Informatik I WS 07/08

Tutorium 24

17.01.08

Bastian Molkenthin

E-Mail: infotut@sunshine2k.de

Web: http://infotut.sunshine2k.de

Universität Karlsruhe (TH)
Forschungsuniversität · gegründet 1825

Organisatorisches

- I*noffizielle* Info-1 Probeklausur findet am Samstag, 19.1.2008 statt.

Wann?	13.00 Uhr - ca. 14.30 Uhr
Wo?	Audimax

➡ Anmeldung heute noch möglich!

➡ http://anmeldungipk.webhop.net

- 3-Wochen-Regelung

- Wer mehr als 300 Theoriepunkte und 120 Praxispunkte hat, hat den Schein sicher.
  Ihr müsst in diesem Fall eure Blätter/Aufgaben nicht mehr zwingend abgeben, dürft aber.

- Anmeldung zur richtigen Info1-Klausur ab 21.1 möglich!
  Genaueres ist auf der Info1-Seite unter Aktuelles nachzulesen.

Kleine Wiederholung

Der Reguläre Ausdruck ab*c*a…

… beschreibt eine CH-3 Sprache

… beschreibt die Sprache $L_1 = \{ab^nc^na \mid n \in \mathbb{N}_0\}$

… beschreibt eine endliche Sprache

Gegeben $G_2 = (\{a, b\}, \{A,B\}, \{A \rightarrow Ab|Ba|a,\ B \rightarrow Aa|b\}, A)$. $L(G_2)$…

… enthält unendlich viele Elemente

… kann nicht durch eine EBNF dargestellt werden

… wird von einem endlichen Akzeptoren A akzeptiert.

Ein Akzeptor, bei dem der Startzustand auch Endzustand ist…

… akzeptiert alle Worte $\in \Sigma^*$

… akzeptiert alle Worte $\in \Sigma$

… akzeptiert auch immer das leere Wort ε

Kleine Wiederholung - Lösung

Der Reguläre Ausdruck ab*c*a…

… beschreibt eine CH-3 Sprache

… beschreibt die Sprache $L_1 = \{ab^nc^na \mid n \in \mathbb{N}_0\}$

… beschreibt eine endliche Sprache

Gegeben $G_2 = (\{a, b\}, \{A,B\}, \{A \rightarrow Ab|Ba|a,\ B \rightarrow Aa|b\}, A)$. $L(G_2)$…

… enthält unendlich viele Elemente

… kann nicht durch eine EBNF dargestellt werden

… wird von einem endlichen Akzeptoren A akzeptiert.

Ein Akzeptor, bei dem der Startzustand auch Endzustand ist…

… akzeptiert alle Worte $\in \Sigma^*$

… akzeptiert alle Worte $\in \Sigma$

… akzeptiert auch immer das leere Wort ε

Prozesse

Was ist ein *Prozess*?

Ein Prozess ist ein Programm und seine Daten, dass sich im Hauptspeicher befindet und vom Prozessor ausgeführt werden kann.

Prozess ≠ Programm! Was ist ein *Programm*?

Folge von Anweisungen (hinterlegt z.B. als Datei auf dem Hintergrundspeicher)

Welchen Status kann ein Prozess besitzen?

Erzeugt	Der Prozess wurde erstellt und benötigt noch Ressourcen (z.B. Daten, freier Speicher, etc.)
Bereit	Der Prozess ist zur Ausführung bereit.
Laufend	Der Prozess wird gerade auf dem Prozessor ausgeführt.
Blockiert	Der Prozess wartet auf ein Ereignis (z.B. Daten, Betriebsmittel)
Beendet	Prozess ist beendet, befindet sich aber noch im Hauptspeicher

Prozesse (2)

Betriebsmittel Prozessor:

Es befinden sich mehrere Prozesse im Speicher, aber nur einer kann auf Prozessor laufen. Man muss sich Gedanken machen wie man die Rechenzeit „gerecht“ verteilt, damit z.B. wichtige Prozesse nicht nach weniger wichtigen Prozessen ausgeführt werden bzw. ein zeitintensiver Prozess nicht einen anderen unnötig lange warten lässt.

⇨ Scheduler wählt Prozess mittels einer Strategie aus

Auswahlstrategien unterscheiden sich in…

Durchsatz	Möglichst hohe Anzahl bearbeiteter Prozesse pro Zeiteinheit
Verweilzeit	Gesamtzeit des Prozesses in der Rechenanlage soll so gering wie möglich sein
Wartezeit	Möglichst kurze Gesamtzeit, in der der Prozess im Zustand „bereit“ ist
Antwortzeit	Möglichst kurze Reaktionszeit des Prozesses im interaktiven Betrieb

First Come First Serve (FCFS)

- Entspricht einer FIFO-Warteschlange – ankommender Prozess wird hinten eingereiht

  - Prozesse werden hinten eingereiht
  - Prozesse werden vorne entnommen

- nicht verdrängend

⇒ Nenne Vor- und Nachteile!

