GYMNASIUM MIT SCHÜLERHEIM PEGNITZ
Wilhelm-von-Humboldt-Str. 7
91257 Pegnitz
Tel. 09241/48333 Fax. 09241/2564
sek@gympeg.de www.gympeg.de

Grundwissen Informatik JS 1x — 10. September 2019

Grundlagen der JS 10

1. a) Gib einige grundlegende Datentypen von Java und ihre Bedeutung an.

   b) Welche Syntaxregeln müssen bei Datentypen eingehalten werden?

   c) Warum sind Datentypen in Programmiersprachen nötig?

Lösung

a)

Typ	Beschreibung	Literale
int	ganze Zahl	Bereich ca. ±2 Mrd
boolean	boolescher Wert	true, false
double	Gleitkommazahl	ca. $-10^{308} \ldots 10^{308}$
char	Zeichen	'a','b','c',')'
String	Zeichenkette	"Hallo Welt"
...		

b)
- primitive Datentypen werden klein geschrieben, Objekttypen (z.B. `String`) groß.
- Attributwerte vom Datentyp `char` / `String` müssen in einfache / doppelte Hochkommata einschlossen werden.

c) Abhängig vom Datentyp einer Variable muss entsprechend Platz im Datenspeicher reserviert werden.

2. Gib mit Hilfe eines Beispiels an, wie Arrays in Java implementiert werden (zwei Möglichkeiten).

Lösung

- Langform bei unbekannter oder wählbarer Länge

```
private double[] zeiten;
zeiten = new double[8];
zeiten[0] = 13.1;
...
```

- Kurzform bei bekannten Daten und bekannter Länge

```
private double[] zeiten = {10.1,
    11.1, 12.3, 12.4, 12.6, 13.0,
    13.5, 15.7};
```

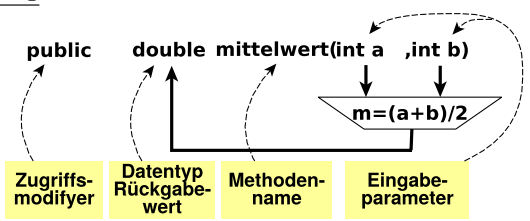
3. a) Erkläre an einem Beispiel, welchen formalen Aufbau eine Methodensignatur in Java hat.

   b) Was versteht man unter einem Konstruktor, wie ist er aufgebaut, welche Aufgaben hat er?

   c) Was versteht man unter Datenkapselung, wozu ist sie gut?

Lösung

a)



b) Der Konstruktor ist eine **Methode zur Erzeugung eines Objektes**. Er trägt den gleichen Namen wie die Klasse, hat aber keine Rückgabewerte. Er wird mit dem Befehl new(< P−Liste >) aufgerufen. Dabei werden **alle Attribute initialisiert**.

c) Um Daten vor unerlaubten Zugriffen zu schützen werden Attribute stets `private` deklariert, der Zugriff darauf erfolgt ausschließlich über Methoden.

4. a) Was versteht man unter lokalen Variablen, was unterscheidet sie von Attributen?

   b) Welchen Vorteil haben lokale Variablen gegenüber den global definierten Attributen?

Lösung Lokale Variablen...

a) werden in einzelnen Methoden ohne Modifizierer deklariert und belegen nur während der Ausführungszeit einer der Methode Speicherplatz.

b) dürfen in verschiedenen Methoden den gleichen Namen tragen, haben geringeren Speicherbedarf und erzeugen übersichtlicheren Programmcode.

```
1 public class Kugel{
2     private double radius;
3     public double oberflaecheGeben(){
4         double oberflaeche;
5         oberflaeche = 4/3*3.1414*
              radius*radius*radius;
6         return oberflaeche;
7     }
8 }
```

5. Wie werden einfache Programmverzweigungen in Java implementiert? Welche Schreibkonventionen werden eingehalten?

Lösung

```
if <logischer Ausdruck> {
   Programmcode fuer
    <logischer Ausdruck> == true
}
else{
   Programmcode fuer
         <logischer Ausdruck> == false
}
```

- Der `else`-Block kann weggelassen werden.
- Der Programmcode für den `if`- und den `else` - Teil wird jeweils eingerückt, bei verschachtelten Verzweigungen auch mehrfach.
- Das zu einem `if` gehörende `else` beginnt in der gleichen Spalte

6. a) Was versteht man unter logischen Ausdrücken? Beispiele!

   b) Welche Vergleichsoperatoren gibt es bei logischen Ausdrücken für einfache Datentypen und Strings?

   c) Mit welchen logischen Grundoperationen könnne logische Ausdrücke zusammengesetzt werden?

Lösung

a) Logische Ausdrücke sind Ausdrücke und Methoden, deren Ergebnis vom Datentyp `boolean` ist, z.B. `x==7`; `x<=6`; `true`; `roboterhandGeschlossen()`

b)

Name	Term
Gleichheit	A==B
Ungleichheit	A!=B
größer/kleiner	A>B, A<B
größer/kleiner gleich	A>=B, A<=B

c)

Name	Term	wahr wenn
Und	L && M	L und M wahr sind
Oder	L || M	L oder M oder beide wahr
Nicht	!L	L falsch ist.

7. a) Wozu sind `switch-case`-Anweisungen nützlich?

   b) Erkläre an einem aussagekräftigem Beispiel, wie eine `switch-case`-Anweisung aufgebaut ist und erläutere ihre Bestandteile

Lösung

a) `switch-case`-Anweisungen werden beim Fallunterscheidungen mit vielen unterschiedlichen Fällen verwendet.

b) Beispiel: Noten in ein Prädikat umwandeln

```
switch ( note ) {
    case 1: nText = "sehr gut"; break;
    ... //andere Faelle hier ausgelassen
    case 6: nText = "ungenuegend"; break;
    default: nText = "Fehler"; break;
}
```

- Es sind beliebig viele `case`-Klauseln erlaubt
- `break` sorgt dafür, dass nicht(!) die nächste `case`-Klausel bearbeitet, sondern ans Ende der `switch`-Anweisung gesprungen wird.
- Die `default`-Klausel ist optional und sammelt alle Fälle, die vorher nicht aufgeführt sind.

