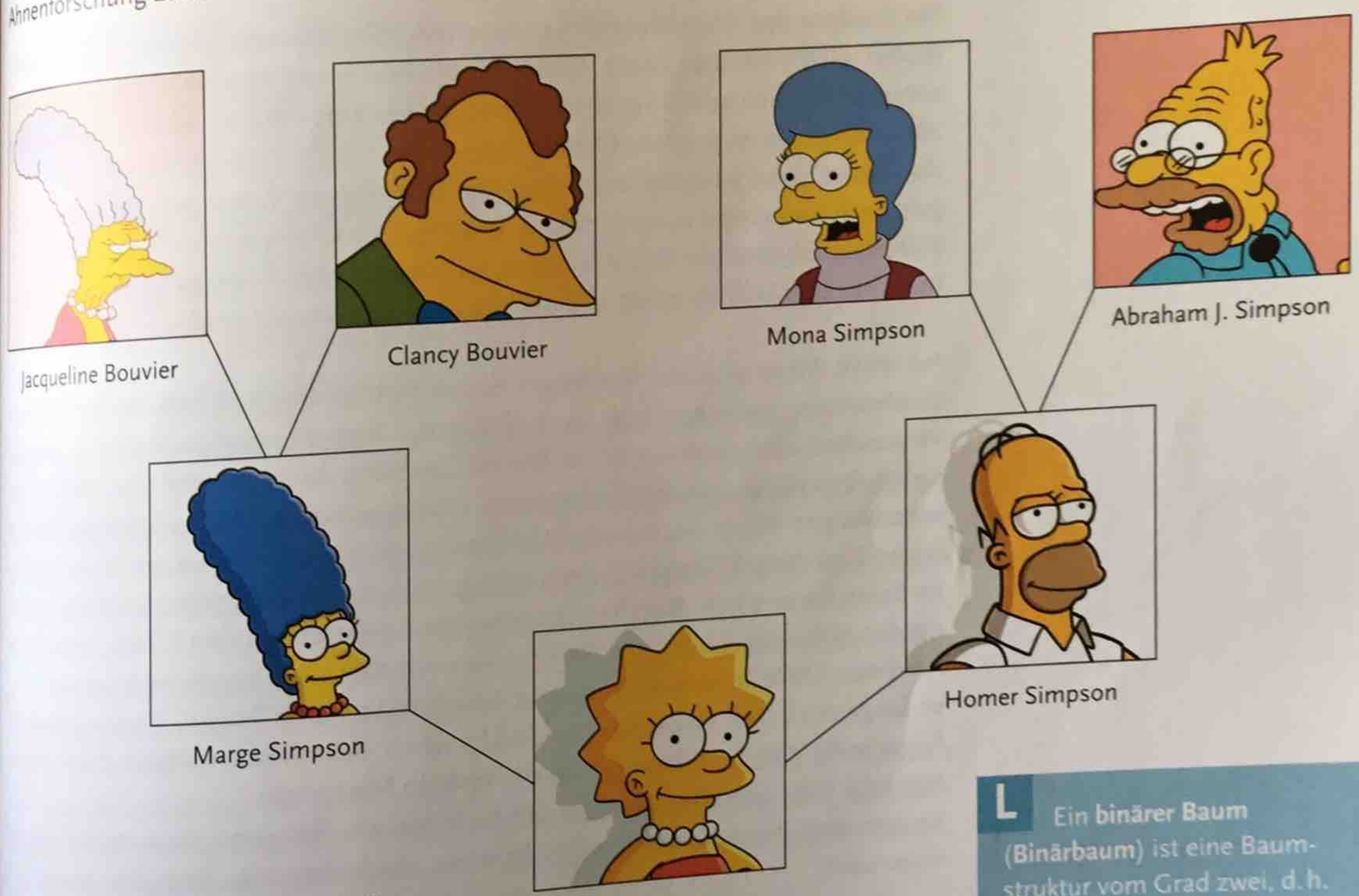
5.2 Zwei Nachfolger sind genug! – Binäre Bäume

Lineare Datenstrukturen organisieren gespeicherte Daten so, dass das Suchen, Einfügen und Löschen geplant durchgeführt werden kann. Dies scheint bei Baumstrukturen auf den ersten Blick erst einmal nicht der Fall zu sein. Denn welchen Nachfolgeknoten wählt man, wenn man in der Struktur weitergehen möchte? In einer Baumstruktur kann ein Knoten schließlich durchaus mehrere Nachfolger besitzen.

