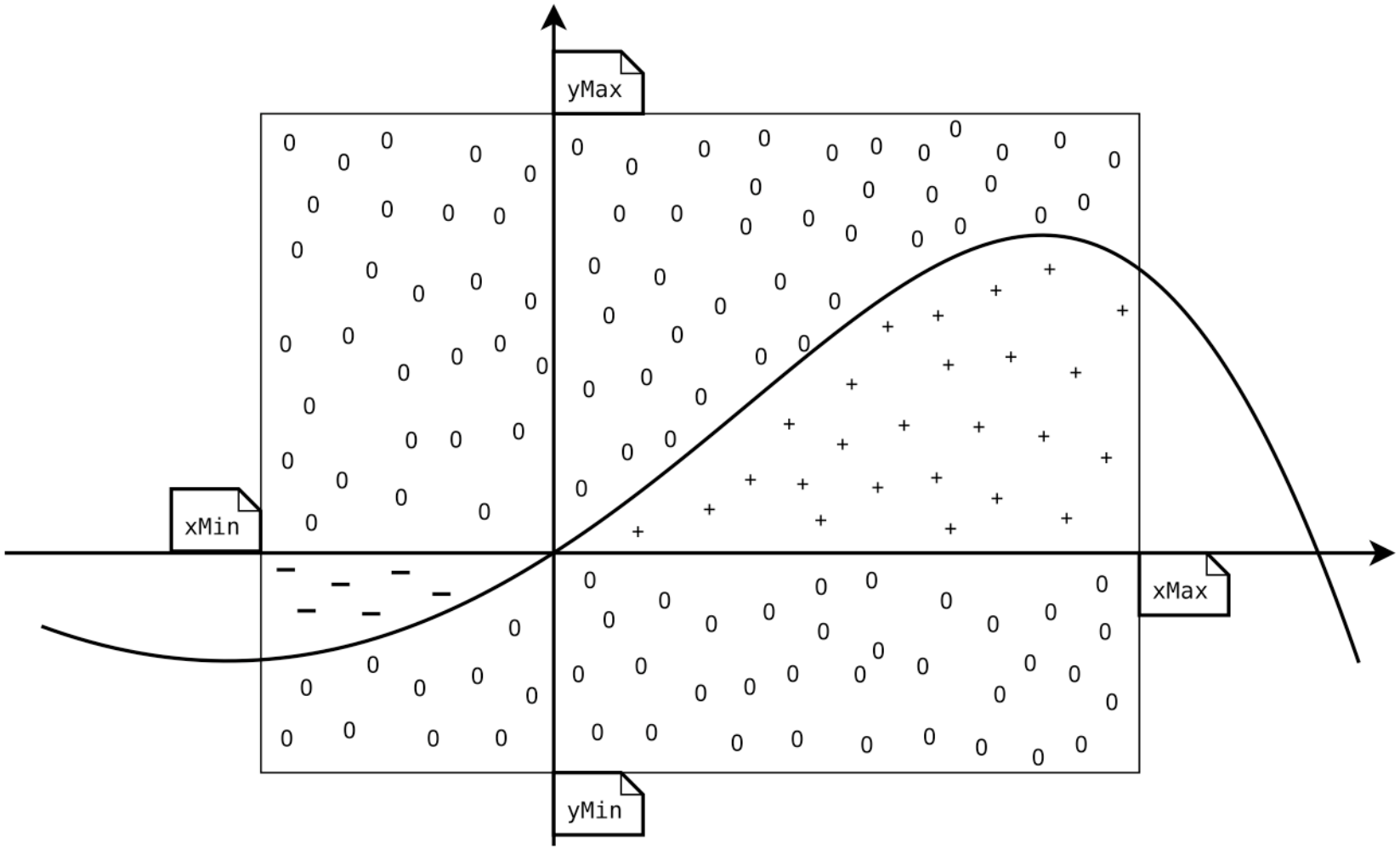
Integration durch Rechtecke



Integration durch Rechtecke

- Unterteilung der x-Achse von *xMin* bis *xMax* in n Rechtecke
- Berechnung der Funktionswerte jeweils an der Mitte der Rechtecke
- Summation der Flächen der Rechtecke

Integration durch Monte-Carlo



Integration durch Monte-Carlo

- Auswahl eines Rechtecks (*xMin*, *xMax*, *yMin*, *yMax*)
- zufällige Auswahl eines *x*- und *y*-Werts innerhalb des Rechtecks
- liegt der Punkt „unterhalb“ des Graphen
  - ja: Treffer-Zähler um eins erhöhen falls der Funktionswert positiv ist;
    Treffer-Zähler um eins erniedrigen andernfalls
  - nein: tue nichts
- Fläche des Rechtecks mit dem Verhältnis Treffer/Versuche multiplizieren

Fragen

Fragen?