



Informatik II Tutorium

SS 05

Vorlesung:

Prof. Dr. J. Calmet

Übungsleitung:

Dipl.-Math. Ralf Eberhardt

Tutorium:

12 & 13

Tutor:

Christian Maier

Tutorium 11: Mittwoch 06. Juli 2005



Übersicht heute:

- Themenliste Klausur (ohne Gewähr!)
- Probeklausur
- Aufwandspezifikation
- Java Vererbung
- Java Exceptions



Anmerkung:

- Am **11.07.2005** sollten **ALLE** ihre Praxis Aufgaben (Üb. 9 & 11) in der Rechnerübung vorrechnen!!!
- Programme die **NICHT** rechtzeitig per E-Mail an mich gingen, können auch nicht in der Rechnerübung abgenommen werden!!!!
- Wer unentschuldig der Rechnerübung ferngeblieben ist und **NICHT** seine Programme vorgeführt hat, hat **keinen** Anspruch auf „Nachprüfung“!!!



Tutorium 11

Letztes Tutorium



- Prädikatenlogik
 - Struktur, Interpretation
 - Pränexform, bereinigte Form
 - Skolemform
 - Resolution
 - allgemeinsten Unifikator
- Datentypen
 - Allgemeine Datentypen, z.B. Bool, Schlange, Liste, ...
 - verkettete Liste
 - Heap
 - Bäume, Suchbäume, RS-Bäume*
 - Hashtabellen
 - Graphen/Relationen
- Algorithmenschemata
 - Greedy
 - Teile und Herrsche
 - Dynamische Programmierung (Trellis, Viterbi)
 - Probabilistische Algorithmen*
- Algorithmen
 - Sortieren
 - Suchen (Suchbäume, Hashtabellen)
 - Graphenalgorithmen (Kruskal)
 - KMP
 - Booyer-Moore
 - Polynome (Faltung, Karatsuba)
- Analyse
 - o , O , Ω , Θ -Notation
 - vom Programm zur Rekurrenz
 - Lösen von Rekurrenzen (Mastertheorem*, generierende Funktion, Raten & Induktion)
 - Aufwandsabschätzung
- OOP
 - UML
 - Vererbung
 - ...?

*) Diese Themen werden wahrscheinlich höchstens in einer Multiple-Choice-Frage geprüft. Alle Angaben ohne Gewähr!



- Inoffizielle Probeklausur findet am 13.07.2005 im -101 von 1730h bis 1830h statt
- Anschließend kurze Besprechung & Austeilung der Lösungen.
- Anmeldungen unter: tutor@christianmaier.name
- Anmeldeschluss ist: So. 10.07.05
- Es werden Kopien der Aufgabenstellung & anschließend Lösung der Probeklausur ausgeteilt, wer sich nicht anmeldet hat keinen Anspruch darauf
- ***Hinweis: Es handelt sich um eine INOFFIZIELLE Probeklausur, diese wurde nicht durch die offiziellen Organe angeschaut.***



Aufwandspezifikation



O-Notation

$$o(f) = \{g : \forall k > 0 : \exists N > 0 : \forall n \geq N : g(n) \leq k \cdot f(n)\}$$

$$O(f) = \{g : \exists N > 0, k > 0 : \forall n \geq N : g(n) \leq k \cdot f(n)\}$$

$$\Theta(f) = \{g : \exists N > 0, k > 0, \ell > 0 : \forall n \geq N : k \cdot f(n) \leq g(n) \leq \ell \cdot f(n)\}$$

$$\Omega(f) = \{g : \exists N > 0, k > 0 : \forall n \geq N : k \cdot f(n) \leq g(n)\}$$

$$\omega(f) = \{g : \forall k > 0 : \exists N > 0 : \forall n \geq N : k \cdot f(n) \leq g(n)\}$$

o	O	Θ	Ω	ω
$<$	\leq	$=$	\geq	$>$



Aus der Vorlesung:

