Name: _____

Abiturprüfung 2019

Informatik, Leistungskurs

Aufgabenstellung:

Im Folgenden soll die Simulation eines Paketverteilungszentrums entwickelt werden. In einem Paketverteilungszentrum gibt es viele Förderbänder, die Pakete zu Verladestellen transportieren. Alle Pakete werden am Anfang eines eindeutigen Zulieferbandes in das System gegeben und in regelmäßigen Zeittakten zum nächsten Segment auf dem Band transportiert. Entweder laufen sie so zum Ende des Bandes und werden dort im nächsten Transportschritt verladen oder wechseln bei einem Übergang auf den Anfang eines anderen Bandes, das zu einer anderen zugehörigen Verladestelle führt.

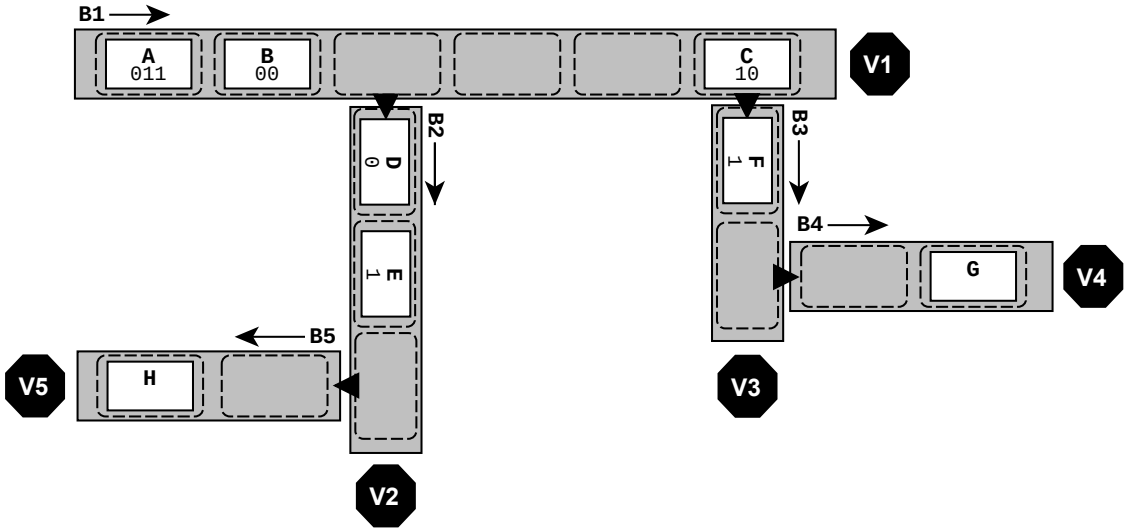


Abbildung 1: Ein Paketverteilungszentrum mit Zulieferband B1

Abbildung 1 zeigt ein Paketverteilungszentrum mit fünf Bändern (`B1` bis `B5`) und den zugehörigen Verladestellen (`V1` bis `V5`). Zwischen den Bändern existieren vier Übergänge (Pfeilsymbol ►).

Das Paketverteilungszentrum transportiert zurzeit acht Pakete mit den IDs `A` bis `H`. Jedes Paket verfügt zusätzlich über einen Weg-Code aus den Zeichen `0` und `1`. Immer, wenn ein Paket in einem Segment mit einem Übergang steht, wird beim nächsten Transportschritt zunächst das vorderste Zeichen des Weg-Codes entfernt. Ist dieses Zeichen eine `0`, wird der Übergang ignoriert und das Paket auf demselben Band weiterbefördert bzw. verladen; ist es eine `1`, wird der Übergang benutzt und somit auf das andere Band gewechselt. Kommt das Paket an keinem Übergang mehr vorbei, ist der Weg-Code leer.

Name: _____

In Abbildung 1 wird Paket `D` also zu Verladestation `V2` transportiert, das Paket `E` zu `V5` und `G` zu `V4`.

Zur Simulation des Paketverteilungszentrums werden die Segmente der Förderbänder als Knoten eines Binärbaums modelliert.