8. a) Gib in Java ein aussagekräftiges Beispiel für eine Wiederholung mit fester Anzahl an.

   b) Gib in Java ein aussagekräftiges Beispiel für eine bedingte Wiederholung an.

Lösung

a)
```
for ( int i=0 ; i<zeiten.length ; i
    =i+1 ){
        System.out.println("Platz "+
            (i+1) +": "+ zeiten[i]
            +"s");
}
```

b)
```
// 10 Kugel transportieren
i=1;
while (i<11) {
  System.out.println(i+" hoch 2 =
      "+(i*i));
  i=i+1;
}
```

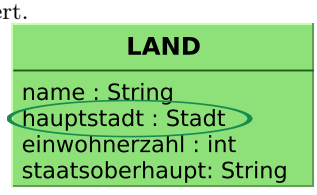
9. Erkläre an einem Beispiel, was man unter einer Aggregation und unter einem Referenzattribut versteht? Welchen Vorteil hat die Verwendung von Referenzattributen?

Lösung

Die Beziehung der Klassen `STADT` und `LAND` ist eine Aggregation, weil `hauptstadt` ein Attribut vom Datentyp `STADT` ist.

Dabei werden aber nicht die Daten des Attributes `hauptstadt` in dem Speicherbereich des Landes abgelegt. **Es wird nur eine Referenzadresse angegeben**, unter der die Daten zu finden sind.

Wenn die Stadtdaten aktualisiert wurden, sind damit die Daten der `Hauptstadt` ebenfalls aktualisiert.

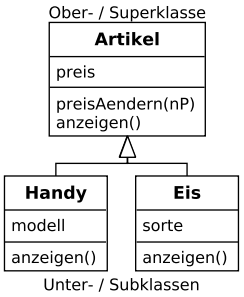


Zelle	Inhalt
...	
10000	name
10001	„Deutschland“
10002	hauptstadt
10003	10100
10004	einwohnerzahl
10005	80000000
10006	staatsoberhaupt
10007	„Merkel“
...	
10100	name
10101	„Berlin“
10102	einwohnerzahl
10103	4000000
10104	buergermeister
10105	„Wowereit“

10. Erkläre die Grundprinzipien der Vererbung?

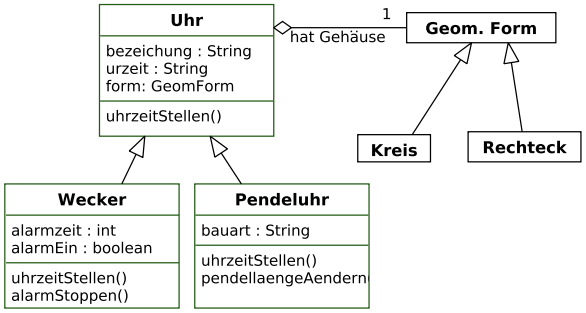
Lösung

- Die Subklassen besitzen (erben) alle Attribute und Methoden der Oberklasse. Die Methoden können aber in den Subklassen überschrieben (neu implementieren) und auf deren Bedürfnisse angepasst werden.
- Will man aus der Unterklasse den Konstruktor der Oberklasse aufrufen, geschieht das mit dem Befehl `super(<P-Liste>)`.
- Will man aus der Unterklasse eine anderen Methode (z.B. `anzeigen()`) der Oberklasse aufrufen, geschieht das mittels `super.anzeigen()`.



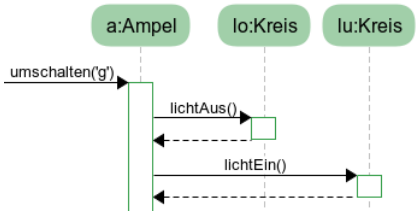
11. Erkläre an einem Beispiel, wie ein Klassendiagramm aufgebaut ist und wozu es nützlich ist. Gehe auch auf die Unterschiede zwischen Aggregation und Vererbung ein.

Lösung Das Klassendiagramm stellt die Beziehungen (Aggregation, Vererbung) zwischen den Klassen dar. Die Aggregation modelliert dabei eine `hat`-Beziehung (Eine Uhr **hat** einen Antrieb) und wird durch eine Raute samt Kardinalität dargestellt, die Vererbung modelliert eine `ist`-Beziehung (Ein Wecker **ist** eine Uhr) und wird durch einen Dreieckspfeil dargestellt.



12. Erkläre an einem Beispiel, wie ein Sequenzdiagramm aufgebaut ist und wozu es nützlich ist.

Lösung Im Sequenzdiagramm verlaufen vertikal die Lebenslinien der Objekte. Horizontal werden die ausgetauschten Anfangsbotschaften (Methodenaufrufe) und Antwortbotschaften (Rückgabewerte) dargestellt. Aktivitätsbalken auf den Lebenslinien zeigen, wenn eine Methode gerade ausgeführt wird.



13. Welchen Sinn hat beim Organisieren von Daten die „Trennung von Inhalt und Struktur“ und wie kann man sie programmiertechnisch realisieren?

Lösung Durch die Trennung können die Daten und die Organisationsstruktur (z.B. Liste oder Baum) unabhängig voneinander durch neue Formen ersetzt werden. Dazu muss man aber

- eine abstrakte Schnittstelle `Datenelement` deklarieren,
- konkrete Datenklassen als Unterklassen von `Datenelement` implementieren und die
- Datenverwaltung in einer weiteren Klasse für die abstrakten Datenelemente implementieren.

14. a) Beschreibe allgemein, also nicht nur auf die Informatik beschränkt, was man unter Rekursion versteht.

b) Gib das Grundprinzip rekursiver Methodenaufrufe in Pseudocode an.

Lösung

a) Von Rekursion spricht man, wenn bei der Lösung eines Problems bei mindestens einem der Teilprobleme die gleiche Strategie angewendet werden muss wie bei dem Gesamtproblem, nur in verkleinerter oder irgendwie anders reduzierter Form. Durch eine Abbruchbedingung terminiert die Rekursion.