Ein paar Rechenoperationen

Wenn $f(n) = O(r(n))$ und $g(n) = O(s(n))$, dann ist $f(n) + g(n) = O(r(n) + s(n))$

Wenn $f(n) = O(r(n))$ und $g(n) = O(s(n))$, dann ist $f(n) * g(n) = O(r(n) * s(n))$

aber nicht:

Wenn $f(n) = O(r(n))$ und $g(n) = O(s(n))$, dann $f(n) - g(n) = O(r(n) - s(n))$

auch nicht:

Wenn $f(n) = O(r(n))$ und $g(n) = O(s(n))$, dann $f(n) / g(n) = O(r(n) / s(n))$

Beweis?



Aufwandspezifikation

Gegeben ist die Funktion `Fakultaet` zur Berechnung der Fakultätsfunktion. Berechnen Sie mit Hilfe einer geeigneten Rekurrenzrelation den Aufwand dieser Funktion, und geben Sie ihn im O-Kalkül an.

```
int Fakultaet(int n) {  
    if (n==1) return 1;  
    else return n*Fakultaet(n-1);  
}
```

Lösung:

$$\begin{aligned} \text{Laufzeit } T(n) &= d, && \text{für } n \leq 1 \\ &= c + T(n-1), && \text{für } n > 1 \end{aligned}$$

allgemein ergibt sich daher für $n > 1$:

$$T(n) = c \cdot i + T(n-i) = c \cdot (n-1) + T(1) = c \cdot (n-1) + d \in O(n)$$



Berechnen Sie mit Hilfe des O-Kalküls eine möglichst kleine obere Schranke $g(n)$ jeweils für $f(n)$, so dass gilt $f_1(n) = O(g_1(n))$ und $f_2(n) = O(g_2(n))$ wobei:

$$1. \quad f_1(n) = \frac{n^3 + 3n^2 + 2n}{3}$$

Der Rechenweg muss ersichtlich sein.

Lösung:

Nach Definition des O-Kalküls gilt:

$$f_1(n) = \frac{1}{3}n^3 + n^2 + \frac{2}{3}n \leq \frac{1}{3}n^3 + n^3 + \frac{2}{3}n^3 = 2n^3 \Rightarrow f_1(n) \in O(g_1(n))$$

mit $g_1(n) = n^3$



Berechnen Sie mit Hilfe des O-Kalküls eine möglichst kleine obere Schranke $g(n)$ jeweils für $f(n)$, so dass gilt $f_1(n) = O(g_1(n))$ und $f_2(n) = O(g_2(n))$ wobei:

$$2. \quad f_2(n) = \sqrt{n^3 + n^2}$$

Der Rechenweg muss ersichtlich sein.

Lösung:

Nach Definition des O-Kalküls gilt:

$$f_2(n) = \sqrt{n^3 + n^2} \leq \sqrt{n^3 + n^3} = \sqrt{2n^3} = \sqrt{2}n^{\frac{3}{2}} \Rightarrow f_2(n) \in O(g_2(n))$$

$$\text{mit } g_2(n) = n^{\frac{3}{2}}$$



Aufwandsvergleich unterschiedlicher Sortieralgorithmen

Analysieren Sie den folgenden Sortieralgorithmus:

```
class RandomSortAlgorithm extends SortAlgorithm {
    private boolean wrong(int a[]) throws Exception {
        for (int i=1;i<a.length;i++)
            if (a[i-1]>a[i])
                return true;
        return false;
    }

    void sort(int a[]) throws Exception {
        boolean cont=true;
        while(cont) {
            int i=(int)(a.length * Math.random());
            int j=(int)(a.length * Math.random());
            int k=a[i];
            a[i]=a[j];
            a[j]=k;
            cont=wrong(a);
        } } }
}
```

Berechnen Sie den Erwartungswert der Anzahl der Schleifendurchläufe für den Algorithmus Random Sort. Warum liefert der Algorithmus ein unbefriedigendes Ergebnis?