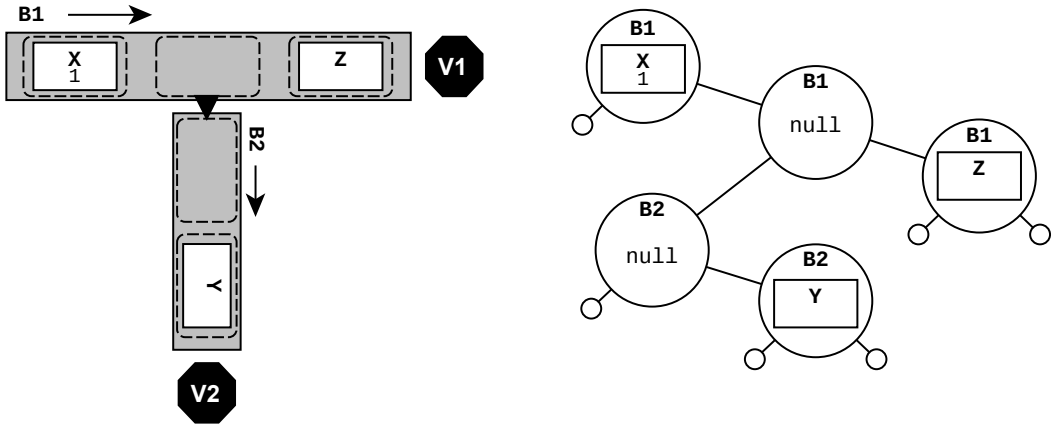


Abbildung 2: Beispiel der Modellierung als Binärbaum

Abbildung 2 zeigt ein beispielhaftes Paketverteilungszentrum und seine Modellierung als Binärbaum. Jeder Knoten verwaltet genau ein Bandsegment. Jedes Bandsegment besitzt eine Band-ID und verweist auf ein Paket oder `null`, falls sich auf dem Segment kein Paket befindet (das zweite Bandsegment auf Band `B1` enthält beispielsweise kein Paket). Der rechte Nachfolgeknoten jedes Bandsegments stellt das nachfolgende Segment desselben Bands dar. Gibt es keinen rechten Nachfolger, ist das Band zu Ende und eine Verladestation schließt sich an (diese wird im Baum nicht repräsentiert). Hat ein Knoten einen linken Nachfolger, so handelt es sich um ein Bandsegment mit einem Übergang und der linke Nachfolgeknoten modelliert das erste Segment des anderen Bands. Ein Binärbaum dieser Art wird im Folgenden als **Transportbaum** bezeichnet.

a) Betrachtet wird das Paketverteilungszentrum aus Abbildung 1.

Beschreiben Sie allgemein die Struktur eines Binärbaums.

Überführen Sie die Situation aus Abbildung 1 in einen Transportbaum.

Ermitteln Sie, in welche Verladestationen die Pakete `A`, `B` *und* `C` *befördert werden und erläutern Sie jeweils Ihren Lösungsweg.*

Ermitteln Sie die anfängliche Weg-Codierung von Paket `G`*, das auf Band* `B1` *gestartet war.*

(13 Punkte)

Name: _____

Das folgende Implementationsdiagramm zeigt eine Teilmodellierung eines Paketverteilungszentrums.

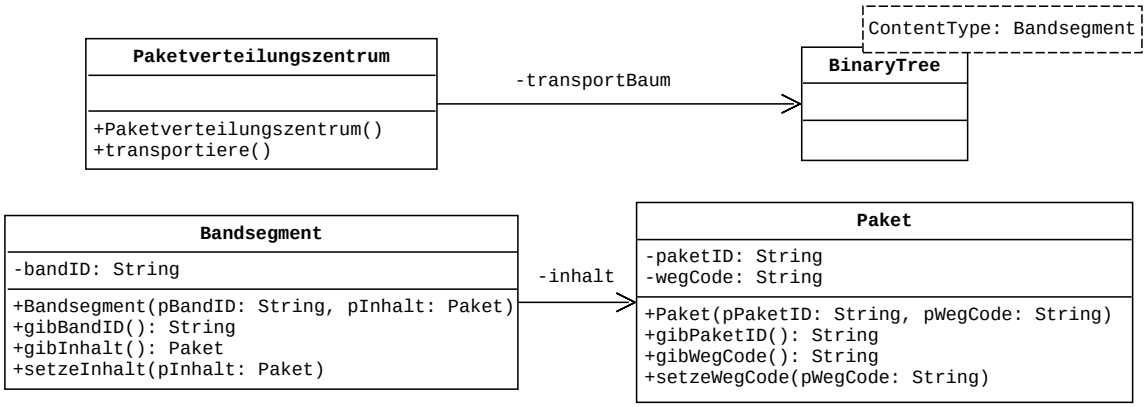


Abbildung 3: Ausschnitthaftes Implementationsdiagramm der Simulationssoftware

Eine detaillierte Dokumentation der verwendeten Klassen findet sich im Anhang.

b) Ein Paketverteilungszentrum sei durch den Transportbaum in Abbildung 4 codiert und zur Laufzeit durch das Attribut `transportBaum` der Klasse `Paketverteilungszentrum` repräsentiert:

