JAVA 10: Programmieraufgaben zu Bäumen in Java

panitz

Zusammenfassung

Dieser Kurs besteht aus 10 kleinen Programmieraufgaben in der Programmiersprache Java.
Es sind 10 Methoden auf einer allgemeinen Baumstruktur zu schreiben.
Dieser Kurs ist adaptiert von dem Originalkurs für die Programmiersprache Haskell. Daher können manche Methoden etwas ungewöhnlich in Java erscheinen, weil nie eine Datenstruktur modofiziert wird, sondern immer neue Objekte erzeugt werden. (In Haskell gibt es keine Modifikation von Daten.)

Frage: Größe des Baumes

Schreiben Sie eine Funktion, die die Elemente, die in einem Baum enthalten sind, zählt. Java

```
      import java.util.*;
class Baum<A>{
  A element;
  List<Baum<A>> childNodes;
  boolean isEmpty;
  Baum(A e,List<Baum<A>> cN){
    element=e; childNodes=cN; isEmpty= element==null;
  }
  Baum(A e){this(e,new ArrayList<>());}
  Baum(){this(null);}

  int groesse(){
    return 0;
  }
}
^^I
```

Java

Lösung

Java

```
        import java.util.*;
class Baum<A>{
  A element;
  List<Baum<A>> childNodes;
  boolean isEmpty;
  Baum(A e,List<Baum<A>> cN){element=e; childNodes=cN; isEmpty= element==null;}
  Baum(A e){this(e,new ArrayList<>());}
  Baum(){this(null);}

  int groesse(){
    if(isEmpty) return 0;
    return 1 + childNodes.stream().map(c->c.groesse()).reduce(0,(x,y)->x+y);
  }
}
^^I
```

Java

Erläuterung

Der leere Baum enthält kein Element. Ansonsten wird die Summe dder Kindgrößen berechnet und 1 hinzu addiert.

Frage: Test auf Element im Baum

Schreiben Sie eine Funktion, die testet, ob ein bestimmtes Element im Baum gespeichert ist. Java

```
      import java.util.*;
class Baum<A>{
  A element;
  List<Baum<A>> childNodes;
  boolean isEmpty;
  Baum(A e,List<Baum<A>> cN){
    element=e; childNodes=cN; isEmpty= element==null;
  }
  Baum(A e){this(e,new ArrayList<>());}
  Baum(){this(null);}

  boolean enthaelt(){
    return false;
  }
}
^^I
```

Java

Lösung

Java

```
        import java.util.*;
class Baum<A>{
  A element;
  List<Baum<A>> childNodes;
  boolean isEmpty;
  Baum(A e,List<Baum<A>> cN){element=e; childNodes=cN; isEmpty= element==null;}
  Baum(A e){this(e,new ArrayList<>());}
  Baum(){this(null);}

  boolean enthaelt(A a){
    if (isEmpty) return false;
    return
      childNodes.stream().reduce(element.equals(a),(x,y)->x||y.enthaelt(a),(x,y)->x||y);
  }
}
^^I
```

Java

Erläuterung

Der leere Baum enthält keine Elemente. Ansonsten wird das Wurzelelement verglichen. Ist das Element nihct das Wurzelelement, so wird in den Kindern rekursiv gesucht.

Frage: Maximale Baumtiefe

Schreiben Sie eine Funktion, die die maximale Baumtiefe Berechnet. Java

```
      import java.util.*;
class Baum<A>{
  A element;
  List<Baum<A>> childNodes;
  boolean isEmpty;
  Baum(A e,List<Baum<A>> cN){
    element=e; childNodes=cN; isEmpty= element==null;
  }
  Baum(A e){this(e,new ArrayList<>());}
  Baum(){this(null);}

  int maxDepth(){
    return 0;
  }
}
^^I
```

Java

Lösung

Java

```
        import java.util.*;
class Baum<A>{
  A element;
  List<Baum<A>> childNodes;
  boolean isEmpty;
  Baum(A e,List<Baum<A>> cN){element=e; childNodes=cN; isEmpty= element==null;}
  Baum(A e){this(e,new ArrayList<>());}
  Baum(){this(null);}
  int maxDepth(){
    if (isEmpty) return 0;
    return
      1+childNodes.stream().reduce(0,(x,y)->Math.max(x,y.maxDepth()),Math::max);
  }
}
^^I
```

Java

Erläuterung

Der leere Baum hat die Tiefe 0. Ansonsten wird der Maximum der Baumtiefen der Kinder genommen und 1 hinzu addiert.

