Name: _____

Abiturprüfung 2016

Informatik, Leistungskurs

Aufgabenstellung:

Am Schiller-Gymnasium steht der diesjährige Projekttag unter dem Thema „Geschichten“. Eine Lehrerin bietet für die Schülerinnen und Schüler der fünften Klassen einen Workshop zum Thema „Geschichten schreiben“ an. Ziel dieses Workshops ist es, ein Geschichtenbuch zu schreiben, bei dem der Leser an Schlüsselstellen Entscheidungen treffen kann, die zu unterschiedlichen Geschichten führen. Schnell wird klar, dass das Geschichtenbuch mit zunehmender Entscheidungsfreiheit des Lesers immer komplexer wird.

Die Informatiklehrerin der Schule hat die Idee, mit ihrem Informatikkurs ein elektronisches Geschichtenbuch zu entwickeln, welches alle Geschichten elektronisch verwaltet. Sie schlägt vor, die einzelnen Abschnitte der Geschichten in einem Binärbaum zu speichern, so dass man jedem Kanten-Pfad innerhalb des Binärbaums genau eine einzelne Geschichte eindeutig zuordnen kann (siehe Aufgabenstellung a)). Abbildung 1 zeigt ein Beispiel eines in einem Binärbaum verwalteten Geschichtenbuchs mit verschiedenen Geschichten.

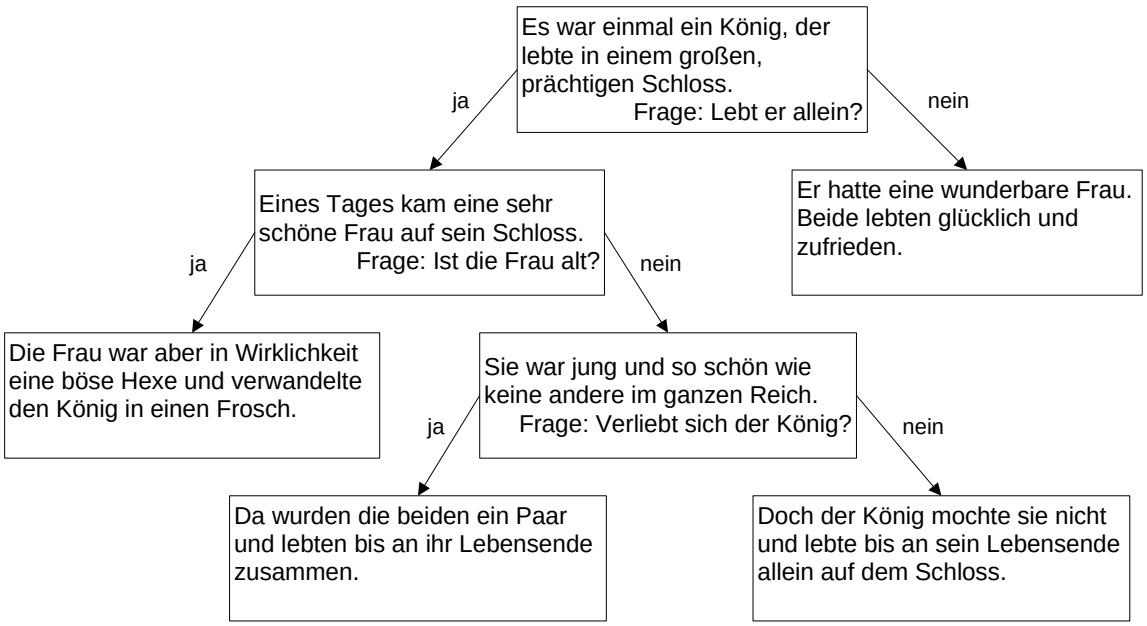

Abbildung 1: Beispiel eines in einem Binärbaum verwalteten Geschichtenbuchs

Name: _____

a) *Geben Sie den Kanten-Pfad, also die Kantenbeschriftung entlang des Pfades an, der zur folgenden Geschichte gehört.*

„Es war einmal ein König, der lebte auf einem großen, prächtigen Schloss. Eines Tages kam eine sehr schöne Frau auf sein Schloss. Sie war jung und so schön wie keine andere im ganzen Reich. Da wurden die beiden ein Paar und lebten bis an ihr Lebensende zusammen."