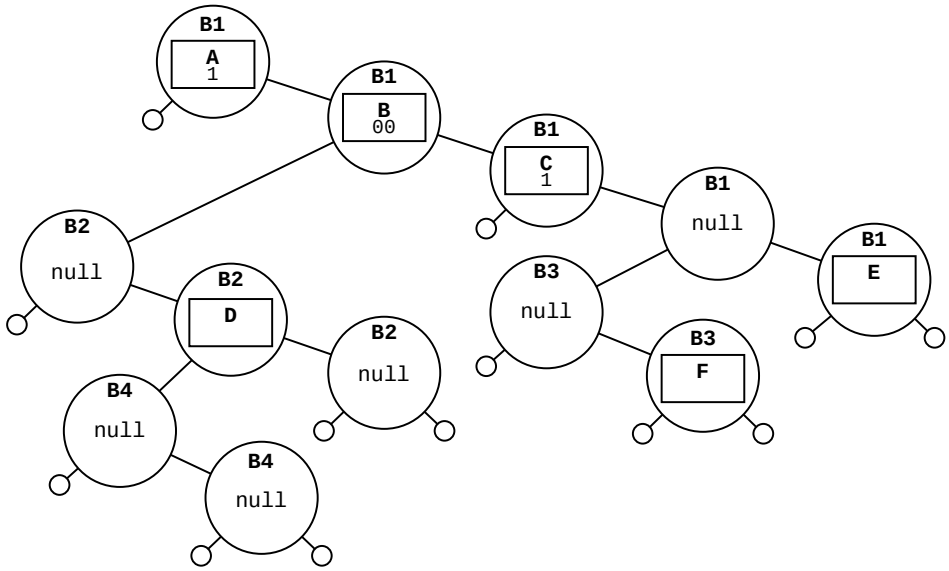


Abbildung 4: Ein Beispiel-Transportbaum

Name: _____

In der Klasse `Paketverteilungszentrum` finden sich folgende undokumentierte und unkommentierte Methoden:

```
 1 public String wasLiefereIch() {
 2   return wasLiefereIch(transportBaum);
 3 }
 4 private String wasLiefereIch(BinaryTree<Bandsegment> pB) {
 5   String r = pB.getContent().gibBandID() + ":";
 6   Queue<BinaryTree<Bandsegment>> q
 7       = new Queue<BinaryTree<Bandsegment>>();
 8   while (!pB.isEmpty()) {
 9     if (!pB.getLeftTree().isEmpty()) {
10       q.enqueue(pB.getLeftTree());
11     }
12     Paket p = pB.getContent().gibInhalt();
13     if (p != null) {
14       r = r + "[" + p.gibPaketID() + "]";
15     } else {
16       r = r + "[_]";
17     }
18     pB = pB.getRightTree();
19   }
20   while (!q.isEmpty()) {
21     r = r + "#" + wasLiefereIch(q.front());
22     q.dequeue();
23   }
24   return r;
25 }
```

Analysieren Sie die Methode `wasLiefereIch`*, indem Sie deren Rückgabe für den in Abbildung 4 gegebenen Transportbaum ermitteln.*

Erläutern Sie den Zweck und die Funktionsweise der Methode im Sachkontext. Erläutern Sie dabei insbesondere die Bedeutung der Schlange `q` *im Sachzusammenhang.*

(15 Punkte)

Name: _____

c) Unter einem Pfad versteht man den Weg, den ein Paket von seiner derzeitigen Position im Verteilungszentrum aus bis zur Verladestation noch gehen wird. Dieser Pfad kann angegeben werden, indem man nacheinander die Band-IDs der Bandsegmente angibt, die das Paket noch besuchen wird.

In Abbildung 1 hat das Paket `D` ab der aktuellen Position den Pfad `B2` – `B2`, das Paket `E` hat den Pfad `B2` – `B5` – `B5`.

Die Klasse `Paketverteilungszentrum` soll um eine Methode `lieferePfad` erweitert werden, die den Pfad desjenigen Pakets, das sich an einem beliebigen Bandsegment befindet, ermittelt und als Schlange zurückgibt. Enthält das Bandsegment kein Paket, soll eine leere Schlange zurückgegeben werden. Der Weg-Code des betrachteten Pakets darf durch die Ausführung der Methode nicht verändert werden.

Die Methode hat folgenden Methodenkopf:

```
public Queue<String> lieferePfad(BinaryTree<Bandsegment> pBaum)
```

Die Wurzel des durch den Parameter `pBaum` angegebenen Teilbaums des Transportbaums entspricht dabei der betrachteten Position.

Entwickeln und erläutern Sie ein algorithmisches Verfahren für diese Methode.

Implementieren Sie die Methode `lieferePfad`.

(14 Punkte)

Name: _____

d) Ein Kritikpunkt am Design des Paketverteilungszentrums ist, dass jeder Übergang von einem Förderband lediglich zum Anfangssegment eines anderen Bandes führen kann und nicht zu einem beliebigen Segment. Ein Mitarbeiter schlägt daher vor, die Spezifikation der Anlage in genau diesem Punkt zu ändern und solche Übergänge in Zukunft zuzulassen.

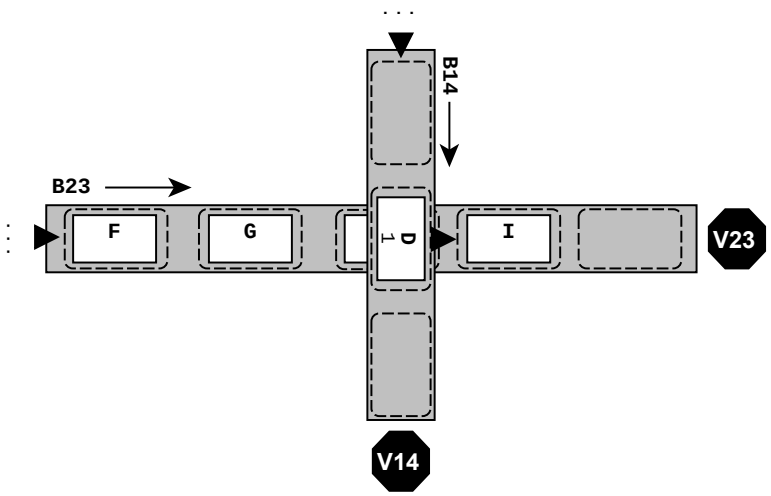


Abbildung 5: Ausschnitt aus einem Paketverteilungszentrum nach geänderter Spezifikation

Abbildung 5 zeigt einen Ausschnitt aus einer Beispiel-Anlage. Hier gibt es einen Übergang von `B14` nach `B23` (`B23` läuft unter `B14` durch), der nicht auf den Anfang von `B23` führt.