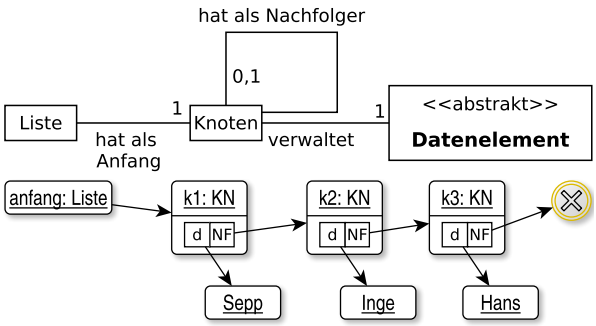
b)

```
rekMethode
  wenn Abbruchbedingung erfüllt
    gib ergebnis aus
  sonst
    bestimme Teilergebnis und wende
    rekMethode auf den Nachfolger an
```

Listing 1: Pseudocode rekursiver Methode

15. Beschreibe das Klassendiagramm und ein mögliches (verkürztes) Objektdiagramm einer einfach verketteten Liste, die rekursiv deklariert ist. Woran erkennt man, dass die Datenstruktur rekursiv ist? Woran erkennt man, dass Struktur und Inhalt getrennt sind?

Lösung



Die Datenstruktur ist rekursiv, weil die Klasse `Knoten` sich selbst als Attribut enthalten kann. Struktur und Inhalt sind getrennt, weil die Daten in die Klasse `Datenelement` ausgelagert sind und diese als abstrakte Klasse deklariert ist.

16. Gib die Java-Implementation einer typische rekursive Methode der Klasse `Knoten` bei einer Liste ohne Kompositum an.

Lösung

```
int AnzahlGeben(){
  if (nachfolger == null)
    return 1;
  else
    return nachfolger.AnzahlGeben
        () + 1;
}
```

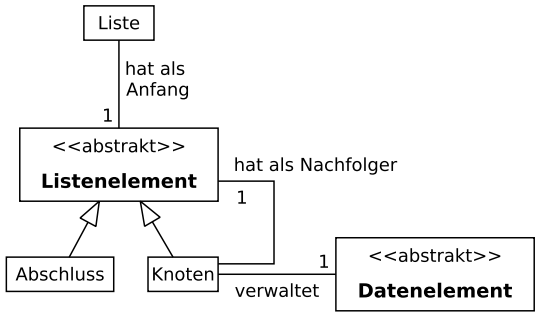
Listing 2: Knoten.anzahlGeben()

17. a) Beschreibe das Klassendiagramm einer einfach verketteten Liste als Kompositum.

b) Welche Vorteile hat das Entwurfsmuster `Kompositum` bei rekursiven Datenstrukturen.

Lösung

a)



b) Bei rekursiven Methoden werden in der Klasse `Knoten` keine Sicherheitsabfragen an die leere Liste nötig, weil sie in der Klasse `Abschluss` gesondert implementiert werden.

18. Gib die Java-Implementation einer typische rekursive Methode bei einer Liste mit Kompositum an in allen Klassen an.

Lösung

```
private LISTENELEMENT anfang;
return anfang.AnzahlAbHierGeben();
```

Listing 3: Liste.anzahlGeben()

```
abstract int AnzahlAbHierGeben();
```

Listing 4: Listenelement.anzahlGeben()

```
return nachfolger.
    AnzahlAbHierGeben() + 1;
```

Listing 5: Knoten.anzahlAbHierGeben()

```
return 0;
```

Listing 6: Abschluss.anzahlAbHierGeben()

19. a) Was versteht man in der Informatik unter einem Baum?

    b) Wie viele Elemente haben bei einem Binärbaum in einer Ebene, bzw. im ganzen Baum Platz?

Lösung

a) Bestandteile der Datenstruktur Baum sind **Knoten** und **Kanten**. Der erste Knoten, der keinen Vorgänger hat, heißt **Wurzel**. Bei Knoten unterscheidet man zwischen **Blättern** (Knoten ohne Nachfolger) und **inneren Knoten**. Die **Tiefe** eines Knotens ist die Anzahl der Kanten des Weges von der Wurzel bis hin zu diesem Knoten. Die **Tiefe des Baums** ist gleich der maximalen Knotentiefe.

b) In die n-te Ebene eines Binärbaumes passen maximal 2^n Elemente, ein Binärbaum der Tiefe n kann maximal $2^{n+1} - 1$ aufnehmen.

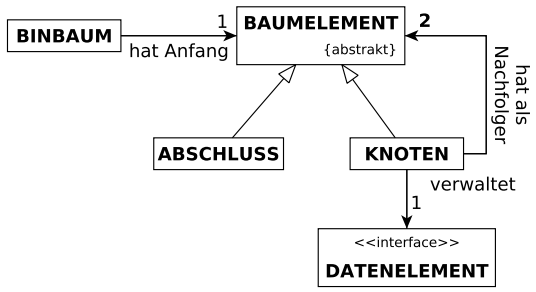
20. a) Was versteht man in der Informatik unter geordneten Binärbaum?

    b) Beschreibe das Klassendiagramm eines geordneten Binärbaums als Kompositum.

Lösung

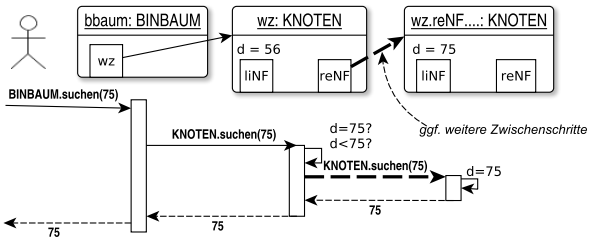
a) Ein **geordneter Binärbaum** ist ein Baum der Ordnung 2 bei dem für jeden Knoten K die Werte in allen Knoten des linken Teilbaums kleiner sind als der Wert dieses Knotens K, die Werte in allen Knoten des rechten Teilbaums sind größer (oder gleich) als der Wert dieses Knotens K.

b)



21. Beschreibe mit Hilfe eines Sequenzdigramms an einem Beispiel, wie bei einem Binärbaum die Suche eines Datenelements abgearbeitet wird.