Random Sort: Erwartungswert der Schleifendurchläufe aufgrund der möglichen Anordnungen der n Elemente: $O(n!)$. Unbefriedigendes Ergebnis, da Algorithmus nicht notwendiger Weise terminiert (ist daher kein Monte Carlo Algorithmus nach Definition in der Vorlesung).



Aufwandsvergleich unterschiedlicher Sortieralgorithmen

Analysieren Sie den folgenden Sortieralgorithmus:

```
class ExchangeSortAlgorithm extends SortAlgorithm
{
    void sort(int a[]) throws Exception {
        for (int i=0;i<a.length-1;i++)
            for (int j=i+1;j<a.length;j++)
                if (a[j]<a[i]) {
                    int k=a[i];
                    a[i]=a[j];
                    a[j]=k;
                }
    }
}
```

Berechnen Sie den Aufwand für den
Algorithmus Exchange Sort

Exchange Sort: $O(n^2)$ in jedem Fall).



Aufwandsvergleich unterschiedlicher Sortieralgorithmen

Analysieren Sie den folgenden Sortieralgorithmus:

```
class InsertSortAlgorithm extends SortAlgorithm
{
    void sort(int a[]) throws Exception {
        for (int j=0;j<a.length;j++) {
            int k=a[j];
            int i=j-1;
            while(i>=0 && a[i]>k) {
                a[i+1]=a[i];
                i--;
            }
            a[i+1]=k;
        }
    }
}
```

Berechnen Sie den Aufwand für den
Algorithmus Insert Sort

Insert *Worst Case*
Exchange Sort: $O(n^2)$ im *Best Case*, $O(n)$ bei sortierter Liste.



Java

Vererbung/Exceptions

(by Joachim Wilke)



- Definition einer Relation zwischen zwei Klassen durch “extends” in der Klassendeklaration:
- `class B extends A { ... }`
- Klasse B ist damit Unterklasse von Klasse A
- Welche Klasse ist Oberklasse von Klasse C ?
- `class C { ... }`
- Antwort: Klasse “Object” ist allgemeine Oberklasse.

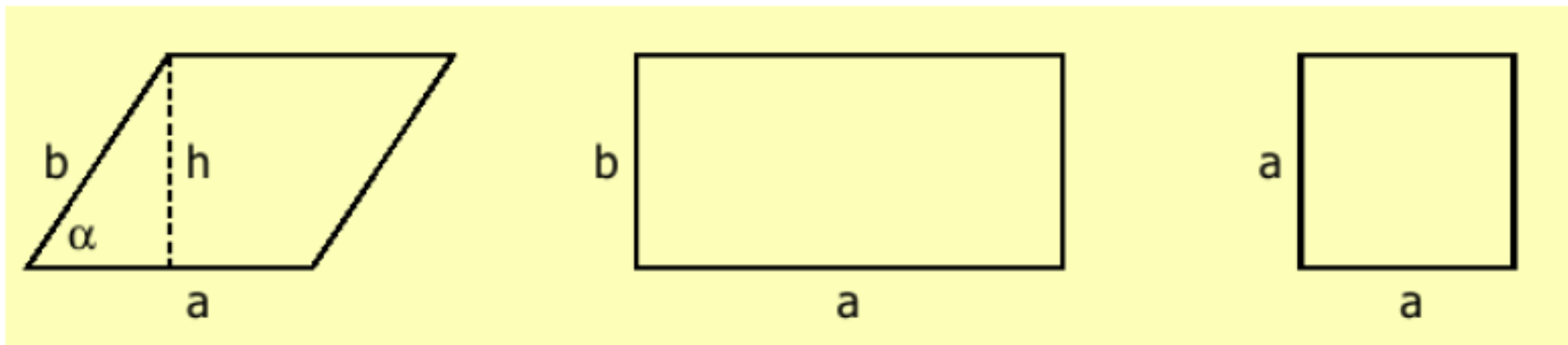


- Vererbt werden
 - Instanzattribute
 - Instanzmethoden
- Nicht vererbt werden
 - mit `private` gekennzeichnete Attribute und Methoden
 - Klassenattribute (“`static`”)
 - Klassenmethoden (“`static`”)
 - Konstruktoren
- Nur einfache Vererbung. Mehrfache Vererbung nicht direkt, sondern nur über sog. “`interfaces`” möglich.