Eine besondere Baumstruktur stellen binäre Bäume dar, deren Knoten immer höchstens zwei Nachfolger haben dürfen. Ein solcher binärer Baum kommt z. B. bei der Ahnenforschung zum Einsatz.



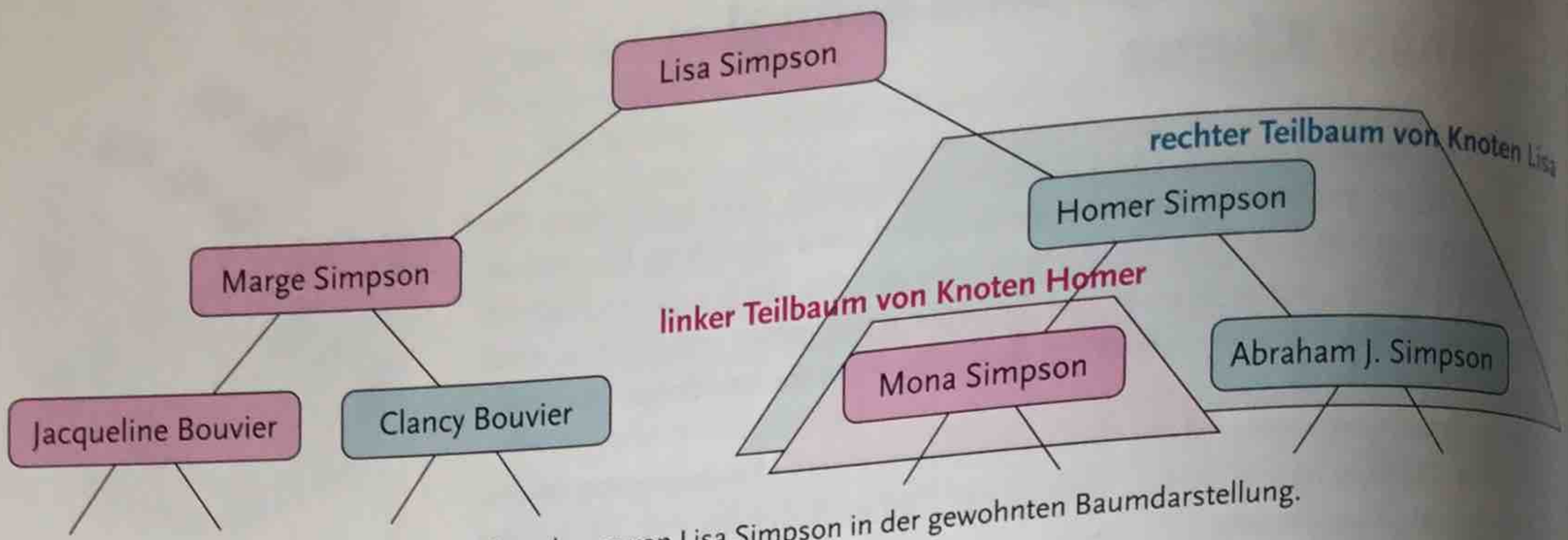
Der Ahnenbaum von Lisa Simpson stellt ihre direkten Vorfahren übersichtlich dar.

Ein Ahnenbaum bildet die direkten Vorfahren einer Person ab. Ausgehend von *Lisa Simpson* werden zunächst ihre Eltern, dann deren Eltern, also ihre Großeltern usw. dargestellt. Da in der Regel nur die leiblichen Eltern aufgeführt werden, hat somit jeder Knoten zwei Nachfolger: eine leibliche Mutter und einen leiblichen Vater.

Es ist üblich, bei der Darstellung von Baumstrukturen die Wurzel oben zu platzieren. Für den Ahnenbaum bedeutet dies, dass *Lisa Simpson* ganz oben steht, gefolgt von ihren Eltern usw. Zudem werden die Knoten aller weiblichen Ahnen rosa und die der männlichen Vorfahren blau gefärbt.

L Ein **binärer Baum** (**Binärbaum**) ist eine Baumstruktur vom Grad zwei, d. h. jeder Knoten kann höchstens zwei Nachfolger haben.
Dies lässt sich auch rekursiv definieren:
Ein binärer Baum ist entweder leer, oder er besteht aus einer Wurzel sowie einem linken und einem rechten Teilbaum. Bei diesen Teilbäumen handelt es sich wiederum um binäre Bäume.

→ Aufgabe 1



Ahnenbaum von Lisa Simpson in der gewohnten Baumdarstellung.