Lösung `Knoten` sein mit Kompositum implementiert, die Zahl 75 vorhanden.
`BINBAUM.suchen(75)` wird aufgerufen und `KNOTEN.suchen` auf `wurzel` angewendet. Darin wird die rekursive Methode so lang an die Nachfolger weitergegeben, bis die Abbruchbedingung erreicht ist und somit der Rückgabewert über die darüber liegenden Knoten zurückgegeben wird. In der Klasse `Abschluss` wird `null` zurückgegeben.



22. Gib die Implementierung eines Baumdurchlaufes deiner Wahl im Entwurfsmuster `Kompositum` an.

Lösung

```
wurzel. PreOrder ();
```

Listing 7: Baum.PreOrder()

```
abstract void PreOrder ();
```

Listing 8: Baumelement.PreOrder()

```
daten. Ausgeben ();
linkerNachfolger. PreOrder ();
rechterNachfolger. PreOrder ();
```

Listing 9: Knoten.PreOrder()

```
{}
```

Listing 10: Abschluss.PreOrder()

23. a) Erkläre, was man unter einem Graph versteht und welche Typen von Graphen man unterscheidet.

    b) Welche typischen Fragen werden bei der Datenstruktur Graph untersucht?

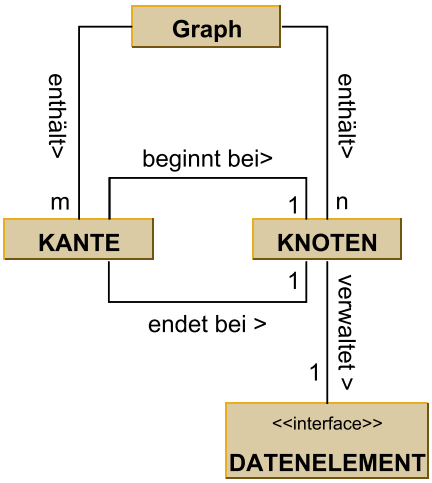
Lösung

a) Ein Graph besteht aus einer endlichen Menge von Knoten und Kanten. Jede Kante verbindet genau zwei Knoten. Die Kanten können
- gerichtet oder ungerichtet sein,
- eine Bewertung haben oder nicht.

b)
- Wie kann eine Netzstruktur modelliert werden?
- Wie ist es möglich, alle Knoten eines Graphen systematisch zu besuchen?
- Welche Wege gibt es von einem vorgegebenen Startknoten zu einem Zielknoten?
- Wie kann ein bestimmter Knoten im Netz gefunden werden?

24. Beschreibe das Klassendiagramm der Datenstruktur `Graph`.

Lösung



25. a) Wie werden die Kanten eines Graphen implementiert?

    b) Welche Spezialfälle gibt es?

Lösung

a) Die Kanten werden in der Adjazenzmatrix als Attribut der Klasse `Graph` gespeichert. Es gibt also keine eigene Klasse `Kante`.

Die Adjazenzmatrix ist eine quadratische Tabelle, deren Anzahl an Spalten bzw. Zeilen mit der Anzahl der Knoten des Graphen übereinstimmt. Die Einträge in den Zellen der Matrix geben wieder, ob zwischen zwei Knoten eine Kante existiert und welche Bewertung diese gegebenenfalls hat.
- Für unbewertete Graphen reichen boolsche Werte in der Adjazenzmatrix.
- Für einen ungerichteten Graphen ist die Adjazenzmatrix symmetrisch bezüglich der Diagonalen von links oben nach rechts unten.

26. Beschreibe zwei Verfahren mit denen man alle Knoten eines Graphen systematisch besuchen kann in Pseudocode. Beispiel!

Lösung

```
markiere einen Knoten v als besucht
füge v in eine (zunächst leere) Liste D
    ein
solange D nicht leer ist
   entnimm einen Knoten w aus D
   markiere alle noch nicht besuchten
       Nachbaren von w als besucht und
       füge sie in die Liste D ein
```

Breitensuche: Es wird das zuerst abgelegte Element als erstes abgeholt (FIFO-Prinzip, Schlange)

Tiefensuche: Es wird das zuerst abgelegte Element als letztes abgeholt (FILO-Prinzip, Stapel)

Beispiel ...

27. a) Warum müssen größere Projekte geplant werden?

    b) Welche zwei grundlegenden Projektplanungsstrategien unterscheidet man? Wann setzt man sie sinnvollerweise ein?

Lösung

a)
- Die **Anforderungen** des Auftraggebers müssen erfüllt werden,
- die **Ressourcen** (Geld, Zeit, Material . . .) müssen eingeteilt werden,
- es sind viele **Personen** beteiligt, die nicht ständig zur Verfügung stehen,
- das Projekt dauert Jahre und muss **gewartet** werden.

b) Man unterscheidet **statische und dynamsiche Strategien**. Statische Strategien werden eingesetzt, wenn die Anforderungen bekannt und stabil sind, sonst verwendet man dynamische Strategien.

28. a) Erkläre den grundsätzlichen Aufbau einer statischen Projektplanungsstrategie.

    b) Welche Nachteile haben statische Projektplanungsstrategien?

Lösung

a) Der Auftraggeber stellt die Anforderungen in einem **Lastenheft** dar. Der Auftragnehmer beschreibt im **Pflichtenheft**, wie er die Anforderungen erfüllen will. Akzeptiert der Auftraggeber das Pflichtenheft, wir das Projekt nach einem **standardisierten Ablauf** umgesetzt, und zwar durch Zerteilen des Projektes in **Teilprojekte** und Festlegung von **Meilenstein**en.

b)
- Das Ergebnis entspricht möglicherweise nicht den tatsächlichen Erfordernissen weil Anforderungen übersehen wurden oder sich geändert haben.
- Möglicherweise unnötiger und sehr großer Planungsaufwand.

29. a) Erkläre den grundsätzlichen Aufbau einer dynamischen Projektplanungsstrategie.

    b) Welche Nachteile haben dynamische Projektplanungsstrategien?

