Informatik II

Übungsstunde 7

simon.mayer@inf.ethz.ch
Distributed Systems Group, ETH Zürich

Ablauf

- Besprechung der Vorlesung
- Uebungsbezogene Themen:
  Java Generics, Binaerbaeume
- Zeit zum Programmieren...
  und fuer noch mehr Fragen

Java Generics

http://download.oracle.com/javase/1.5.0/docs/guide/language/generics.html

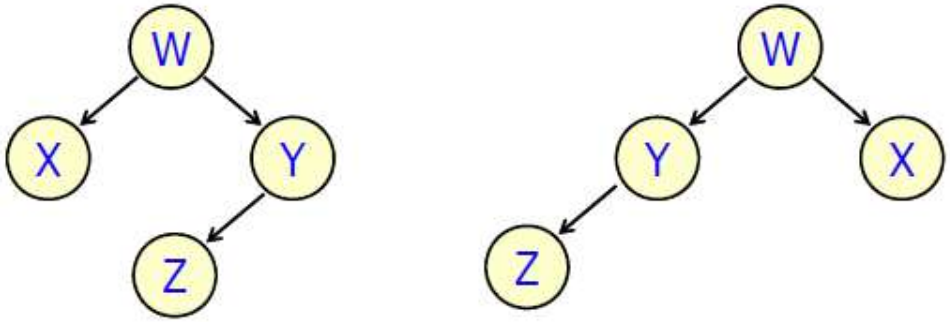
- Ich kann nun dem Compiler den Typ einer Collection mitteilen
  -> Erlaubt type checking!
  - ArrayList<T> zeigt an, was darin gespeichert ist
  - Type checking beim Hinzufuegen zur Liste
  - ArrayList<T> liefert direkt Objekte des korrekten Typs (kein casting)

- Sehen wir uns (um ein bisschen Einblick in das Zeugs zu bekommen) mal die Vererbungsverhaeltnisse an (Demo!) ☺

Mi a bináris keresőfa?

Zur Erinnerung: Binärbaum

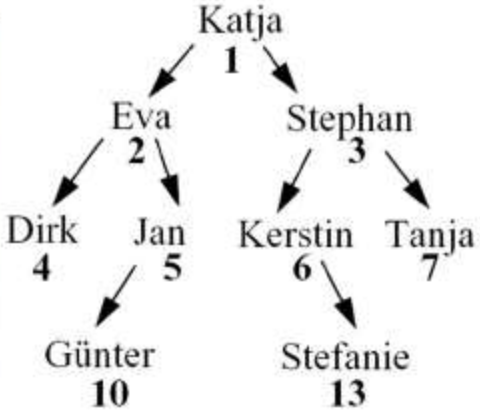
- Jeder Knoten hat *höchstens* zwei Nachfolger
- Unterscheide linken und rechten Nachfolger / Unterbaum



Binärbäume als Referenzstrukturen (2)

- Ein wenig umständlich könnte man dann so den Baum aufbauen:

```
Person Wurzel;
Wurzel = new Person();
Wurzel.name = "Katja";
Wurzel.left = new Person();
Wurzel.left.name = "Eva";
Wurzel.right = new Person();
Wurzel.right.name = "Stephan";
Wurzel.left.left = new Person();
...
```

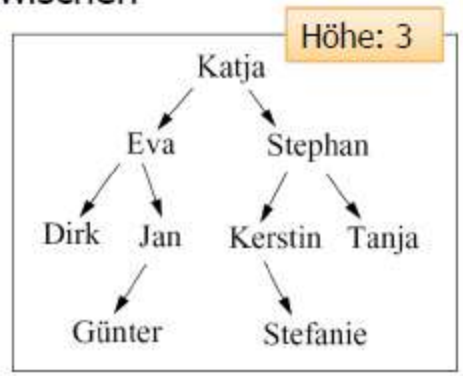


- Es können dynamisch neue Knoten hinzugefügt werden
  - Im Unterschied zur früheren Array-Repräsentation eines Baums
  - Jedenfalls solange der Speicherplatz reicht...