Erläutern Sie im Sachzusammenhang die Funktion eines Blattes eines Binärbaums, welcher die Abschnitte eines Geschichtenbuchs verwaltet.

(4 Punkte)

Ein Schüler des Informatikkurses entwickelt einen ersten Modellierungsvorschlag in Form eines Implementationsdiagramms, welcher ausschnittsweise in Abbildung 2 dargestellt ist.

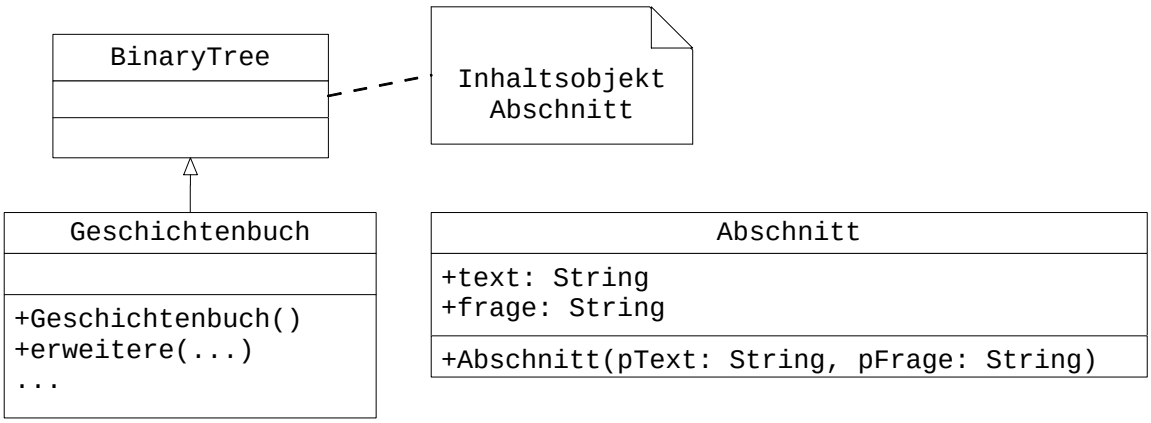

Abbildung 2: Modellierungsvorschlag

b) *Beurteilen Sie diesen Vorschlag bezüglich der Grundsätze der objektorientierten Modellierung.*

(6 Punkte)

Name: _____

Der Informatikkurs einigt sich schließlich auf eine Modellierung für die Verwaltung eines Geschichtenbuchs, deren Implementationsdiagramm ausschnittsweise in Abbildung 3 dargestellt ist.