Lösung

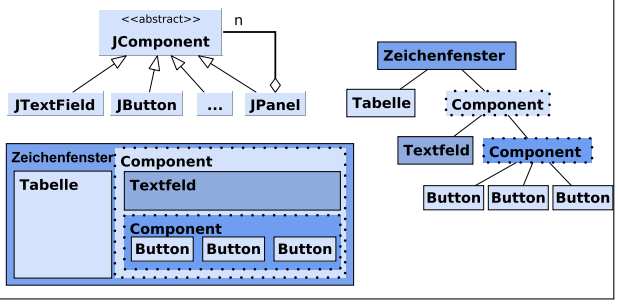
a) Die reine Entwurfsphase wir auf ein Mindestmaß reduziert und es wird **so früh wie möglich ein verwendbares Produkt zur Verfügung gestellt**, das dann in kurzen Abständen dem Kunden zur gemeinsamen Abstimmung vorgelegt werden kann. Auf diese Weise soll es jederzeit möglich sein, **flexibel auf Kundenwünsche einzugehen**, um so die Kundenzufriedenheit insgesamt zu erhöhen.

b) Ein planvolles Handeln findet kaum statt, es werden nur halbwegs taugliche Lösungen geliefert (Gefahr der „Flickschusterei“).

30. In welchen Fällen kann man das Entwurfsmuster `Kompositum` verwenden? Nenne einige Einsatzgebiete die Ober- und die Unterklassen.

Lösung Es wird verwendet, wenn Gruppen von Objekten das gleiche Verhalten zeigen sollen wie einzelne Objekte. Beispiele: Implementation von

- Listen- und Baumelementen (`Knoten`, `Abschluss`)
- Zeichenelementen (`Kreis`, `Rechteck`, `Gruppe`)
- Grafikkomponenten (`JTextField`, `JButton`, `JPanel` etc.)



31. a) Erkläre das Grundprinzip des Beobachtungsmusters.

b) Gib ein Beispiel, das nichts mit Informatik zu tun hat, das aber den Aufbau des Beobachtungsmusters erläutert.

c) Gib ein Beispiel aus der Informatik, das das Beobachtungsmuster verwendet.

Lösung

a) Der Informant bietet ein Interface an, das Interessenten nutzen können. Dazu müssen sich die **Beobachter beim Informanten registrieren**.

b) Der Zeitungsverleger tritt als Informant auf, der die Information (Zeitung) für alle in gleicher Weise zur Verfügung stellt. Der Abonnent ist bei Verlag registriert.

c) EventHandling in Java:

Informant: z.B. Mousebutton, Beobachter: Interface-Klasse `ActionListener`. Der Listener muss die Methode `actionPerformed(ActionEvent)` implementieren, in der festgelegt wird, wie auf den Klick reagiert werden soll.

Findet ein Mausklick beim Button statt, wird der registrierte Listener benachrichtigt und die Methode `actionPerformed` ausgeführt.

32. a) Erkläre das Grundprinzip des MVC-Musters

b) Wo wird es eingesetzt?

c) Welche Vorteile bietet es?

Lösung

a) Es basiert auf der klaren Trennung von funktionalem Kern (model), der Darstellung der Daten einschließlich Ein- und Ausgabefelder (view) sowie Entgegennahme und Verarbeitung der Benutzereingaben (controller). View und controller bilden das Userinterface, view und model werden oft nach dem Beobachtungsmuster entwickelt wobei der View als Beobachter agiert.

b) Graphical-User-Interface (GUI)-Programmierung, Verschiedene Endgeräte (Computer, Handy, ipad) nutzen gleiches Programm (E-Mail), Nutzung verschiedener Web-Browser für http.

c) Spätere Änderungen oder Erweiterungen werden erleichtert, die Wiederverwendbarkeit der einzelnen Komponenten wird ermöglicht.

33. Gib ein einfaches Beispiel für eine Grammatik an und erläutere daran die dazugehörigen Fachbegriffe.

Lösung

$$\begin{array}{lll} R_1: S & \rightarrow & aB \\ R_2: B & \rightarrow & \epsilon|bB \end{array}$$

Eine Grammatik (V, Σ, P, S) besteht aus ...

- Einer Menge V von Nichtterminalsymbolen (Variablen (S,B)),
- Einer Menge Σ von Terminalsymbolen (Alphabet (a,b)).
- Einer Menge P von Produktionen (auch Regeln genannt (R_1, R_2)).
- Einem Startsymbol S aus der Menge der Nichtterminalsymbole.
- Σ und V dürfen kein gemeinsames Element besitzen ($\Sigma \cap V = \{\}$).

34. Welcher Zusammenhang besteht zwischen einer Grammatik und einer Formalen Sprache?

Lösung Ausgehend vom Startsymbol werden durch Anwendung der Produktionsregeln gültige Zeichenketten produziert. Die Menge aller gültigen Zeichenketten bildet die Formale Sprache. Dabei ist jeder Grammatik eine eindeutige Formale Sprache zugeordnet aber nicht umgekehrt. Z.B. erzeugen

$$\begin{array}{lll} R_1: S & \rightarrow & aB \\ R_2: B & \rightarrow & \epsilon|bB \end{array} \quad \text{und} \quad \begin{array}{lll} R_1: S & \rightarrow & a|aB \\ R_2: B & \rightarrow & b|bB \end{array}$$

die Worte $a\{b\}$.

Im allgemeinen produziert eine Grammatik auch bedeutungslose Worte, also solche, die zwar syntaktisch korrekt aber semantisch falsch sind. Z.B. das Datum 31.Februar 2012.

35. a) Gib die wesentlichen Symbole der EBNF bzw. eines regulären Ausdrucks an.

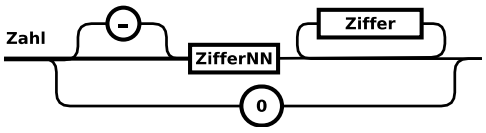
b) Gib ein aussagekräfiges Beispiel für eine EBNF, den dazugehörigen regulären Ausdruck und das Syntaxdiagramm an.