Shortest Job First (SJF)

- Hierbei wird die Warteschlange nach der Dauer der Prozesse, beginnend mit dem Kürzesten, geordnet. Kommt ein neuer Prozess an, wird er hinter den Prozessen mit kürzerer oder gleicher Dauer und vor den Prozessen mit längerer Dauer eingereiht. Der erste Prozess in der Schlange wird vollständig ausgeführt.

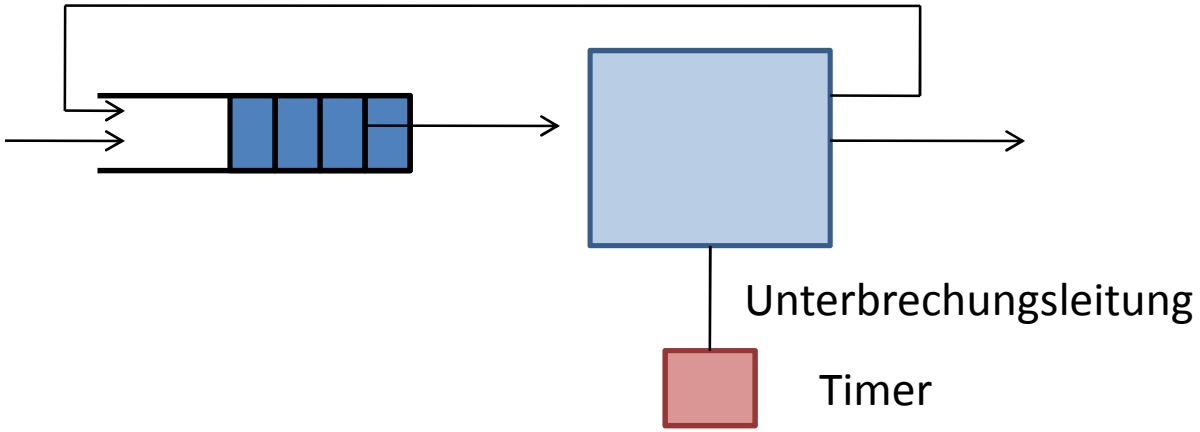
- nicht verdrängend

- optimiert mittlere Wartezeit

⟹ Nenne Vor- und Nachteile!

Round Robin

- Hier erhalten die Prozesse jeweils ein Zeitfenster, in denen sie ausgeführt werden. Danach wird der Prozess abgespeichert und erneut an das Ende der Schlange eingereiht. Ein vorheriges Sortieren entfällt. Sollte ein Prozess innerhalb oder vor dem Ende seines Fensters beendet sein, schließt der nächste Prozess sofort an.



- verdrängend

➡ Nenne Vor- und Nachteile!

Auswahlstrategien - Aufgabe

Gegeben seien folgende Prozesse:

Prozess	**Dauer**
A	9
B	7
C	1
D	5
E	4

Alle Prozesse seien ablaufbereit.
Die Reihenfolge der Ankunft im System entspreche der Reihenfolge in der Tabelle.

Plane die Prozesse nach folgenden Strategien ein. Stelle auf einer Zeitachse dar, wann welcher Prozess läuft!

FCFS

AAAAAAAAAA	BBBBBBB	C	DDDDD	EEEE

SJF

C	EEEE	DDDDDD	BBBBBBB	AAAAAAAAAA

Round Robin
(time slice = 2)

AA	BB	C	DD	EE	AA	BB	DD	EE	AA	BB	D	AA	B	A

Auswahlstrategien - Aufgabe

Berechne für jede Strategie durchschnittliche Wartezeit & Verweilzeit!