Begründen Sie, dass bei der Umsetzung der vorgeschlagenen Änderung die ursprünglich für die Modellierung verwendete Datenstruktur des Binärbaums nicht mehr geeignet ist, und geben Sie eine Alternative an.

Begründen Sie, dass die vorgeschlagene Änderung grundlegende Konsequenzen für die Transport-Algorithmik nach sich zieht.

(8 Punkte)

Zugelassene Hilfsmittel:

- GTR (grafikfähiger Taschenrechner) oder CAS (Computer-Algebra-System)
- Wörterbuch zur deutschen Rechtschreibung

Name: _____

Anhang

Dokumentationen der verwendeten Klassen

Alle in diesem Abschnitt nicht weiter dokumentierten **`gib`**...- bzw. **`setze`**...-Methoden entsprechen den üblichen Funktionsweisen.

Die Klasse `String`

Ein Objekt der Klasse `String` repräsentiert eine Zeichenkette.

Ausschnitt aus der Dokumentation der Klasse `String`

Anfrage **`char charAt(int index)`**

Die Anfrage gibt das Zeichen am angegebenen Index zurück. Der Index hat einen Wertebereich von `0` bis `length()-1`. Das erste Zeichen dieser Zeichenkette ist an Index `0`, das nächste an Index `1` und so weiter, wie bei Array-Indizes.

Anfrage **`String substring(int beginIndex)`**

Gibt eine neue Zeichenkette zurück, die eine Unter-Zeichenkette dieser Zeichenkette ist. Die Unter-Zeichenkette beginnt mit dem Zeichen, das sich am angegebenen Index befindet (beginnend bei Index `0`) und reicht bis zum Ende dieser Zeichenkette.
Beispiele:

- `"unschön".substring(2)` gibt `"schön"` zurück
- `"Leere".substring(5)` gibt `""` zurück (eine leere Zeichenkette)

Die Klasse `Paketverteilungszentrum`

Die Klasse dient der Simulation eines Paketverteilungszentrums. Sie erzeugt einen Transportbaum und simuliert schrittweise den Transport der Pakete.

Ausschnitt aus der Dokumentation der Klasse `Paketverteilungszentrum`

Kontruktor **`Paketverteilungszentrum()`**

Ein Paketverteilungszentrum mit dem durch die reale Anlage gegebenen Transportbaum wird initialisiert.

Auftrag **`void transportiere()`**

Die Methode rückt alle im Transportbaum enthaltenen Pakete um ein Bandsegment (entsprechend ihres jeweiligen Weg-Codes) weiter oder aber verlädt sie (verladene Pakete werden vom Transportbaum nicht mehr verwaltet). Lässt ein Paket bei diesem Transportschritt einen Übergang hinter sich, wird das vorderste Zeichen seines Weg-Codes gelöscht.

Name: _____

Die Klasse Paket

Jedes Objekt dieser Klasse modelliert ein Paket auf einem bestimmten Segment eines Förderbands.

Ausschnitt aus der Dokumentation der Klasse `Paket`

Konstruktor `Paket(String pPaketID, String pWegCode)`

Ein Paket wird initialisiert. Erläuterung der Parameter:

- `pPaketID` gibt die eindeutige ID des Pakets an.
- `pWegCode` codiert den Weg, den dieses Paket durch die Anlage nehmen muss. Das jeweils vorderste Zeichen im String markiert dabei, wie am nächsten Bandsegment der Anlage, das einen Übergang besitzt, zu verfahren ist:
  - `0` bedeutet, dass der Übergang nicht benutzt wird.
  - `1` bedeutet, dass der Übergang benutzt wird, also zum Anfang eines anderen Bands umgelenkt wird.

  Alle weiteren Zeichen stehen analog für die im weiteren Verlauf des Transports anstehenden Entscheidungen.

Die Klasse Bandsegment

Der Transportbaum verwaltet Objekte dieser Klasse. Ein Objekt dieser Klasse modelliert ein bestimmtes Segment eines Förderbands. Jedes Förderband besteht aus beliebig vielen Segmenten. Segmente sind gekennzeichnet durch die ID des Förderbandes, zu dem sie gehören, und verwalten ein Objekt der Klasse `Paket`, falls das Bandsegment ein Paket enthält (andernfalls ist die entsprechende Referenz `null`).

Ausschnitt aus der Dokumentation der Klasse `Bandsegment`

Konstruktor `Bandsegment(String pBandID, Paket pInhalt)`

Ein Bandsegment, das zum Förderband mit der ID `pBandID` gehört und das Paket `pInhalt` als Inhalt besitzt, wird initialisiert. Hat `pInhalt` den Wert `null`, wird ein leeres Bandsegment initialisiert.

Name: _____

Die generische Klasse Queue<ContentType>

Objekte der generischen Klasse **`Queue`** (Warteschlange) verwalten beliebige Objekte vom Typ **`ContentType`** nach dem First-In-First-Out-Prinzip, d. h., das zuerst abgelegte Objekt wird als erstes wieder entnommen. Alle Methoden haben eine konstante Laufzeit, unabhängig von der Anzahl der verwalteten Objekte.