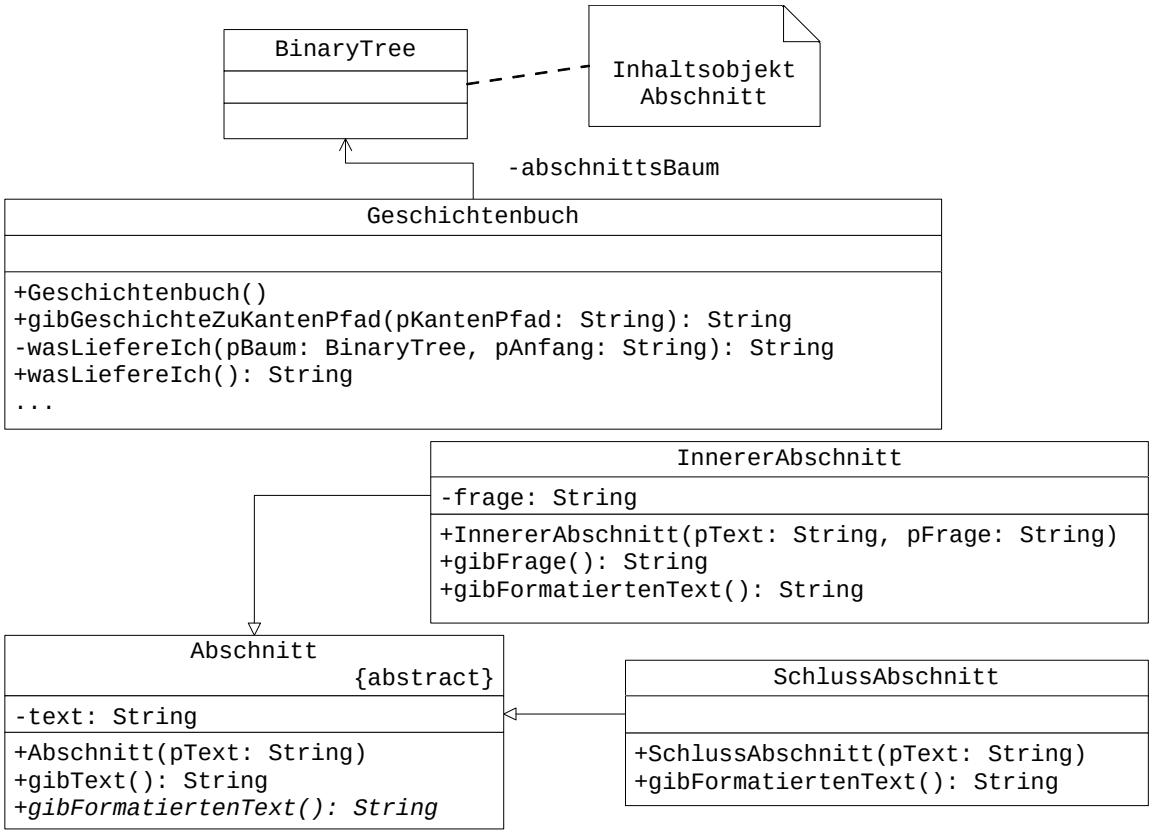

Abbildung 3: Auszug aus dem Implementationsdiagramm

c) *Implementieren Sie die Methode* `gibGeschichteZuKantenPfad` *der Klasse* `Geschichtenbuch` *entsprechend der Dokumentation im Anhang.*

Erläutern Sie Ihren Quelltext durch geeignete Kommentare.

(13 Punkte)

Name: _____

d) Gegeben seien die folgenden zwei Methoden der Klasse `Geschichtenbuch`.

```
 1  private String wasLiefereIch(BinaryTree pBaum,
                                              String pAnfang) {
 2    if (pBaum.isEmpty()) {
 3      return pAnfang;
 4    } else {
 5      Abschnitt a = (Abschnitt) pBaum.getObject();
 6      pAnfang = pAnfang + a.gibFormatiertenText();
 7      BinaryTree links = pBaum.getLeftTree();
 8      BinaryTree rechts = pBaum.getRightTree();
 9      if (links.isEmpty() && rechts.isEmpty()) {
10        return pAnfang;
11      } else {
12        return wasLiefereIch(links, pAnfang)
                       + " # " + wasLiefereIch(rechts, pAnfang);
13      }
14    }
15  }

16  public String wasLiefereIch() {
17    return wasLiefereIch(abschnittsBaum, "");
18  }
```

Vor dem Aufruf der ab Zeile 16 dargestellten öffentlichen Methode `wasLiefereIch` enthalte der in der Variablen `abschnittsbaum` referenzierte Binärbaum die in Abbildung 4 dargestellten Abschnitte eines Geschichtenbuchs.