Frage: Maximale Kinderanzahl

Schreiben Sie eine Funktion, die die maximale Anzahl der Kinder der Knoten innerhalb eines Baumes berechnet. Java

```
      import java.util.*;
class Baum<A>{
  A element;
  List<Baum<A>> childNodes;
  boolean isEmpty;
  Baum(A e,List<Baum<A>> cN){
    element=e; childNodes=cN; isEmpty= element==null;
  }
  Baum(A e){this(e,new ArrayList<>());}
  Baum(){this(null);}

  int maxWidth(){
    return 0;
  }
 }
^^I
```

Java

Lösung

Java

```
        import java.util.*;
class Baum<A>{
  A element;
  List<Baum<A>> childNodes;
  boolean isEmpty;
  Baum(A e,List<Baum<A>> cN){
    element=e; childNodes=cN; isEmpty= element==null;
  }
  Baum(A e){this(e,new ArrayList<>());}
  Baum(){this(null);}

  int maxWidth(){
    if (isEmpty) return 0;
    return
      Math.max(childNodes.size()
              ,childNodes.stream().reduce(0,(r,c)->Math.max(r,c.maxWidth()),Math::max));
  }
}
^^I
```

Java

Erläuterung

Beim leeren Baum, gibt es nie Kinder, also ist die maximale Länge 0. Ansonsten wird das Maximum der rekursiven Aufrufe auf der Kinder genommen und mit der Anzahl der Kinder verglichen.

Frage: Alle Baumelemente

Schreiben Sie eine Funktion, die einer Liste aller im Baum gespeicherten Elemente selektiert. Die Elemente seien in Präorder sortiert. Java

```
      import java.util.*;
class Baum<A>{
  A element;
  List<Baum<A>> childNodes;
  boolean isEmpty;
  Baum(A e,List<Baum<A>> cN){
    element=e; childNodes=cN; isEmpty= element==null;
  }
  Baum(A e){this(e,new ArrayList<>());}
  Baum(){this(null);}

  List<A> elemente(){
    return new ArrayList<>();
  }
}
^^I
```

Java

Lösung

Java

```
        import java.util.*;
class Baum<A>{
  A element;
  List<Baum<A>> childNodes;
  boolean isEmpty;
  Baum(A e,List<Baum<A>> cN){
    element=e; childNodes=cN; isEmpty= element==null;
  }
  Baum(A e){this(e,new ArrayList<>());}
  Baum(){this(null);}

  List<A> elemente(){
    return elemente(new ArrayList<>());
  }
  List<A> elemente(List<A> result){
    if (isEmpty)return result;
    result.add(element);
    for (var c:childNodes)c.elemente(result);
    return result;
  }
}
^^I
```

Java

Erläuterung

Das Wurzelelement wird vorne an die Konkatination aller Kinderlemente angefügt.

Frage: Blattelemente eines Baumes

Java

```
      import java.util.*;
class Baum<A>{
  A element;
  List<Baum<A>> childNodes;
  boolean isEmpty;
  Baum(A e,List<Baum<A>> cN){
    element=e; childNodes=cN; isEmpty= element==null;
  }
  Baum(A e){this(e,new ArrayList<>());}
  Baum(){this(null);}

  List<A> fringe(){
    return new ArrayList<>();
  }
}
^^I
```

Java

Lösung

Java

```
        import java.util.*;
class Baum<A>{
  A element;
  List<Baum<A>> childNodes;
  boolean isEmpty;
  Baum(A e,List<Baum<A>> cN){element=e; childNodes=cN; isEmpty= element==null;}
  Baum(A e){this(e,new ArrayList<>());}
  Baum(){this(null);}

  List<A> fringe(){
    return fringe(new ArrayList<>());
  }
  List<A> fringe(List<A> result){
    if (isEmpty)return result;
    if (childNodes.isEmpty())result.add(element);
    for (var c:childNodes)c.fringe(result);
    return result;
  }
}
^^I
```

Java

Erläuterung

Blattknoten ergeben einelementige Ergebnislisten. Ansonsten sind die Ergebnisse der rekursiven Aufrufe zu konkatenieren.