Dokumentation der generischen Klasse `Queue<ContentType>`

Konstruktor **`Queue()`**

Eine leere Schlange wird erzeugt. Objekte, die in dieser Schlange verwaltet werden, müssen vom Typ `ContentType` sein.

Anfrage **`boolean isEmpty()`**

Die Anfrage liefert den Wert `true`, wenn die Schlange keine Objekte enthält, sonst liefert sie den Wert `false`.

Auftrag **`void enqueue(ContentType pContent)`**

Das Objekt `pContent` wird an die Schlange angehängt. Falls `pContent` gleich `null` ist, bleibt die Schlange unverändert.

Auftrag **`void dequeue()`**

Das erste Objekt wird aus der Schlange entfernt. Falls die Schlange leer ist, wird sie nicht verändert.

Anfrage **`ContentType front()`**

Die Anfrage liefert das erste Objekt der Schlange. Die Schlange bleibt unverändert. Falls die Schlange leer ist, wird `null` zurückgegeben.

Name: _____

Die Klasse BinaryTree<ContentType>

Mithilfe der generischen Klasse **`BinaryTree`** können beliebig viele Objekte vom Typ **`ContentType`** in einem Binärbaum verwaltet werden. Ein Objekt der Klasse stellt entweder einen leeren Baum dar oder verwaltet ein Inhaltsobjekt sowie einen linken und einen rechten Teilbaum, die ebenfalls Objekte der generischen Klasse **`BinaryTree`** sind.

Dokumentation der Klasse `BinaryTree<ContentType>`

Konstruktor **`BinaryTree()`**

Nach dem Aufruf des Konstruktors existiert ein leerer Binärbaum. Objekte, die in diesem Binärbaum verwaltet werden, müssen vom Typ `ContentType` sein.

Konstruktor **`BinaryTree(ContentType pContent)`**

Wenn der Parameter `pContent` ungleich `null` ist, existiert nach dem Aufruf des Konstruktors der Binärbaum und hat `pContent` als Inhaltsobjekt und zwei leere Teilbäume. Falls der Parameter `null` ist, wird ein leerer Binärbaum erzeugt.

Konstruktor
```
BinaryTree(ContentType pContent,
           BinaryTree<ContentType> pLeftTree,
           BinaryTree<ContentType> pRightTree)
```

Wenn der Parameter `pContent` ungleich `null` ist, wird ein Binärbaum mit `pContent` als Inhaltsobjekt und den beiden Teilbäumen `pLeftTree` und `pRightTree` erzeugt. Sind `pLeftTree` oder `pRightTree` gleich `null`, wird der entsprechende Teilbaum als leerer Binärbaum eingefügt. Wenn der Parameter `pContent` gleich `null` ist, wird ein leerer Binärbaum erzeugt.

Anfrage **`boolean isEmpty()`**

Diese Anfrage liefert den Wahrheitswert `true`, wenn der Binärbaum leer ist, sonst liefert sie den Wert `false`.

Auftrag **`void setContent(ContentType pContent)`**

Wenn der Binärbaum leer ist, wird der Parameter `pContent` als Inhaltsobjekt sowie ein leerer linker und rechter Teilbaum eingefügt. Ist der Binärbaum nicht leer, wird das Inhaltsobjekt durch `pContent` ersetzt. Die Teilbäume werden nicht geändert. Wenn `pContent` `null` ist, bleibt der Binärbaum unverändert.

Name: _____

Anfrage **`ContentType getContent()`**

Diese Anfrage liefert das Inhaltsobjekt des Binärbaums. Wenn der Binärbaum leer ist, wird `null` zurückgegeben.

Auftrag **`void setLeftTree(BinaryTree<ContentType> pTree)`**

Wenn der Binärbaum leer ist, wird `pTree` nicht angehängt. Andernfalls erhält der Binärbaum den übergebenen Baum als linken Teilbaum. Falls der Parameter `null` ist, ändert sich nichts.

Auftrag **`void setRightTree(BinaryTree<ContentType> pTree)`**

Wenn der Binärbaum leer ist, wird `pTree` nicht angehängt. Andernfalls erhält der Binärbaum den übergebenen Baum als rechten Teilbaum. Falls der Parameter `null` ist, ändert sich nichts.

Anfrage **`BinaryTree<ContentType> getLeftTree()`**

Diese Anfrage liefert den linken Teilbaum des Binärbaumes. Der Binärbaum ändert sich nicht. Wenn der Binärbaum leer ist, wird `null` zurückgegeben.

Anfrage **`BinaryTree<ContentType> getRightTree()`**

Diese Anfrage liefert den rechten Teilbaum des Binärbaumes. Der Binärbaum ändert sich nicht. Wenn der Binärbaum leer ist, wird `null` zurückgegeben.