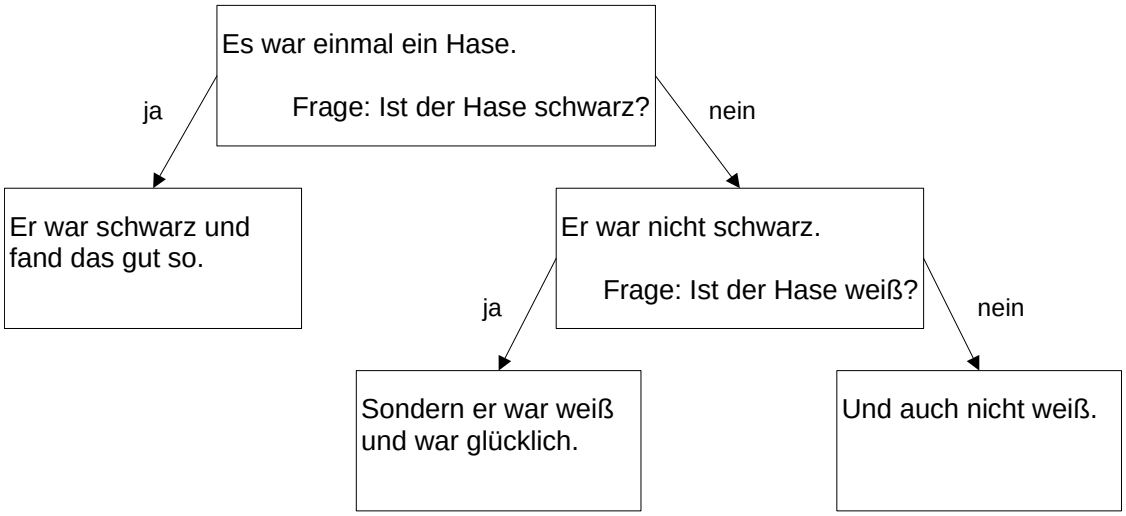

Abbildung 4: Beispiel eines in einem Binärbaum verwalteten Geschichtenbuchs

Name: _____

Ermitteln Sie jeweils den Wert, den die Variable `pAnfang` *in Zeile 10 bei Anwendung auf den Baum aus Abbildung 4 hat, sofern dieser Zweig der Fallunterscheidung erreicht wird.*

Geben Sie die Rückgabe des Methodenaufrufs in Zeile 17 an, wenn die Variable `abschnittsBaum` *den in Abbildung 4 dargestellten Baum referenziert.*

Erläutern Sie die Strategie der privaten Methode `wasLiefereIch` *der Zeilen 1 bis 15.*

Erläutern Sie den Zweck der beiden Methoden insgesamt im Sachzusammenhang.

(15 Punkte)

e) Die Inhalte des Baums sollen dauerhaft gespeichert werden. Ein Schüler schlägt vor, die Inhalte des Baums über eine Pre-Order-Traversierung in eine lineare Form zu überführen, um sie dann auf einem Datenträger speichern zu können.

Ein Beispiel einer linearisierten Form eines Binärbaums mit Abschnitten eines Geschichtenbuchs ist in Abbildung 5 angegeben, wobei die gespeicherten inneren Abschnitte mit `Innen_1`, `Innen_2`,… und die gespeicherten Schluss-Abschnitte mit `Schluss_1`, `Schluss_2`,… abgekürzt sind.

`Innen_1`	`Schluss_1`	`Innen_2`	`Innen_3`	`Schluss_2`	`Schluss_3`	`Schluss_4`

Abbildung 5: Pre-Order-Traversierung eines Binärbaums

Erläutern Sie, inwiefern hierbei die Unterscheidung zwischen inneren Abschnitten und Schluss-Abschnitten für die Rekonstruktion des ursprünglichen Binärbaums aus einer durch Pre-Order-Traversierung erzeugten Linearisierung von Bedeutung ist.

Entwickeln Sie einen Algorithmus zur Rekonstruktion des Binärbaums eines Geschichtenbuchs, der auf die oben dargestellte Weise linearisiert wurde.

Hinweis: Eine Implementierung ist nicht erforderlich!

(12 Punkte)

Zugelassene Hilfsmittel:

- Wörterbuch zur deutschen Rechtschreibung
- Taschenrechner (wissenschaftlicher Taschenrechner ohne oder mit Grafikfähigkeit/ CAS-Taschenrechner)

Name: _____

Anhang

Die Klasse `Geschichtenbuch`

Objekte der Klasse `Geschichtenbuch` verwalten Abschnitte eines Geschichtenbuchs entsprechend der Aufgabenstellung in Form eines Binärbaums.

Auszug aus der Dokumentation der Klasse `Geschichtenbuch`

Die Dokumentation der Methoden, welche die Entwicklung und den Aufbau eines Geschichtenbuchs ermöglichen, wurde zugunsten der Übersichtlichkeit nicht aufgeführt.