Frage: Baum abbilden

Schreiben Sie eine Funktion, die aus einem Baum einen neuen Baum mit gleicher Struktur erzeugt, indem ein Funktionsparameter auf jedes Element des Baumes angewendet wird.

Java

```
      import java.util.*;
import java.util.function.*;
import java.util.stream.*;
class Baum<A>{
  A element;
  List<Baum<A>> childNodes;
  boolean isEmpty;
  Baum(A e,List<Baum<A>> cN){
    element=e; childNodes=cN; isEmpty= element==null;
  }
  Baum(A e){this(e,new ArrayList<>());}
  Baum(){this(null);}

  <B> Baum<B> abbilden(Function<A,B> f){
    return new Baum<>();
  }
}
^^I
```

Java

Lösung

Java

```
        import java.util.*;
import java.util.function.*;
import java.util.stream.*;
class Baum<A>{
  A element;
  List<Baum<A>> childNodes;
  boolean isEmpty;
  Baum(A e,List<Baum<A>> cN){element=e; childNodes=cN; isEmpty= element==null;}
  Baum(A e){this(e,new ArrayList<>());}
  Baum(){this(null);}
  <B> Baum<B> abbilden(Function<A,B> f){
    if (isEmpty) return new Baum<>();
    return
      new Baum<>
        (f.apply(element)
        ,childNodes.stream().map(c->c.abbilden(f)).collect(Collectors.toList()));
  }
}
^^I
```

Java

Erläuterung

Es handelt sich um die klassische Funktion `fmap` der Typklasse Funktor. Es wird die Funktion `f` auf das Wurzelelement angewendet und die gesamte Funktion rekursiv auf alle Kinder angewendet.

Frage: Kinderlisten Splitten

Schreiben Sie eine Funktion, die einem Baum so umwandelt, dass er keinen Knoten mit mehr als 5 Kindern mehr enthält. Hierzu sind bei einem Knoten mit mehr als 5 Kindern, die Kinder 2 bis 6 dem ersten Kind als neue Kinder hinten anzustellen. Sie werden also zu Enkelkindern. Dieses ist rekursiv so lange durchzuführen, bis kein Knoten mehr mehr als 5 Kinder hat.
Achtung Falle: `subList` erzeugt in Java keine neue Liste, sondern einen Blick auf die Ursprungsliste. Verwenden Sie: `new ArrayList<>(childNode.subList(1,6))` um an eine neue Teilliste zu kommen.
Java

```
      import java.util.*;
import java.util.function.*;
import java.util.stream.*;
class Baum<A>{
  A element;
  List<Baum<A>> childNodes;
  boolean isEmpty;
  Baum(A e,List<Baum<A>> cN){
    element=e; childNodes=cN; isEmpty= element==null;
  }
  Baum(A e){this(e,new ArrayList<>());}
  Baum(){this(null);}

  Baum<A> splitten(){
    return this;
  }
}
^^I
```

Java

Lösung

Java

```
        import java.util.*;
import java.util.function.*;
import java.util.stream.*;
class Baum<A>{
  A element;  List<Baum<A>> childNodes; boolean isEmpty;
  Baum(A e,List<Baum<A>> cN){element=e; childNodes=cN; isEmpty= element==null;}
  Baum(A e){this(e,new ArrayList<>());} Baum(){this(null);}
  Baum<A> splitten(){if (isEmpty)return new Baum<>();
    if (childNodes.size()<=5)
      return new Baum<>
        (element,childNodes.stream().map(c->c.splitten()).collect(Collectors.toList()));
    var c1   = childNodes.get(0);
    var c2_6 = new ArrayList<>(childNodes.subList(1,6));
    var cs   = new ArrayList<>(childNodes.subList(6,childNodes.size()));
    c2_6.addAll(0,c1.childNodes);
    cs.add(0,new Baum<>(c1.element,c2_6));
    return new Baum<>(element,cs).splitten();
  }}
^^I
```