Unterlagen für die Lehrkraft

Abiturprüfung 2019

Informatik, Leistungskurs

1. Aufgabenart

Analyse, Modellierung und Implementation von kontextbezogenen Problemstellungen mit Schwerpunkt auf den Inhaltsfeldern Daten und ihre Strukturierung, Algorithmen und Informatiksysteme

2. Aufgabenstellung[1]

siehe Prüfungsaufgabe

3. Materialgrundlage

entfällt

4. Bezüge zum Kernlehrplan und zu den Vorgaben 2019

Die Aufgaben weisen vielfältige Bezüge zu den Kompetenzerwartungen und Inhaltsfeldern des Kernlehrplans bzw. zu den in den Vorgaben ausgewiesenen Fokussierungen auf.
Im Folgenden wird auf Bezüge von zentraler Bedeutung hingewiesen.

1. *Inhaltsfelder und inhaltliche Schwerpunkte*
   Daten und ihre Strukturierung
   - Objekte und Klassen
     - Entwurfsdiagramme und Implementationsdiagramme
     - Nicht-lineare Strukturen (Binärbaum)
     - Lineare Strukturen (Schlange)

   Algorithmen
   - Analyse, Entwurf und Implementierung von Algorithmen

   Formale Sprachen und Automaten
   - Syntax und Semantik einer Programmiersprache
     - Java

2. *Medien/Materialien*
   - entfällt

5. Zugelassene Hilfsmittel

- GTR (grafikfähiger Taschenrechner) oder CAS (Computer-Algebra-System)
- Wörterbuch zur deutschen Rechtschreibung

[1] Die Aufgabenstellung deckt inhaltlich alle drei Anforderungsbereiche ab.

6. Modelllösungen

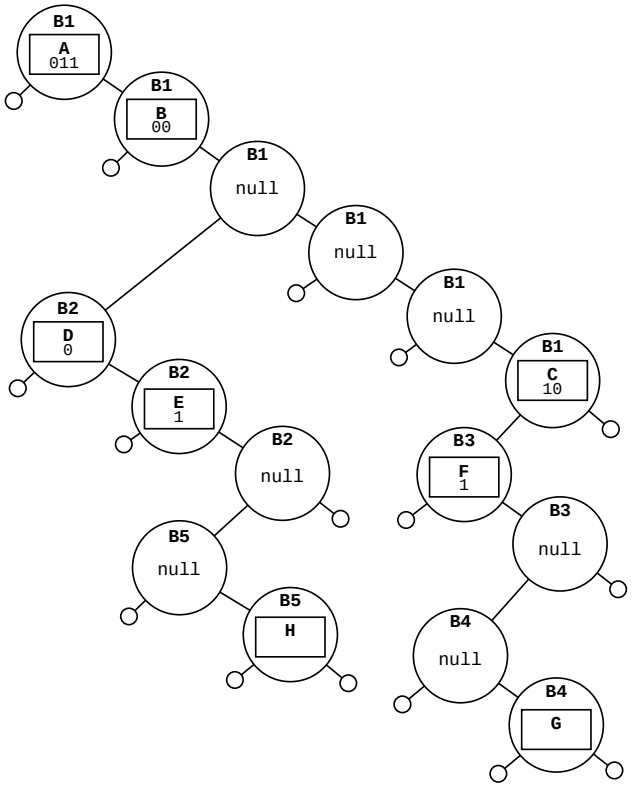
Die jeweilige Modelllösung stellt eine mögliche Lösung bzw. Lösungsskizze dar. Der gewählte Lösungsansatz und -weg der Schülerinnen und Schüler muss nicht identisch mit dem der Modelllösung sein. Sachlich richtige Alternativen werden mit entsprechender Punktzahl bewertet (Bewertungsbogen: Zeile „Sachlich richtige Lösungsalternative zur Modelllösung“).

Teilaufgabe a)

Der Binärbaum ist eine rekursive Datenstruktur, die sich durch folgende Eigenschaften auszeichnet:

- Jeder Binärbaum ist entweder leer oder besteht aus einem Wurzelelement, das zwei Binärbäume als Nachfolger besitzt.
- Jedes Element eines nicht-leeren Binärbaums verwaltet ein Inhaltsobjekt.

Der sich aus dem gegebenen Paketverteilungszentrum ergebende Transportbaum hat die rechtsstehende Form.



Paket `A` muss laut seiner Weg-Codierung `011` den ersten sich auf seinem Pfad durch die Anlage anbietenden Übergang auslassen und die danach folgenden zwei nutzen. Vom Segment des Pakets `A` aus führt der nächste verfügbare Übergang zu Band `B2`, dieser wird ausgelassen. Der nächste Übergang führt zu `B3` und der dortige nächste schließlich zu `B4`. Paket `A` wird also auf Verladestation `V4` verladen, die am Ende von Band `B4` steht.

Paket `B` wird laut seiner Weg-Codierung `00` noch an 2 möglichen Übergängen vorbeigeführt, die beide ohne Umlenkung passiert werden. Paket `B` bleibt also auf dem Band, auf dem es sich momentan befindet und wird folglich bei Station `V1` verladen.

Paket `C` wird laut seiner Weg-Codierung `10` noch auf 2 Übergänge stoßen, von denen der erste zu nehmen ist und der zweite auszulassen ist. Paket `C` steht gerade an diesem zu nehmenden Übergang und wird daher auf Band `B3` umgeleitet und bei Station `V3` verladen.