Prozess	Dauer
A	9
B	7
C	1
D	5
E	4

FCFS

AAAAAAAAAA	BBBBBBB	C	DDDDD	EEEE

SJF

C	EEEE	DDDDDD	BBBBBBB	AAAAAAAAAA

Round Robin
(time slice = 2)

AA	BB	C	DD	EE	AA	BB	DD	EE	AA	BB	D	AA	B	A

Durchschn. Wartezeit:

FCFS : (0+9+16+17+22) : 5 = 12,8

SJF : (17+10+0+5+1) : 5 = 6,6

RR : (17+18+4+17+13) : 5 = 13,8

Durchschn. Verweilzeit:

FCFS : (9+16+17+22+26) : 5 = 18

SJF : (26+17+1+5+10) : 5 = 11,8

RR : (26+25+5+22+17) : 5 = 19

OOP – Objektorientierte Programmierung

Zur Veranschaulichung der Begriffe "Klasse", "Objekt", "Konstruktor", …
übertragen wir diese Begriffe auf den Alltag (Autokauf):

Der Kunde kann sich ein Modell aussuchen.
(Programmierer) (Klasse)

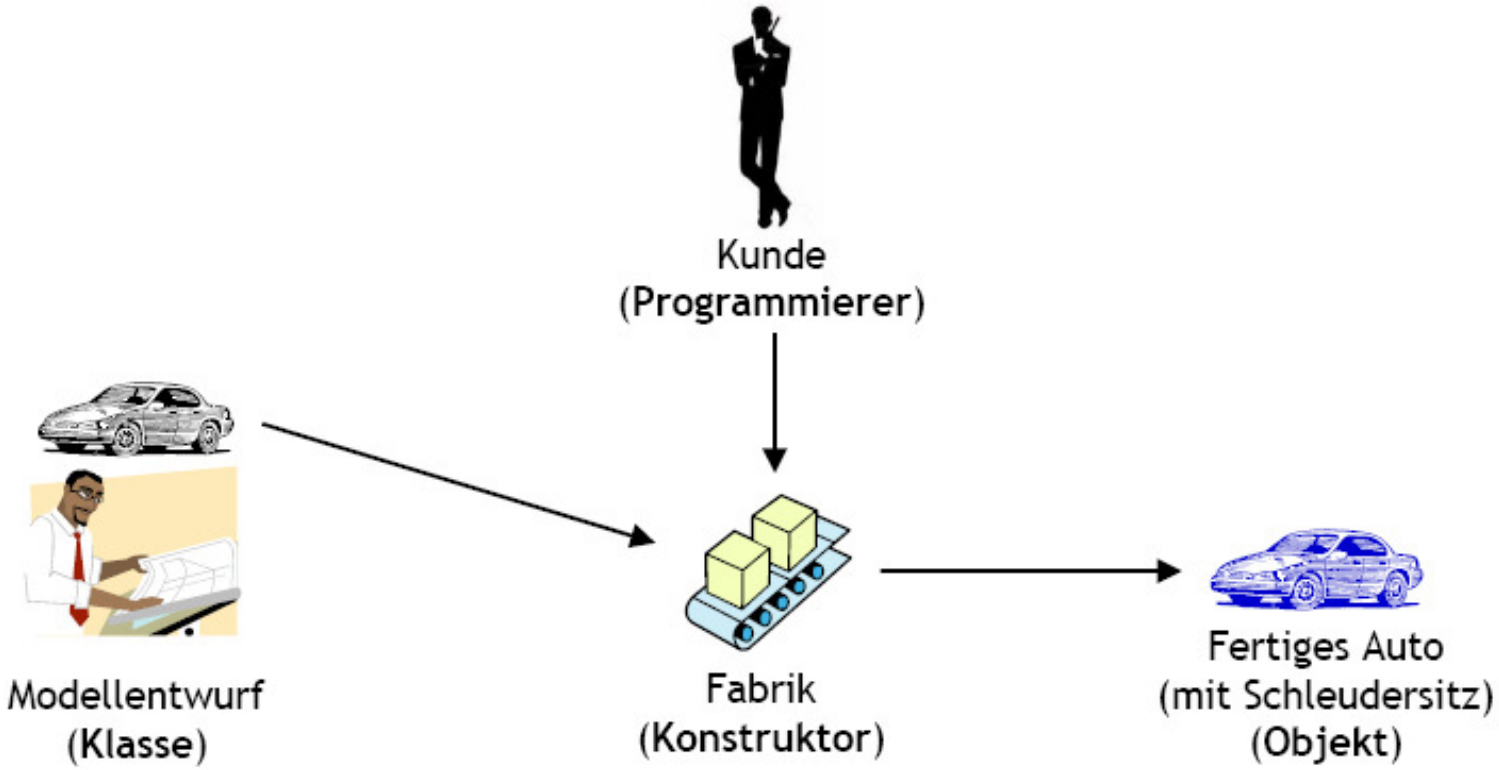
Eigenschaften des Modells können verändert werden.
(Attribute)

Das Auto wird in der Fabrik nach den Kundenwünschen produziert.
(Konstruktor)