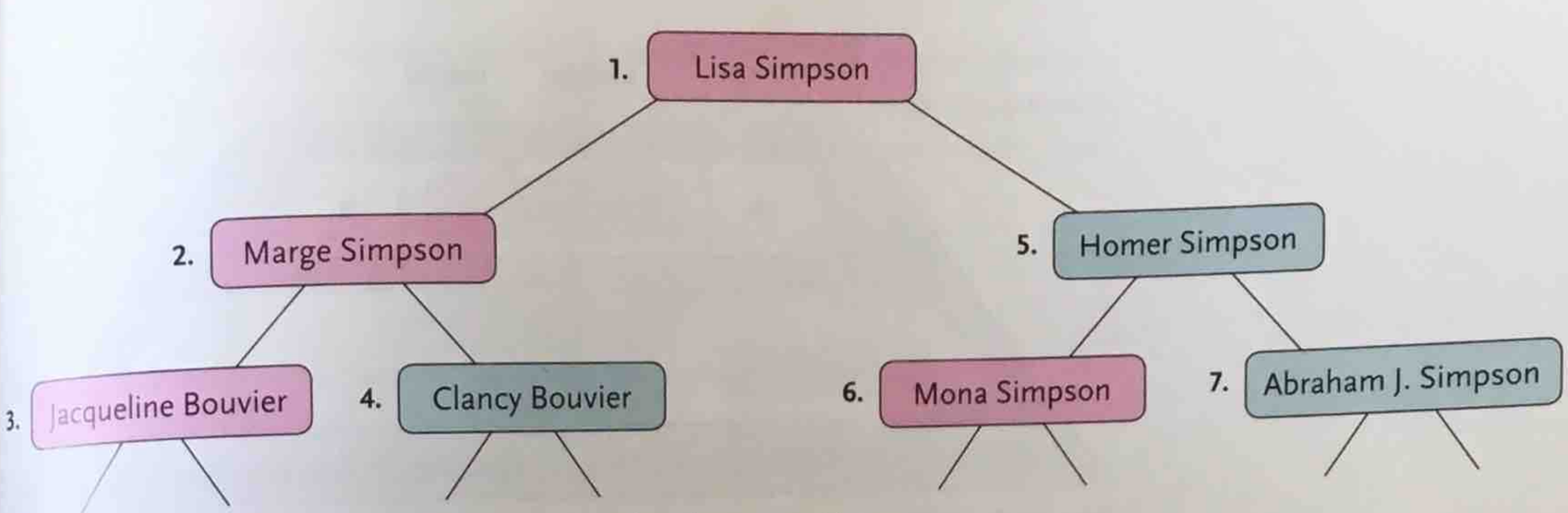
Die Struktur des Ahnenbaums ist ungemein hilfreich, wenn man z. B. Lisas Großmutter väterlicherseits sucht. Diese Information muss nicht in einem der Knoten, entweder bei Lisa oder ihrer Großmutter, gespeichert werden, sondern sie lässt sich allein durch die vorgegebene Struktur ermitteln.
Da klar ist, dass ausgehend von Lisa die Mutter ihres Vaters gesucht werden soll, geht man also zuerst zum rechten Teilbaum und von dort aus zum linken Teilbaum. Wären die Objekte alle hintereinander, also linear angeordnet, so wären in der Regel mehr als zwei Schritte nötig, um die gesuchte Person aufzufinden.

Auf diese Weise lässt sich der Baum jedoch nicht nach einem bestimmten Element durchsuchen. Sucht man z. B. nach *Abraham J. Simpson*, ohne zu wissen, in welcher Verwandtschaftsbeziehung er zur Wurzel *Lisa* steht, so muss wohl oder übel der gesamte Ahnenbaum durchforstet werden. Man benötigt somit eine Strategie, um den vollständigen Baum nach einem Element zu durchsuchen. Dies erscheint auf den ersten Blick ein schwieriges Unterfangen zu sein, denn ist man erst einmal einen Pfad im Baum bis zu einem Blatt hinuntergewandert, dann gibt es kein Zurück mehr, da ein Knoten in der Regel nur seine Nachfolger, nicht aber seine Vorgänger speichert.
An dieser Stelle kommt die rekursive Baumstruktur zum Tragen, die es ermöglicht, so lange die linken Nachfolger zu durchsuchen, bis es keine weiteren linken Nachfolger mehr gibt, und dann die jeweils rechten Nachfolger.
Wie folgt lässt sich eine rekursive Methode *besuche* definieren, die systematisch alle Knoten des übergebenen binären Baumes abarbeitet. Man sagt: Der Baum wird traversiert.

```
besuche (binBaum b)
  bearbeite die Wurzel von b
  falls b einen nichtleeren linken Teilbaum hat dann
    besuche(linker Teilbaum von b)
  ende falls
  falls b einen nichtleeren rechten Teilbaum hat dann
    besuche(rechter Teilbaum von b)
  ende falls
```