Paket `G` ist von `B1` nach `B4` gelangt. Unterwegs wurde der Übergang `B1` – `B2` ausgelassen und die Übergange `B1` – `B3` und `B3` – `B4` genommen. Als ursprüngliche Weg-Codierung ergibt sich also `011`.

Teilaufgabe b)

Der von der Methode zurückgegebene String lautet:

```
B1:[A][B][C][_][E]#B2:[_][D][_]#B4:[_][_]#B3:[_][F]
```

Die Methode gibt einen String zurück, der den Inhalt des Transportbaums in linearisierter Form repräsentiert. Die Paket-IDs werden darin in der Reihenfolge, in der die Pakete auf den Bändern liegen, jeweils blockweise ausgegeben. Jeder Block wird eingeleitet mit der Band-ID, gefolgt von einem Doppelpunkt. Die einzelnen Paket-IDs werden jeweils in einem eckigen Klammerpaar ausgegeben; bei leeren Paketen wird die Paket-ID durch einen Unterstrich ersetzt. Die Abgrenzung eines Blocks zum jeweils nächsten erfolgt durch ein Doppelkreuz.

Die Methode hat also den Zweck, den Beladungszustand des Paketverteilungszentrums nach Bändern gruppiert auszugeben. Ausgehend vom Startband werden die Bänder dabei in der Reihenfolge rekursiv ausgegeben, in der die zugehörigen Übergänge in Transportrichtung auftauchen.

Algorithmisch wird dies dadurch erreicht, dass der Ausgabestring `r` zunächst mit der aktuellen Band-ID und dem Doppelpunkt initialisiert wird (Z. 5). In den Zeilen 8 bis 19 erfolgt dann eine Iteration über alle rechten Nachfolger (Z. 18), was im Sachkontext bedeutet, dass durch die Schleife alle Bandsegmente des aktuellen Bandes der Reihe nach besucht werden. Für jeden zugehörigen Knoten wird zunächst überprüft, ob er auch einen linken Nachfolger besitzt; in diesem Fall wird der Knoten in der Queue `q` abgelegt (Z. 9 – 11). Enthält das Bandsegment des aktuellen Knotens ein Paket, wird dessen ID an den Rückgabe-String angehängt bzw. andernfalls das leere Bandsegment durch den Unterstrich angedeutet (Z. 13 – 17).

Beim Durchlaufen des entsprechenden Astes dient die Queue `q` also dazu, alle Knoten mit Übergängen zwischenzuspeichern, die unterwegs vorkommen. Nach der kompletten Iteration des Astes wird die Queue `q` abgebaut, indem für jeden ihrer Einträge `wasLiefereIch` rekursiv aufgerufen wird und damit die Ausgabe des Förderbands, das am entsprechenden Übergang beginnt, durch das Doppelkreuz von der bisher erzeugten Ausgabe getrennt, rekursiv an die Ausgabe angehängt wird (Z. 20 – 23).

Teilaufgabe c)

Um den Pfad zu ermitteln, wird zunächst das Paket im Bandsegment der Wurzel des betrachteten Teilbaumes ermittelt. Falls sich dort ein Paket befindet, wird sich mithilfe einer Index-Variablen gemerkt, an welcher Position sich im `wegCode` dieses Pakets die Angabe über „nehmen“ oder „nicht nehmen“ für den nächsten Übergang befindet. Der Baum wird nun nach den vorgegebenen Regeln durchlaufen: Immer dann, wenn es einen linken Nachfolger gibt, wird der `wegCode` des Pakets am gerade aktuellen Index ausgelesen und dieser anschließend inkrementiert. Gibt es keinen linken Nachfolger, wird immer der rechte gewählt. Die ID des gewählten Nachfolgers (sofern dieser nicht leer ist) wird danach als neue Pfadkomponente in die Rückgabeliste aufgenommen. Dies geschieht so lange, bis man an einem leeren Knoten angelangt ist.

Eine mögliche Implementation der Methode lautet:

```
public Queue<String> lieferePfad(BinaryTree<Bandsegment> pBaum) {
  Queue<String> ret = new Queue<String>();
  if (!pBaum.isEmpty()) {
    Bandsegment aktBandsegment = pBaum.getContent();
    Paket paket = aktBandsegment.gibInhalt();
    if (paket != null) {
      int wegIndex = 0;
      do {
        char aktWegCode = '0';
        if (!pBaum.getLeftTree().isEmpty()) {
          aktWegCode = paket.gibWegCode().charAt(wegIndex);
          wegIndex++;
        }
        if (aktWegCode == '0') {
          pBaum = pBaum.getRightTree();
        } else {
          pBaum = pBaum.getLeftTree();
        }
        if (!pBaum.isEmpty()) {
          ret.enqueue(pBaum.getContent().gibBandID());
        }
      } while (!pBaum.isEmpty());
    }
  }
  return ret;
}
```

Teilaufgabe d)

Aus der rekursiven Definition eines Binärbaums folgt, dass alle enthaltenen Teilbäume genau einen Vorgänger haben, der sie als linken oder rechten Nachfolger verwaltet (mit Ausnahme der Wurzel des Gesamtbaumes, die selbst keinen Vorgänger hat). In einer Anlage, in der jedes Bandsegment neben seinem „natürlichen“ Vorgänger-Segment auf demselben Band auch einen Zugang von einem anderen Band besitzen kann, muss es dementsprechend einen weiteren Vorgänger auch im Datenmodell geben. Dies ist im Binärbaum nicht mehr abbildbar. Die Datenstruktur des Graphs wäre zur Modellierung nun eine sinnvolle Wahl.