Konstruktor **`Geschichtenbuch()`**

Ein Objekt der Klasse wird erstellt. Der Binärbaum, der die Abschnitte des Geschichtenbuchs verwaltet, ist leer, d. h., das Geschichtenbuch enthält noch keine Abschnitte.

Anfrage **`String gibGeschichteZuKantenPfad(String pKantenPfad)`**

Die Anfrage ermittelt die Geschichte ohne Fragetexte zum Kanten-Pfad, der in der Zeichenkette `pKantenPfad` gespeichert ist. Die Geschichte setzt sich dabei aus den formatierten Abschnittstexten zusammen, welche entlang des Kanten-Pfades liegen.
Die einzelnen Kantenbeschriftungen des Kanten-Pfades sind jeweils mit dem Trennzeichen `"-"` voneinander getrennt, so ist z. B. `"ja-nein-nein"` ein gültiger Kanten-Pfad.
Die Anfrage besitzt keine Fehlerbehandlung, d. h., der Kanten-Pfad muss gültig sein und im Binärbaum existieren.

Anfrage **`String wasLiefereIch()`**

Die Funktion dieser Methode wird in Aufgabenteil d) analysiert.

Name: _____

Die abstrakte Klasse `Abschnitt`

Objekte dieses Typs verwalten den Geschichtentext eines einzelnen Abschnitts des Geschichtenbuchs.

Dokumentation der abstrakten Klasse `Abschnitt`

Konstruktor **`Abschnitt(String pText)`**
Der Geschichtentext `pText` wird übernommen.

Anfrage **`String gibText()`**
Die Anfrage liefert den unformatierten Geschichtentext des Objekts ohne eventuell abschließende Textergänzungen.

Anfrage ***`String gibFormatiertenText()`***
Diese Methode ist hier noch abstrakt definiert, weshalb die Dokumentation dieser Methode in den abgeleiteten Klassen erfolgt.

Die Klasse `SchlussAbschnitt`

Die Klasse ist von der Klasse `Abschnitt` abgeleitet. Objekte dieser Klasse verwalten den Geschichtentext eines einzelnen Schluss-Abschnitts des Geschichtenbuchs. Das heißt, Objekte dieser Klasse bilden den Abschluss einer einzelnen Geschichte des Geschichtenbuchs.

Dokumentation der Klasse `SchlussAbschnitt`

Konstruktor **`SchlussAbschnitt(String pText)`**
Ein neuer Schluss-Abschnitt mit dem Geschichtentext `pText` wird erzeugt.

Anfrage **`String gibFormatiertenText()`**
Die Anfrage liefert den formatierten Textinhalt des Abschnitts. Dieser besteht aus dem unformatierten Geschichtentext gefolgt von der abschließenden Zeichenkette `" ENDE."`

Name: _____

Die Klasse `InnererAbschnitt`

Diese Klasse ist von der Klasse `Abschnitt` abgeleitet. Objekte dieser Klasse verwalten den Geschichtentext und eine Fragestellung eines einzelnen inneren Abschnitts des Geschichtenbuchs. Das heißt, Objekte dieser Klasse befinden sich innerhalb oder am Anfang einer einzelnen Geschichte des Geschichtenbuchs.

Dokumentation der Klasse `InnererAbschnitt`

Konstruktor **`InnererAbschnitt(String pText, String pFrage)`**
Ein neuer innerer Abschnitt mit dem Geschichtentext `pText` und der Fragestellung `pFrage` wird erzeugt.

Anfrage **`String gibFrage()`**
Die Anfrage liefert die Fragestellung des Objekts.

Anfrage **`String gibFormatiertenText()`**
Die Anfrage liefert den formatierten Textinhalt des Abschnitts. Dieser besteht aus dem unformatierten Geschichtentext gefolgt von der abschließenden Zeichenkette `" "`. Die Fragestellung des Objekts bleibt in dieser Anfrage unberücksichtigt.

Name: _____

Die Klasse BinaryTree

Mithilfe der Klasse **`BinaryTree`** können beliebig viele Inhaltsobjekte 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 Klasse `BinaryTree` sind.