Höhe eines Binärbaums

- Die **Tiefe eines Knotens** ist sein Abstand (d.h. die Länge seines Weges) zur Wurzel
  - D.h. die Wurzel hat die Tiefe 0

- Die **Höhe eines Baums** ist die maximale Tiefe
  - D.h. die Länge eines längsten Weges von der Wurzel zu einem Blatt (Es kann mehrere solche – gleich langen – Wege geben!)
  - Mit anderen Worten: Der maximale Abstand zwischen einem Knoten und der Wurzel
  - Ein Baum, der nur aus einer Wurzel besteht, hat also die Höhe 0.



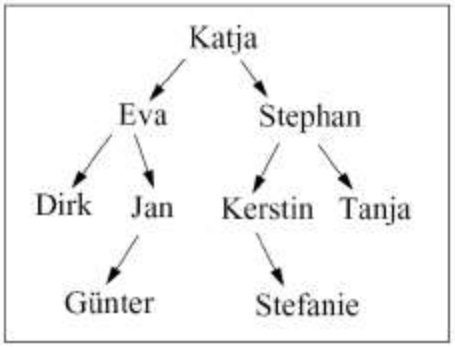
Höhe eines Binärbaums rekursiv

- Als lokale Methode in einer Knoten-Klasse:

```
int height() {
  if (left!=null && right!=null)
    return 1 + Math.max(left.height(),
                        right.height());
  else if (left!=null)
    return 1+left.height();
  else if (right!=null)
    return 1+right.height();
  else
    return 0;
}
```

Binäre Suchbäume

- Voraussetzung:
  - Jeder Knoten hat ein Schlüsselattribut
  - Die Menge der Schlüsselattribute ist total geordnet
- Für jeden Knoten mit Schlüsselattribut **s** gilt:
  - Alle Schlüssel im linken Unterbaum sind kleiner als **s**
  - Alle Schlüssel im rechten Unterbaum sind grösser als **s**



Zu den gleichen Schlüsselattributen gibt es verschiedene Suchbäume. Bsp: „Günter" anstelle von „Jan" (dann mit „Jan" als *rechten* Nachfolger von „Günter").

Einfügen in Suchbäume

```
static void insert (String n,int t,Person p) {
  if (n.compareTo(p.name) < 0)
    if (p.left != null)
      insert(n, t, p.left);
    else {
     p.left = new Person();
     p.left.name = n;
     p.left.telnr = t;
    }
  else
    if (p.right != null)
      insert(n, t, p.right);
    else {
      p.right = new Person();
      p.right.name = n; p.right.telnr = t;
    }
}
```

```
class Person {
  Person left;
  Person right;
  Person back;
  String name;
  int telnr;
}
```

Ein neuer Knoten wird immer als Blatt eingefügt

Einfügen der back-Referenz auf den Vorgänger als Denkübung

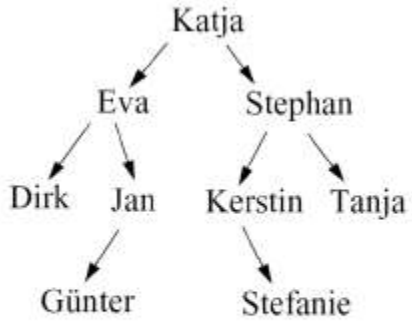
- *Beachte:* Der Aufruf der Methode setzt voraus, dass der dritte Parameter auf die Wurzel eines (nicht leeren) Suchbaums zeigt
- Was geschieht eigentlich, wenn es im Baum den Namen schon gibt?

Löschen von Knoten mit zwei Nachfolgern

- Ersetze das zu entfernende Element entweder durch
  - das grösste Element seines linken Teilbaums
    - (im linken Teilbaum ganz nach rechts laufen!)
  - oder das kleinste seines rechten Teilbaums
    - (im rechten Teilbaum ganz nach links laufen!)