Lösung

a)

	EBNF	reg. A
Alternative	`A|B`	`A|B`
Option	`[A]`	`A?`
Wiederholung	`{A}`	`A*`
log. Gruppierung	`(A)`	`(A)`
Terminalsymbol	`'x'`	`'x'`

b)
```
Zahl = ( ['-'] ZifferNN {Ziffer} ) | '0'
Zahl = ( '-'? ZifferNN Ziffer* ) | '0'
```

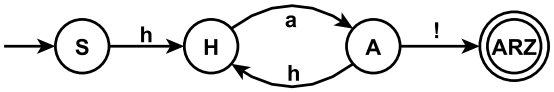


36. Was versteht man unter einem endlichen erkennenden Automat? Erkläre die Begriffe anhand eines Beispiels.

Lösung Er besteht aus ...

- einer Menge der Zustände, z.B. $Z = \{S, H, A, ARZ, Fehler\}$, einem Startsymbol: $S \in Z$ und einer Menge von Endzuständen: $F \subset Z$.
- einer Menge der Eingabezeichen (Alphabet), z.B. $T = \{h, a, !\}$
- einer Übergangsfunktion $\{\delta : Z \times T \to Z\}$

Durch Lesen eines Eingabezeichens wechselt der Automat nach den Regeln der Übergangsfunktion in den nächsten Zustand. Wenn er sich nach Abarbeiten der Zeichenfolge in einem Endzustand befindet, gilt das Wort als erkannt.



37. Wie werden erkennende Automaten in Java implementiert?

Lösung

- Es wird eine Variable `Zustand` definiert, die anfangs auf den Startzustand des Automaten gesetzt wird.
- In einer Schleife wird das zu testende Wort buchstabenweise ausgewertet. Für jeden Buchstaben `x` wird dabei eine Methode `x()` aufgerufen, die abhängig vom aktuellen Zustand den Wert der Zustandsvariable neu festlegt.
- Ist das Wort abgearbeitet, wird noch geprüft, ob sich die Zustandsvariable in einem Endzustand befindet und somit das Wort als erkannt wurde.

38. a) Nenne Möglichkeiten, wie man verschiedene Rechner miteinander verbinden kann, und gib, abgesehen von den Kosten, jeweils einen Vor- und einen Nachteil dieser Möglichkeiten an.

    b) Beschreibe die Netztopologie des Internets und erkläre, wie jeder Rechner eindeutig identifiziert wird.

Lösung

a)

Typ	Vorteil	Nachteil
Vermascht	Hohe Ausfallsicherheit	voll Vermaschung geht nur bei kleinen NW
Bus/ Stern	Ausfall eines Gerätes stört Funktion des NW nicht	völliger Ausfall bei Bus-/Serverausfall
Ring	Kollisionen sind nicht möglich	geringe Übertragungsraten

b) Das Internet ist eine Kombination verschiedener Netzwerktopologien und damit extrem ausfallsicher. Rechner werden durch 32-Bit-IP-Adressen identifiziert, die durch den DNS-Server aufgelöst wird. Router steuern den Datenaustausch zwischen verschiedenen lokalen Netzen.

39. Was versteht man in der Informatik unter einem Protokoll und wozu ist es gut? Gib zwei Beispiele für Protokolle an, die nichts mit EDV zu tun haben.

Lösung Protokolle legen die Kommunikation zwischen zwei Parteien fest. Sie definieren

- das Format und
- die Reihenfolge der auszutauschenden Nachrichten sowie die
- Handlungen, die bei der Übertragung oder dem Empfangen einer Nachricht (oder eines anderen Ereignisses) unternommen werden.

Alltagsbeispiele für Protokolle ist etwa der Gesprächsablauf zwischen Käufer und Verkäufer beim Einkaufen oder die Regeln für die Übergabe eines Paketes durch den Postboten an den Empfänger.

40. Nenne vier typische Dienste des Internets und jeweils mindestens ein dazugehöriges Protokoll. Erkläre auch wie ein Rechner diese verschiedenen Dienste unterscheiden kann.

Lösung

	Dienst	Protokoll
1	E-Mail-Versandt	smtp
2	E-Mail-Empfang	pop3
3	www	http
4	Paketvermittlung	IP

Ein Rechner kann die verschiedenen Dienste unterscheiden durch sogenannte Ports. Dabei handelt es sich um Nummern, die zusätzlich zur IP-Adresse angegeben werden. Typische Beispiele für Portnummern sind 25 (smtp) 80 (Webserver) 110 (imap, pop3).

41. Erkläre die Idee des Schichtenmodells und erläutere den Sachverhalt am Beispiel des TCP/IP-Schichtenmodells. Gib außerdem ein Beispiel eines Schichtenmodells für einen alltäglichen Vorgang, der nichts mit Computern zu tun hat.

Lösung Das Schichtenmodell strukturiert Arbeitsabläufe, die bei der Kommunikation zwischen Prozessen auftreten. Jede Schicht hat ihre spezielle Aufgabe und kommuniziert nur mit der direkt darüber- oder darunterliegenden Schicht über ein schichtenspezifisches Protokoll und festgelegte Schnittstellen.

Schicht	Aufgabe (mögl. Protokoll)
Anwendung	Zusammenarbeit mit Anwendungsprogrammen (http, ftp)
Transport	Übertragung der Daten, Portfestlegung (tcp)
Vermittlung	Weiterleitung von Paketen und Routing (IP, IPv4, IPv6)
Netzzugang	Technik der Datenübertragung

Bsp: Briefversandt ...

42. a) Erkläre was mit unter nebenläufigen Prozessen versteht und welche Vor- und welche Nachteile sie besitzen.

b) Nenne Beispiele für nebenläufige Prozesse aus dem Alltag und dem EDV-Wesen.

c) Erläutere das Synchronisationsproblem genauer.

Lösung

a) Nebenläufige Prozesse (Threads) sind unabhängige Arbeitsschritte, die gleichzeitig bearbeitet werden. Das steigert die Effizienz, erfordert aber ein Synchronisation, d.h. Abstimmung von Prozesse, die gleiche Ressourcen nutzen oder aufeinander warten müssen.

b) Hausbau: Fasade Streichen und Küche einbauen, EDV: Systemuhr und Tastaturabfragen.