Das fertige Auto kann dann fahren, bremsen, blinken, ….
(Objekt) (Methoden der Klasse bzw. Objekt)

OOP – Objektorientierte Programmierung

Der Kunde teilt seine Wünsche der Fabrik mit. In dieser wird auf Grundlage des allgemeinen Entwurf das Auto gefertigt



OOP – Objektorientierte Programmierung

In Java kann das dann so aussehen:

```
class Car {
        String color;
        int hp;
        String specials;
        int drivenDistance = 0;
        public void drive(int distance) {
                this.drivenDistance = this.drivenDistance + distance;
        }
        public Car(String color, int hp, String specials)  {
                this.color = color;
                this.hp = hp;
                this.specials = specials;
        }
}
```

Konstruktor
Erkennbar daran, dass die Methode keinen Typ (int, void, …) besitzt. Methodenname entspricht Klassenname

Unser neues Auto können wir dann z.B. folgendermaßen erzeugen:

```
Car myBlueCar = new Car("blue", 450, "Schleudersitz");
```

Die Methoden können jetzt einfach aufgerufen werden:

```
myBlueCar .drive(120);
```

OOP Aufgabe

Erstellen Sie die Klasse Computer. Die Klasse soll Attribute besitzen, die

- den Namen des Rechners
- die Prozessorfrequenz (in GHz)
- die Größe der Festplatte (in GB)
- ob der Rechner ein Notebook ist

beschreiben.

Lösung:

```
class Computer {
    private String name;
    private float frequency;
    private int hdSize;
    private boolean isNotebook;
}
```

OOP Aufgabe (2)

Erstelle einen Konstruktor, der den Namen fest mit „Dell“ initialisiert und die restlichen Attribute als Parameter nimmt und zuweist.

Lösung:

```
class Computer {
    private String name;
    private float frequency;
    private int hdSize;
    private boolean isNotebook;

    public Computer(float frequency, int hdSize, boolean isNotebook) {
        this.name = “Dell“;
        this.frequency = frequency;
        this.hdSize = hdSize;
        this.isNotebook = isNotebook;
    }
```

OOP Aufgabe (3)

Erstelle für das Attribut hdSize eine Methode zum Setzen und Auslesen. Lasse beim Setzen nur Werte zwischen 1 und 1000GB zu!

Lösung:

```
int getHdSize() {
        return hdSize;
}

void setHdSize(int size) {
        if (size >= 1 && size <= 1000)
                this.hdSize = size;
}
```

Instanziiere einen Computer mit 500GB Festplatte, 2.2 GHZ und als ein Notebook!

Lösung:

```
public class ComputerTest {
        public static void main(String args[]) {
                Computer myCom = new Computer(2.2f, 500, true);
        }
}
```

Attribut- und Objektgleichheit

- Folgende zwei Arten von Vergleichen lassen sich bei Objekten unterscheiden

1) Vergleich auf der Ebene der Objektvariablen:

String x, y; if (x == y) ...

Die Objektvariablen sind dann gleich, wenn sie die selbe Speicherstelle referenzieren

2) Vergleich auf der Ebene der Attribute:

String x, y; if (x.equals(y)) ...

Die Objektvariablen sind dann gleich, wenn sie den gleichen Wert enthalten

Aufgabe Gleichheit

Welche Ausgabe liefert das folgende Programmstück?

```
String  s = new String("Hallo");
String  t = new String("Hallo");
Out.println(s == t);              // Ausgabe?
Out.println(s.equals(t));         // Ausgabe?
s = t;
Out.println(s == t);              // Ausgabe?
Out.println(s.equals(t));         // Ausgabe?
```

Lösung:
false
true
true
true

Static

• Mit static deklarierte Methoden und Variablen gehören der Klasse und nicht einem bestimmten Objekt.

• Das hat den Vorteil, dass alle Objekte einer Klasse *dieselben static-Elemente benutzen*!

Die Objekte sind also nicht nur mit denselben Elementen "ausgestattet" sondern benutzen dieselben Elemente.

Static-Elemente gibt es also exakt einmal!

```
class MyStatic {
    static int counter = 0;
    int value = 2;
}

public class statictest {
    public static void main(String[] args) {
        Out.println(MyStatic.counter);
        MyStatic.counter = 1;
        MyStatic c1 = new MyStatic();
        Out.println(c1.counter);
        Out.println(++c1.value);
        MyStatic c2 = new MyStatic();
        Out.println(c2.value);
    }
}
```

Ausgabe:

```
0
1
3
2
```

Das wars für heute…

Fragen ???

Viel Spaß mit dem Übungsblatt!