Dokumentation der Klasse BinaryTree

Konstruktor **`BinaryTree()`**

Nach dem Aufruf des Konstruktors existiert ein leerer Binärbaum.

Konstruktor **`BinaryTree(Object pObject)`**

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

Konstruktor **`BinaryTree(Object pObject, BinaryTree pLeftTree, BinaryTree pRightTree)`**

Wenn der Parameter `pObject` ungleich `null` ist, wird ein Binärbaum mit `pObject` als Inhaltsobjekt und den beiden Teilbäume `pLeftTree` und `pRightTree` erzeugt. Sind `pLeftTree` oder `pRightTree` gleich `null`, wird der entsprechende Teilbaum als leerer Binärbaum eingefügt. Wenn der Parameter `pObject` 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 setObject(Object pObject)`**

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

Anfrage **`Object getObject()`**

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

Name: _____

Auftrag **`void setLeftTree(BinaryTree 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 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 getLeftTree()`**
Diese Anfrage liefert den linken Teilbaum des Binärbaumes. Der Binärbaum ändert sich nicht. Wenn der Binärbaum leer ist, wird `null` zurückgegeben.

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

Dokumentation ausgewählter Methoden der Klasse `String`

Anfrage **`int length()`**
Diese Anfrage liefert die Länge des Strings.

Anfrage **`String substring(int pBegin)`**
Diese Anfrage liefert den Teilstring beginnend mit der übergebenen Position `pBegin` bis zum Ende des Strings.

Anfrage **`String substring(int pBegin, int pEnd)`**
Diese Anfrage liefert den Teilstring beginnend mit der übergebenen Position pBegin bis eine Position vor `pEnd`.

Anfrage **`boolean equals(String pAnotherString)`**
Diese Anfrage liefert den Wert `true`, wenn die Zeichenketten lexikographisch identisch sind, andernfalls den Wert `false`.

Anfrage **`char charAt(int pIndex)`**
Diese Anfrage liefert den Buchstaben an der Position `pIndex`.

Unterlagen für die Lehrkraft

Abiturprüfung 2016

Informatik, Leistungskurs

1. Aufgabenart

Aufgabenart	Modellierung einer Problemstellung, Entwurf und Implementation von Algorithmen
Syntaxvariante	Java

2. Aufgabenstellung[1]

siehe Prüfungsaufgabe

3. Materialgrundlage

- entfällt

4. Bezüge zu den Vorgaben 2016

1. *Inhaltliche Schwerpunkte*
   Objektorientiertes Modellieren und Implementieren von kontextbezogenen Anwendungen
   - Konzepte des objektorientierten Modellierens
   - Algorithmen und Datenstrukturen
     - Baumstrukturen mit den Akzenten
       Binärbaum (Anwendung der Standardoperationen; Traversierungsalgorithmen)

2. *Medien/Materialien*
   - entfällt

5. Zugelassene Hilfsmittel

- Wörterbuch zur deutschen Rechtschreibung
- Taschenrechner (wissenschaftlicher Taschenrechner ohne oder mit Grafikfähigkeit/ CAS-Taschenrechner)

[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 Kantenbeschriftung entlang des Pfades zur angegebenen Geschichte lautet `"ja-nein-ja"`.

Ein Blatt eines Binärbaums, in dem die Abschnitte eines Geschichtenbuchs verwaltet werden, hat die Funktion, den Text eines Schluss-Abschnitts zu verwalten, d. h., der verwaltete Abschnitt beendet eine eindeutige Geschichte.

Teilaufgabe b)

Die Modellierung entspricht unter mehreren Gesichtspunkten nicht den Grundsätzen der objektorientierten Modellierung.

Die Klasse `Abschnitt` ist so modelliert, dass die Attribute `text` und `frage` beide öffentlich zugänglich sind. Ein Grundsatz der objektorientierten Modellierung ist aber der Grundsatz der versteckten Information, welcher hier verletzt ist. Die Attribute sollten also privat sein und der Zugriff sollte mit entsprechenden Auslese- und Setz-Methoden erfolgen. Darüber hinaus ist es auch nicht sinnvoll, für Abschnitte, welche am Ende einer Geschichte stehen, ein Attribut `frage` zu verwalten.