Java

Erläuterung

Knoten mit mehr als 5 Kindern werden nach den Regeln transformiert. Anschließend wird die Rekursion auf den transformierten Baum ausgeführt. Hat der Knoten maximal 5 Kinder, so wird rekursiv jedes Kind traversiert.

Frage: Längster Pfad

Schreiben Sie eine Funktion, die die Liste der Elemente auf dem längsten Pfad im Baum berechnet. Gibt es mehrere gleichlange Pfade ist der linkeste zu nehmen. Java

```
      import java.util.*;
import java.util.function.*;
import java.util.stream.*;
class Baum<A>{
  A element;
  List<Baum<A>> childNodes;
  boolean isEmpty;
  Baum(A e,List<Baum<A>> cN){
    element=e; childNodes=cN; isEmpty= element==null;
  }
  Baum(A e){this(e,new ArrayList<>());}
  Baum(){this(null);}

  List<A> longestPath(){
    return new Baum<>();
  }
}
^^I
```

Java

Lösung

Java

```
        import java.util.*;
class Baum<A>{
  A element;
  List<Baum<A>> childNodes;
  boolean isEmpty;
  Baum(A e,List<Baum<A>> cN){element=e; childNodes=cN; isEmpty= element==null;}
  Baum(A e){this(e,new ArrayList<>());}
  Baum(){this(null);}
  List<A> longestPath(){
    if (isEmpty) return new ArrayList<>();
    List<A> result =
      childNodes.stream().reduce
        ((List<A>)new LinkedList<A>()
        ,(r,c)->{var cr=c.longestPath(); return r.size() >= cr.size() ? r : cr;}
        ,(x,y)->x.size()>=y.size()?x:y);
    result.add(0,element);
    return result;
  }
}
^^I
```

Java

Erläuterung

Es wird der längste Pfad der Kinder selektiert und auf diesen das Wurzelelement noch vorne dran gehängt.

Frage: Faltung auf Bäumen

Schreiben Sie eine Funktion, die alle Elemente eines Baumes mit Hilfe einer Operatorfunktion in Präorder auf ein Startelement aufrechnet. Java

```
      import java.util.*;
import java.util.function.*;
import java.util.stream.*;
class Baum<A>{
  A element;
  List<Baum<A>> childNodes;
  boolean isEmpty;
  Baum(A e,List<Baum<A>> cN){
    element=e; childNodes=cN; isEmpty= element==null;
  }
  Baum(A e){this(e,new ArrayList<>());}
  Baum(){this(null);}

  <B> B faltung(B start,BiFunction<B,A,B> f){
    return start;
  }
}
^^I
```

Java

Lösung

Java

```
        import java.util.*;
import java.util.function.*;
import java.util.stream.*;
class Baum<A>{
  A element;
  List<Baum<A>> childNodes;
  boolean isEmpty;
  Baum(A e,List<Baum<A>> cN){element=e; childNodes=cN; isEmpty= element==null;}
  Baum(A e){this(e,new ArrayList<>());}
  Baum(){this(null);}
  <B> B faltung(B start,BiFunction<B,A,B> f){
    if(isEmpty)return start;
    var result = f.apply(start,element);
    for (var c:childNodes){
      result = c.faltung(result,f);
    }
    return result;
  }
}
^^I
```

Java

Erläuterung

Zunächst wird das Wurzelelement mit dem Startelement verknüpft. das Ergbnis wird für die Standard-faltung auf Listen über das Ergebnis der rekursiven Kindfaltungen als neues Startelement verwendet.