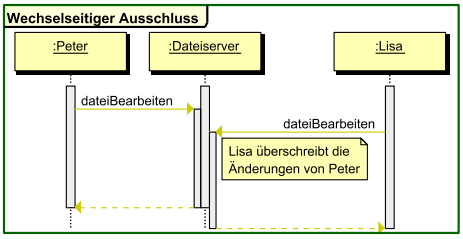
c) Verschiedene Prozesse müssen in **kritischen Abschitten** auf gemeinsame Ressourcen zugreifen. Diese Ressourcen können deshalb oft nur im **wechselseitigen Ausschluss** genutzt werden. Ist eine Reccource einem Prozess einmal zugeteilt, kann sie oft nicht mehr entzogen werden (**ununterbrechbare Ressource**).

43. a) Gib ein Beispiel für ein Synchronisationsproblem.

b) Wie kann man das Synchronisationproblem lösen?

Lösung

a)



b) Man fasst die beiden Methoden `lesen` und `schreiben` zu einem Monitor zusammen, so dass immer nur ein Objekt eine dieser Aktionen ausführen kann. Ein Monitor ist dabei eine Gruppe von Methoden, die zu einem bestimmten Zeitpunkt immer nur von einem einzigen Objekt genutzt werden darf. Die anderen Objekte müssen warten.

44. a) Gib ein Beispiel für einen Deadlock.

b) Bennene mit Fachbegriffen wie in obigem Beispiel die Verklemmung verhindert werden kann.

Lösung

a) ...

b) Einem Prozess müssen Betriebsmittel wieder **entzogen** werden dürfen oder jeder Prozess gibt zu Beginn an, welche Betriebsmittel er benötigt und bekommt sie erst, wenn sie **gleichzeitig** frei sind oder Betriebsmittel werden **exklusiv** zugeteilt.

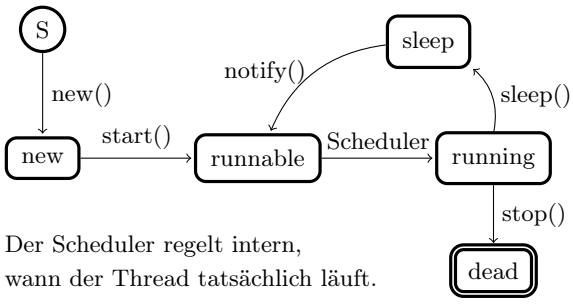
45. a) Erkläre, was man unter Threads versteht und welche Vorteile sie haben.

b) Erläutere das Zustandsübergangsdiagramm der Klasse `Thread`.

Lösung

a) Threads sind Prozesse, die vom Prozessor quasi parallel abgearbeitet werden können, sie nutzen die vorhandenen Ressourcen effizienter, z.B. durch Weiterführung der Berechnungsopperationen, während andere Threads auf Ein-/Ausgaben warten.

b)



Der Scheduler regelt intern, wann der Thread tatsächlich läuft.

46. Wie werden Threads in Java umgesetzt?

Lösung In Java werden eigene Threads als Unterklassen einer vordefinierten Klasse Thread implementiert. Der Benutzer muss die Methode `run()` der Klasse `Tread` überschreiben, auf seine Bedürfnisse anpassen und den eigenen Thread mit `start()` an den Scheduler übergeben.

```
class MeinThread extends Thread {
  //Nötige Attribute, Konstruktor, etc.

  public void run ()
        { // Arbeitsauftrag des Threads }
}

// Erzeugen und aktivieren des Threads
// meist in einer anderen Klasse
   Thread t = new MeinThread ();
   t.start ();
```

47. a) Erläutere das Erzeuger-Verbraucher-Problem.

b) Was sind die kritischen Abschnitte beim EV-Problem? Wie kann man Problem dabei verhindern?

Lösung

a) Mehrere Erzeuger können unabhängig voneinander Produkte in einen begrenzten Pufferspeicher ablegen, die bei Bedarf vom Verbraucher abgeholt werden.

b)
- Einfügen und Entnehmen von Produkten in bzw. aus dem Puffer, weil er überlaufen bzw. leer sein kann. Man verhindert Probleme, indem der Prozess ggf. wartet.
- gleichzeitige Zugreifen von Erzeuger und Verbraucher auf den Puffer. Man verhindert Probleme, indem man die Methoden `einfügen(Objekt o)` und `entnehmen()` der Klasse `Puffer` zu einem Monitor zusammen, so dass jeweils nur ein Prozess auf diesen Methoden arbeiten kann.

48. Wie ist ein von-Neumann-Rechner aufgebaut? Wie werden Programme abgearbeitet?

Lösung

Speicherwerk speichert Programme und Daten binär in gleichgroßen, fortlaufend nummerierten Zellen.

Rechenwerk führt elementare Rechenoperationen aus.

Steuerwerk steuert Ablauf und Informationsaustausch.

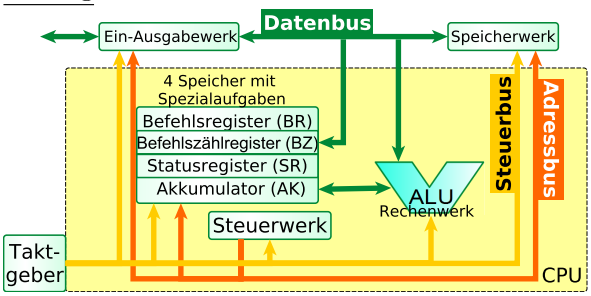
Ein-/Ausgabewerk regelt die Kommunikation mit der Außenwelt.

Bussytem zentral gesteuertes und getaktetes Übertragungssystem bestehend aus Daten-, Adress- und Steuerbus.

Aufeinander folgende Befehle (arithmetische-, logische-, Datentransportbefehle) eines Programms werden in aufeinander folgenden Speicherzellen abgelegt und nacheinander abgearbeitet. Zusätzlich existieren Sprungbefehle.