Außerdem ist die Modellierung mit einer Klasse `Geschichtenbuch` als Vererbung von der Klasse `BinaryTree` nicht sinnvoll, da so alle öffentlichen Methoden der Klasse `BinaryTree` auch den Objekten der Klasse `Geschichtenbuch` zur Verfügung ständen. Diese sollten aber für die Klasse `Geschichtenbuch` versteckt sein, da sonst von außen durch Anwendung der `BinaryTree`-Operationen eine Veränderung der logischen Struktur der Daten eines Geschichtenbaums möglich wäre.

Teilaufgabe c)

Eine mögliche Implementierung mit entsprechender Kommentierung lautet wie folgt:

```
public String gibGeschichteZuKantenPfad(String pKantenPfad) {
  // Beginne an der Wurzel des Geschichtenbaums.
  BinaryTree b = abschnittsBaum;
  // Beginne die Geschichte mit dem ersten Abschnitt.
  Abschnitt abschnitt = (Abschnitt) b.getObject();
  String geschichte = abschnitt.gibFormatiertenText();
  // Wiederhole, solange der Pfad noch nicht abgearbeitet ist.
  while (!pKantenPfad.equals("")) {
    // Je nach Pfadanfang waehle den linken oder rechten Pfad.
    if (pKantenPfad.substring(0, 2).equals("ja")) {
      b = b.getLeftTree();
      pKantenPfad = pKantenPfad.substring(2);
    } else {
      b = b.getRightTree();
      pKantenPfad = pKantenPfad.substring(4);
    }
    // Danach das eventuelle Trennzeichen ("-") entfernen.
    if (!pKantenPfad.equals("")) {
      pKantenPfad = pKantenPfad.substring(1);
    }
    // Erweitere die Geschichte um den aktuellen Abschnitt.
    abschnitt = (Abschnitt) b.getObject();
    geschichte = geschichte + abschnitt.gibFormatiertenText();
  }
  // Gib die gesamte Geschichte zurueck.
  return geschichte;
}
```

Hinweis:
Die angegebene Lösung prüft nicht, ob der Kanten-Pfad gültig ist, d. h. wirklich zu genau einem Geschichten-Ende führt, da die Dokumentation dies ausdrücklich nicht fordert. Lösungen, welche diese Fehlerüberprüfung enthalten, sollen natürlich als gleichwertig erachtet werden.

Teilaufgabe d)

Die Variable `pAnfang` hat in den Rekursionsankern (entspricht jeweils einem Blatt des Baums) die folgenden Belegungen:

Variable / Rek.-Anker	`pAnfang`
1. Blatt	Es war einmal ein Hase. Er war schwarz und fand das gut so. ENDE.
2. Blatt	Es war einmal ein Hase. Er war nicht schwarz. Sondern er war weiß und war glücklich. ENDE.
3. Blatt	Es war einmal ein Hase. Er war nicht schwarz. Und auch nicht weiß. ENDE.

Die Rückgabe der Methode lautet:
Es war einmal ein Hase. Er war schwarz und fand das gut so. ENDE. # Es war einmal ein Hase. Er war nicht schwarz. Sondern er war weiß und war glücklich. ENDE. # Es war einmal ein Hase. Er war nicht schwarz. Und auch nicht weiß. ENDE.

Zu Beginn (Zeile 2) wird geprüft, ob der Baum evtl. leer ist. In diesem Fall würde die Methode den bisher zusammengesetzten Anfang einer Geschichte (`pAnfang`) zurückliefern, welcher im konkreten Fall aus einer leeren Zeichenkette bestünde. Andernfalls (Zeile 4 bis 14) wird der aktuelle Abschnitt an den bisherigen Geschichtenanfang (`pAnfang`) angehängt (Zeilen 5 und 6). Sollte der aktuelle Abschnitt ein Blatt sein (Prüfung in Zeile 9), so wird die bis hierher zusammengestellte Geschichte (`pAnfang`) zurückgeliefert (Rekursionsanker). Andernfalls wird der bisherige Geschichtenanfang einmal mit dem linken und einmal mit dem rechten Teilbaum fortgeführt und beide Geschichten werden durch ein `" # "` getrennt voneinander zusammengefügt und zurückgeliefert (Zeile 12). Die Variable `pAnfang` speichert also stets den Verlauf einer einzelnen Geschichte.