Vererbung in Java - Beispiel

- Rechtecke, Quadrate und Parallelogramme:
 - ein Rechteck ist ein (spezielles) Parallelogramm
 - ein Quadrat ist ein (spezielles) Rechteck





Vererbung in Java - Beispiel

```
public class Parallelogrm {
    private double seite1,seite2,winkel;
    public Parallelogrm(double a,double b,double w) {
        // impliziter Aufruf von super()
        seite1 = a; seite2 = b; winkel = w;
    }
    public double flaeche() {
        return seite1*seite2*Math.sin(winkel);
    }
    public double umfang() {
        return 2*(seite1+seite2);
    }
}
```



Vererbung in Java - Beispiel

```
public class Rechteck extends Parallelogramm {
    public Rechteck(double seite1, double seite2) {
        super(seite1, seite2, Math.PI/2);
    }
    // Methoden flaeche(), umfang() werden geerbt
}

public class Quadrat extends Rechteck {
    public Quadrat(double seite) {
        super(seite, seite);
    }
    // Methoden flaeche(), umfang() werden geerbt
}
```

Wird der Oberklassenkonstruktor nicht explizit aufgerufen, wird der Defaultkonstruktor implizit mit super() aufgerufen.



Vererbung in Java - Beispiel

- Überschreiben ist die Neudefinition geerbter Instanzattribute und -methoden.

```
public class Rechteck extends Parallelogramm {  
    protected double seite1, seite2;  
    public Rechteck(double seite1, double seite2) {  
        this.seite1 = seite1; this.seite2 = seite2;  
    }  
    public double flaeche() {  
        return seite1*seite2;  
    }  
    public double umfang() {  
        return 2*(seite1+seite2);  
    }  
}
```

*Können Konstruktoren
überschrieben werden?*



Beispiel:

```
class A { Attribut f; }  
class B extends A { Attribut f; }  
class C extends B { Attribut f; }
```

Zugriff auf das Attribut f innerhalb der Klasse C

- f bezeichnet f der Klasse C
- this.f bezeichnet f der Klasse C
- super.f bezeichnet f der Klasse B
- ((B)this).f bezeichnet f der Klasse B
- ((A)this).f bezeichnet f der Klasse A
- super.super.f ist nicht erlaubt



- eine Exception in Java ist ein echtes Objekt
- eine Exception wird also mit “new” erzeugt
- eine Exception wird mit “throw” geworfen
- alle Exceptions erben von der Klasse “Exception”

Beispiel:

```
if (month < 1 || month > 12)
    throw new IllegalArgumentException();
```



Exceptions in Java

- eine geworfene Exception muss gefangen werden
- der Fänger muss Exceptions eines Typs erwarten

Beispiel:

```
try {  
    ...  
} catch (ExceptionType1 e1) {  
    ...  
} catch (ExceptionType2 e2) {  
    ...  
} ...
```



Exceptions in Java - Beispiel

```
byte[] text = new byte[50];  
  
try {  
    FileInputStream f = new  
        FileInputStream("input.txt");  
  
    f.read(text);  
  
    f.close();  
  
} catch (FileNotFoundException e) {  
    System.err.println("Datei nicht gefunden.");  
  
} catch (IOException e) {  
    System.err.println("Datei konnte nicht gelesen  
        werden.");  
  
}
```



Exceptions in Java - Beispiel

```
bsp4.java:6: unreported exception:  
java.io.FileNotFoundException  
must be caught or declared to be thrown
```

```
FileInputStream f = new FileInputStream("input.txt");
```

Abhilfe:

```
void readFile ... throws FileNotFoundException {  
  
...  
FileInputStream f = new FileInputStream("input.txt");  
...  
}
```



Wieder a bissala Sarkasmus ?



Aus den Anfängen des Volksentscheids.