49. Beschreibe den Aufbau einer Registermaschine.

Lösung



Die CPU beihaltet die Register, die ALU und das Steuerwerk. Das BR speichert den momentan ausgeführten Befehl, der BZ enthält die Speicheradresse den Befehls, der als nächstes ausgeführt wird, der AK kann eine ganze Zahl speichern, SR setzt Flags, wenn bestimmte Ereignisse (N(egativ), Z(ero), O(verflow)) im AK eingetreten sind.

50. a) Was versteht man unter einer Assemblersprache?

b) Von welchen Parametern des Prozessors CPU sind die Geschwindigkeit und die Anzahl der Befehle einer Assembersprache festgelegt?

Lösung

a) Assemblersprachen enthalten nur sehr elementare, rechnerabhängige Befehle die direkt in Binärcode übersetzt werden. Dazu gehören Transport- und Sprungbefehle sowie logische Verknüpfungen und ein Befehl zum markieren des Programmendes

b) Die Taktfrequenz legt die Geschwindigkeit fest, mit der elementare Programmschritte abgearbeitet werden. Eine Verdopplung der Taktfrequenz führt normalerweise nicht zu einer Verdopplung der Rechengeschwindigkeit, weil dann oft Maschinenbefehle in elementarere Teilbefehle zerlegt werden. Die Bit-Breite (z.B. 64-Bit) des Prozessors bestimmt die Größe der Speicherzellen und damit die Anzahl der Befehl (max. 2^{64})

51. Beschreibe den Grundzyklus, nachdem ein Maschninenbefehl abgearbeitet wird

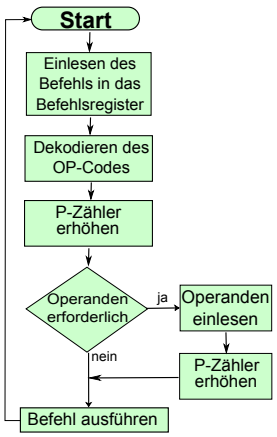
Lösung

FETCH (1. Teil) Befehl holen und Befehlszähler inkrementieren.

DECODE Befehl decodieren.

FETCH (2. Teil) Befehl vervollständigen (z.B. wenn bei der Addition mehrere Operanden zu laden sind) und Befehlszähler inkrementieren.

EXECUTE Operation im Rechenwerk ausgeführen und Befehlszähler richtig setzen (z.B. bei Sprüngen).



52. a) Wie werden allgemein in Assemblerprogrammen Schleifen und bedingte Anweisungen umgesetzt?

b) Gib ein einfaches Assemblerprogramm mit einer einseitig bedingten Anweisung an.

Lösung

a) Indem man durch unbedingte die geeignet gewählte bedingte Sprünge Programmabschnitte mehrfach durchläuft oder überspringt.

b) Sortieren der beiden Zahlen in den Zellen 20 und 21 nach der Größe.

```
1  load 20   //Inhalt Adr 20 in Acc
2  sub 21    //sub Inhalt von Adr 21
3  jmpn 10   //wenn n-Flag gehe zu 10
4  load 20   //4-9: Ringtausch
5  store 22  //Inhalt von 20 nach 22
6  load 21
7  store 20  //Inhalt von 21 nach 20
8  load 22
9  store 21  //Inhalt von 22 nach 21
10 hlt
```

53. a) Erkläre, was man unter Komplexitätsklassen versteht.

    b) Nenne typische Komplexitätklassen und dazugehörige Algorithmen / Vorgänge.

Lösung

a) Komplexitätsklassen beschreiben welcher mathematische Funktionstyp den Aufwand (z.B. für Speicherplatz oder Zeit) angibt, den ein Algorithmus in Abhängigkeit von der Anzahl der zu behandelnden Elemente bebötigt.

Vorgang	Typ	K-Klasse
Rasenmähen	linear	$\mathcal{O}(n)$
Händeschütteln	quadr.	$\mathcal{O}(n^2)$
Hanoi	exp.	$\mathcal{O}(e^n)$
Baumsuche (sort.)	log.	$\mathcal{O}(\ln n)$

54. Wie lautet (vereinfacht) das $P = NP$-Problem? Wie ist der aktuelle Stand? Nennen typische dazugehörige Probleme.

Lösung Wenn es einen Algorithmus gibt, der die Richtigkeit der Lösung eines Problems in polynomialer Zeit **überprüfen** kann (der Algorithmus gehört dann zur Komplexitätsklasse NP), gibt es dann auch einen Algorithmus (er gehört dann zur Komplexitätsklasse P), der die Lösung in polynomialer Zeit **findet**? Es wird davon ausgegangen das $P \neq NP$ ist. Das Problem ist eines der sieben Milleniumsproblemen und von enormer Wichtigkeit für die Bestimmung der Effizienz von Algorithmen, da Algorithmen der Klasse $\mathcal{O}(e^n)$ nur für kleine Anzahlen effizient sind. Der Beweis des $P = NP$-Problem steht noch aus. Typische Probleme sind das Rucksack- und das Travelling-Salesman-Problem ....

55. a) Was versteht man unter dem Kerckhoff'schen Prinzip?

    b) Nenne typische Schlüssel in modernen Kryptographie-Systemen.

    c) Wodurch wird die Geheimhaltung des Schlüssels in modernen Kryptographie-Systemen sichergestellt?

    d) Nenne Anwendungen der modernen Kryptographie.

Lösung

a) Die Sicherheit eines Kryptographiesystems darf nicht von der Geheimhaltung des Algorithmus abhängen, sondern nur auf der Geheimhaltung des Schlüssels.

b) Primzahlen und daraus abgeleitete Potenzen und Produkte mit Längen von etwa 512 bis 1024 Bit (ca. 300 Stellen).

c) Dadurch, dass das Faktorisieren von Produkten durch einen brute-force-Angriff zu der Komplexitätsklasse NP gehört und erwartungsgemäß $P \neq NP$.

d) E-Mail-Verkehr (OpenPGP, S/MIME), SSH, https, Digitale Signaturen, elektronische Zahlungssysteme, Chipkartenauthentifikation.