Die Methoden liefern eine Zeichenkette, in der alle Geschichten des Geschichtenbuchs von links nach rechts durch `" # "` getrennt voneinander aufgeführt sind. Dabei übernimmt die öffentliche Methode lediglich die Funktion, die rekursive, private Methode mit dem intern gespeicherten Binärbaum zu starten und deren Funktionsergebnis wieder zurückzugeben.

Teilaufgabe e)

Die durch Pre-Order-Traversierung erzeugte Linearisierung beinhaltet zunächst keine Informationen über die Position der einzelnen Abschnitte innerhalb des Baums. Durch die Unterscheidung eines inneren Abschnitts von einem Schluss-Abschnitt und die spezielle Form eines Geschichtenbaums (Jeder innere Knoten hat zwei nichtleere Teilbäume.) ist die Rekonstruktion der Position der einzelnen Abschnitte und damit auch die Rekonstruktion des gesamten Geschichtenbaums möglich.

Ein Algorithmus könnte wie folgt aussehen.

- Lies den aktuellen (am Anfang ersten) Datensatz der Pre-Order-traversierten Linearisierung und wechsle zum nächsten Datensatz.
- Ist der aktuelle Datensatz ein Schluss-Abschnitt,
  - erzeuge einen resultierenden Baum, welcher nur aus dem einen Datensatz besteht, also ein Blatt ist.
- Sonst
  - erzeuge mit dem aktuellen Datensatz einen neuen resultierenden Baum,
  - erzeuge mithilfe der Pre-Order-Darstellung rekursiv einen linken Teilbaum,
  - erzeuge mithilfe der Pre-Order-Darstellung rekursiv einen rechten Teilbaum,
  - hänge die rekursiv erstellten Teilbäume links und rechts an den resultierenden Baum an.
- Gib den resultierenden Baum zurück.

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	gibt den zur Geschichte gehörenden Kanten-Pfad an.	2			
2	erläutert die Funktion eines Blattes im Sachzusammenhang.	2			
Sachlich richtige Lösungsalternative zur Modelllösung: (4) …… ……					
	Summe Teilaufgabe a)	**4**			

Teilaufgabe b)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	beurteilt die Modellierung der Klasse `Geschichtenbuch`.	3			
2	beurteilt die Modellierung der Klasse `Abschnitt`.	3			
Sachlich richtige Lösungsalternative zur Modelllösung: (6) …… ……					
	Summe Teilaufgabe b)	**6**			

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

Teilaufgabe c)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	implementiert die Methode `gibGeschichteZuKanten-Pfad` entsprechend der Dokumentation.	9			
2	erläutert den Quelltext durch geeignete Kommentare.	4			
Sachlich richtige Lösungsalternative zur Modelllösung: (13) …… ……					
	Summe Teilaufgabe c)	**13**			

Teilaufgabe d)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	ermittelt jeweils den Wert, den die Variable `pAnfang` hat.	4			
2	gibt die Rückgabe der Methode an.	2			
3	erläutert die Strategie der privaten Methode.	6			
4	erläutert den Zweck der Methoden im Sachzusammenhang.	3			
Sachlich richtige Lösungsalternative zur Modelllösung: (15) …… ……					
	Summe Teilaufgabe d)	**15**			

Teilaufgabe e)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	erläutert, inwiefern die Unterscheidung zwischen inneren Abschnitten und Schluss-Abschnitten von Bedeutung ist.	4			
2	entwickelt einen Algorithmus zur Rekonstruktion aus einer durch Pre-Order-Traversierung erzeugten Linearisierung.	8			
Sachlich richtige Lösungsalternative zur Modelllösung: (12) …… ……					
	Summe Teilaufgabe e)	**12**			

	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 – 40
mangelhaft minus	1	39 – 30
ungenügend	0	29 – 0