Für die Transport-Algorithmik hätte dies gravierende Konsequenzen. In der aktuellen Modellierung, in der Pakete immer nur am Anfang eines Bandes aufgenommen werden, ist sichergestellt, dass das dortige Bandsegment „frei“ ist, denn beim Weitertransport jedes Bandes entsteht nach jedem Takt am Anfang ein leeres Segment. Darf auch in der Mitte eines Bandes „eingefügt“ werden, ist dies nicht gewährleistet, denn das von einem anderen Band zugelieferte Paket könnte in Konflikt geraten mit einem schon auf dem Zielband befindlichen Paket (Abbildung 5 der Aufgabe offenbart dies am Übergang von Band `B14` zu `B23`). Diesem Umstand muss die Transport-Algorithmik Rechnung tragen, z. B. indem das zuliefernde Band angehalten wird (was ggf. das Anhalten weiterer Bänder nach sich ziehen kann) oder ein dynamischer Speicher (z. B. ein Stapelspeicher) an den problematischen Segmenten eingeführt wird, der aktuell nicht aufnehmbare Pakete so lange zwischenspeichert, bis wieder ein freies Segment zur Verfügung steht.

7. Teilleistungen – Kriterien / Bewertungsbogen zur Prüfungsarbeit

Name des Prüflings:_____ Kursbezeichnung:_____

Schule: _____

Teilaufgabe a)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK[2]**	**ZK**	**DK**
1	beschreibt allgemein die Struktur eines Binärbaums.	2			
2	überführt die abgebildete Situation in einen Transportbaum.	4			
3	ermittelt die Verladestationen der Pakete und erläutert den jeweiligen Lösungsweg.	4			
4	ermittelt die Weg-Codierung von Paket `G`.	3			
Sachlich richtige Lösungsalternative zur Modelllösung: (13) …… ……					
	Summe Teilaufgabe a)	**13**			

Teilaufgabe b)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	ermittelt durch Analyse die Rückgabe der Methode.	4			
2	erläutert Zweck und Funktionsweise der Methode.	7			
3	erläutert die Bedeutung der Queue `q` im Sachzusammenhang.	4			
Sachlich richtige Lösungsalternative zur Modelllösung: (15) …… ……					
	Summe Teilaufgabe b)	**15**			

[2] EK = Erstkorrektur; ZK = Zweitkorrektur; DK = Drittkorrektur

Teilaufgabe c)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	entwickelt und erläutert ein algorithmisches Verfahren.	6			
2	implementiert die Methode.	8			
Sachlich richtige Lösungsalternative zur Modelllösung: (14) …… ……					
	Summe Teilaufgabe c)	**14**			

Teilaufgabe d)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	begründet, dass der Binärbaum für die Modellierung nicht mehr geeignet ist und gibt eine Alternative an.	4			
2	begründet, dass die Änderung grundlegende Konsequenzen für die Transport-Algorithmik nach sich zieht.	4			
Sachlich richtige Lösungsalternative zur Modelllösung: (8) …… ……					
	Summe Teilaufgabe d)	**8**			

	Summe insgesamt	**50**			

Festlegung der Gesamtnote (Bitte nur bei der letzten bearbeiteten Aufgabe ausfüllen.)

	Lösungsqualität			
	maximal erreichbare Punktzahl	EK	ZK	DK
Übertrag der Punktsumme aus der ersten bearbeiteten Aufgabe	**50**			
Übertrag der Punktsumme aus der zweiten bearbeiteten Aufgabe	**50**			
Übertrag der Punktsumme aus der dritten bearbeiteten Aufgabe	**50**			
Punktzahl der gesamten Prüfungsleistung	**150**			
aus der Punktsumme resultierende Note gemäß nachfolgender Tabelle				
Note ggf. unter Absenkung um bis zu zwei Notenpunkte gemäß § 13 Abs. 2 APO-GOSt				
Paraphe				

Berechnung der Endnote nach Anlage 4 der Abiturverfügung auf der Grundlage von § 34 APO-GOSt

Die Klausur wird abschließend mit der Note _____ (____ Punkte) bewertet.

Unterschrift, Datum:

Grundsätze für die Bewertung (Notenfindung)

Für die Zuordnung der Notenstufen zu den Punktzahlen ist folgende Tabelle zu verwenden:

Note	**Punkte**	**Erreichte Punktzahl**
sehr gut plus	15	150 – 143
sehr gut	14	142 – 135
sehr gut minus	13	134 – 128
gut plus	12	127 – 120
gut	11	119 – 113
gut minus	10	112 – 105
befriedigend plus	9	104 – 98
befriedigend	8	97 – 90
befriedigend minus	7	89 – 83
ausreichend plus	6	82 – 75
ausreichend	5	74 – 68
ausreichend minus	4	67 – 60
mangelhaft plus	3	59 – 50
mangelhaft	2	49 – 41
mangelhaft minus	1	40 – 30
ungenügend	0	29 – 0