Rekursive Aufrufe von *besuche* auf den Teilbäumen

Diese Strategie, einen binären Baum zu traversieren, bearbeitet also zunächst immer die Wurzel, dann den linken Teilbaum und schließlich den rechten Teilbaum des gegebenen Baumes. Dieses Verfahren nennt sich **Preorder-Traversierung** eines Binärbaums.



Die Reihenfolge der Preorder-Traversierung für den Ahnenbaum von Lisa.

besuche(Lisa Simpson *)*
besuche(Marge Simpson *)*
besuche(Jacqueline Bouvier *)*
besuche(Clancy Bouvier *)*
besuche(Homer Simpson *)*
besuche(Mona Simpson *)*
besuche(Abraham J. Simpson *)*

Aufrufstruktur der rekursiven besuche-Methode bei der preorder-Traversierung.

Neben der Preorder-Traversierung gibt es noch zwei weitere Varianten, alle Knoten eines binären Baumes systematisch zu besuchen:

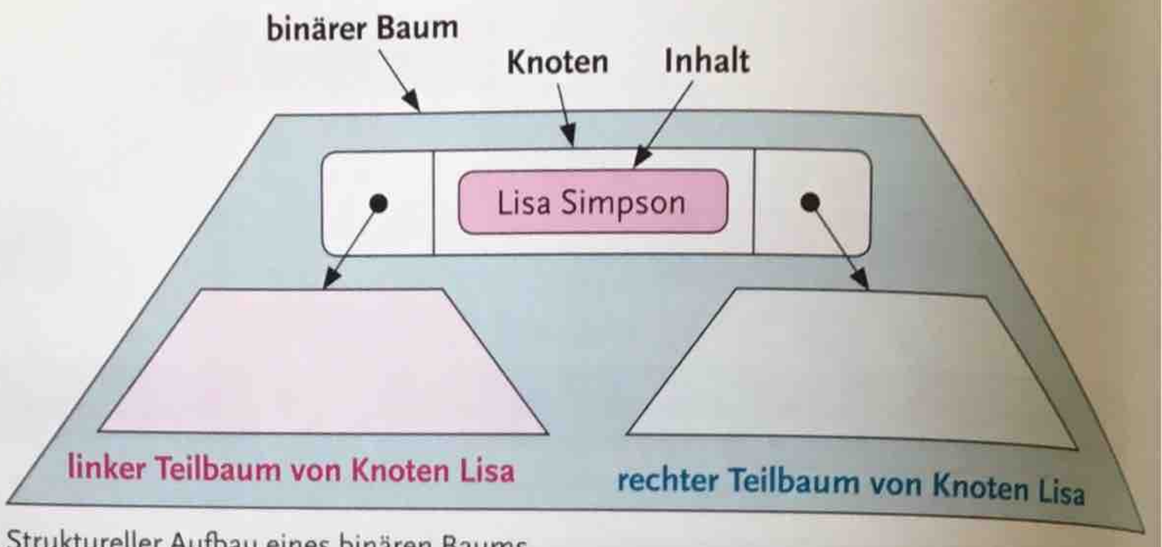
- die **Inorder-Traversierung**, bei der zunächst der linke Teilbaum, dann die Wurzel und schließlich der rechte Teilbaum besucht wird
- die **Postorder-Traversierung**, bei der man zunächst den linken Teilbaum, dann den rechten und schließlich die Wurzel abarbeitet.

Die Bezeichnungen der Verfahren leiten sich aus dem Zeitpunkt ab, zu dem die Wurzel des Baums bearbeitet wird. Dies kann vor *(pre)*, nach *(post)* oder zwischen *(in)* den Durchläufen des linken und des rechten Teilbaums sein. In jedem Fall wird allerdings der linke Teilbaum vor dem rechten bearbeitet.

→ Aufgabe 2 a), b), 3 a)

Modellierung des binären Baums

Ebenso wie die linearen Datenstrukturen Schlange, Stapel und Liste wird der binäre Baum als generische Klasse umgesetzt, d. h. dass beliebige Objekttypen durch die Klasse *BinaryTree<ContentType>* verwaltet werden können.
Der binäre Baum wird mithilfe von Knoten modelliert. Jeder Binärbaum hat einen Knoten, seine Wurzel. Dieser Knoten speichert einen Inhalt und verweist auf einen linken und einen rechten Teilbaum. Somit muss auch die Klasse *Knoten* generisch sein, da sie beliebige Inhalte speichern können soll.