Das Ersatzelement ist entweder ein Blatt oder ein Knoten mit einem einzigen Nachfolger

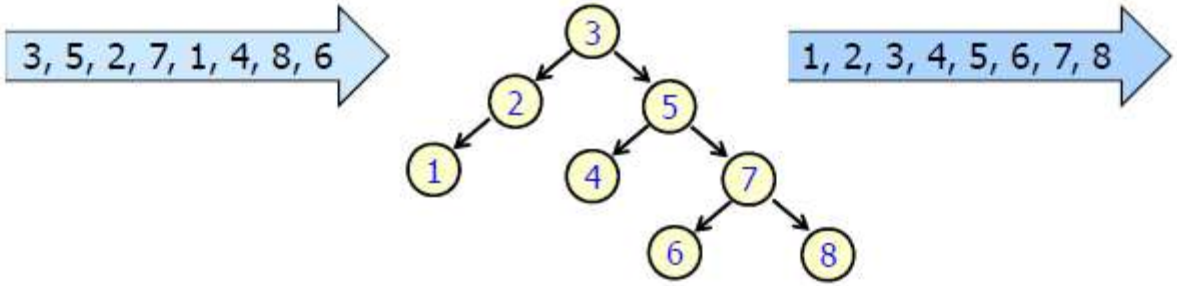
- Also durch den unmittelbaren Vorgänger bzw. Nachfolger entsprechend der Ordnung



Beispiele:
- Katja durch Jan
- Katja durch Kerstin
- Eva durch Dirk
- Eva durch Günter
- Stephan durch Stefanie
- Stephan durch Tanja

Sortieren mit Suchbäumen

- Prinzip:
  1. Unsortierte Eingabeliste in einen (anfangs leeren) Suchbaum überführen (d.h. „insert" nacheinander auf alle Elemente der Eingabeliste anwenden)
  2. „inorder" auf die Wurzel anwenden und beim Traversieren die Werte ausgeben



Ablauf

- Besprechung der Vorlesung
- Uebungsbezogene Themen:
  Java Generics, Binaerbaeume
- Zeit zum Programmieren...
  und fuer noch mehr Fragen

Übung 7

1. Array-Listen und Generics

   a. FilterFactory und (leere) IFilter-Implementierung

   b. Implementieren von filterRaw
      - Keine Generics: ArrayList als Raw Type (Compilerwarnungen)
      - Alle Studenten rausfiltern, die nicht genug Punkte fuer das Testat haben...

   c. Implementieren von filterGeneric
      - ArrayList<T> zeigt an, was darin gespeichert ist
      - Type checking beim Hinzufuegen zur Liste
      - ArrayList<T> liefert direkt Objekte des korrekten Typs (kein casting)

Übung 7

2. Binaere Suchbaeume

   a. Loeschen, Ersetzen durch kleinstes Element des rechten Teilbaums

   b. Factory und Implementierung zu IBinarySearchTreeUtils<T>
      - Tests arbeiten mit IBinarySearchTreeUtils<String>
      - UnlinkSmallest fuer Ersetzen durch kleinstes Element des rechten Teilbaums

Übung 7

2. Binaere Suchbaeume

- `height, isLeaf, hasOneChild`
- `preOrder, inOrder, postOrder`
- `insert`
- `find`
- `unlinkSmallest & remove`

Übung 7

3. Reversi

   a. Erstmal spielen

   b. Implementieren eines Spielers
      - Soll unter allen moeglichen Zuegen einen zufaelligen auswaehlen und durchfuehren
      - Ausgezeichnetes Javadoc! Muss nur eingebunden werden ☺
      - Tipp: Schaut euch mal die Methoden von Gameboard an:

        «gb.» + CTRL + SPACE

Ablauf

- Besprechung der Vorlesung
- Uebungsbezogene Themen:
  Java Generics, Binaerbaeume
- Zeit zum Programmieren...
  und fuer noch mehr Fragen

Projekt: Generics

- Vehicle und Object Lists

- Key insights:
  - Inheritance: List<Car> ist kein Subtyp von List<Vehicle>
  - Inheritance: List<Vehicle> ist kein Subtyp von List<Car>
  - Jedoch : List<?> ist der Supertyp fuer alle Listen!
  - Spezialisieren, z.B.: List<? extends Vehicle>

Informatik II

Übungsstunde 7

simon.mayer@inf.ethz.ch
Distributed Systems Group, ETH Zürich