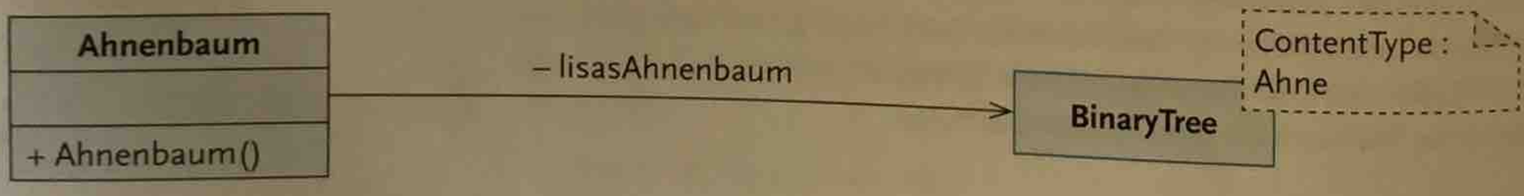
Struktureller Aufbau eines binären Baums

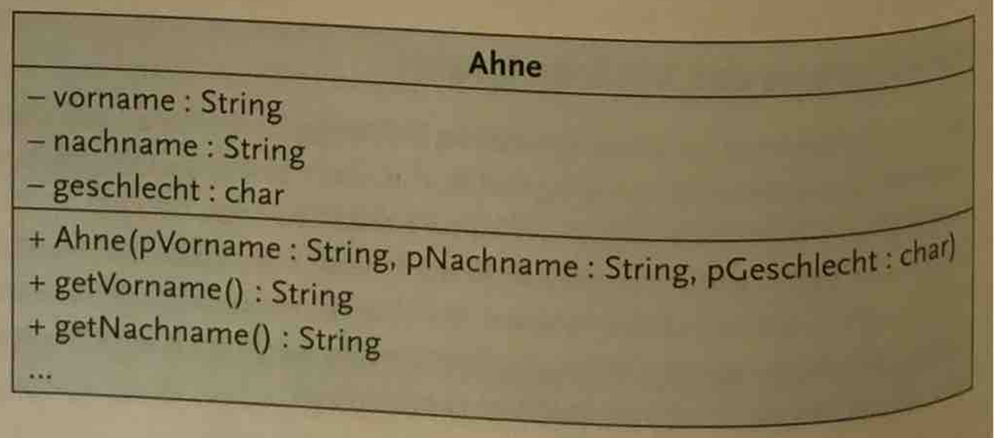
Die Klasse *BinaryTree<ContentType>* verfügt über drei verschiedene Konstruktoren, sodass ein leerer Binärbaum, ein Blatt oder ein innerer Knoten erzeugt werden können.
Die Knoten des Binärbaums werden durch die Klasse *BTNode<CT>* definiert, die wie auch bei den linearen Datenstrukturen als innere Klasse der Klasse *Binary Tree<Content Type>* modelliert ist.

H Die Dokumentation der vorgestellten Klassen findet sich im Anhang. Ihre Implementation steht als Download bereit unter: www.schulentwicklung.nrw.de/Lehrplaene/Lehrplannavigator-s-ii/gymnasiale-Oberstufe/informatik/hinweise-und-beispiele/

Damit Anwendungen sinnvoll mit der Datenstruktur Binärbaum arbeiten können, sollte die Klasse *BinaryTree* noch weitere Methoden anbieten:
- eine Anfrage, ob der Binärbaum leer ist *(isEmpty() : boolean)*
- ein Auftrag zum Einfügen neuer Inhalte *(setContent(pContent : ContentType) : void)*
- eine Anfrage nach dem Inhaltsobjekt des Binärbaums *(getContent() : ContentType)*
- Aufträge zum Anhängen neuer Teilbäume *(setLeftTree(pTree : BinaryTree <Content Type>), setRightTree(pTree : BinaryTree<ContentType>))*
- Anfragen nach den jeweiligen Teilbäumen des Binärbaums *(getLeftTree() : BinaryTree<ContentType>, getRightTree() : BinaryTree<Content Type>)*

Der Ahnenbaum kann den Binärbaum dann wie folgt verwenden.





Soll ein neuer Binärbaum angelegt werden, so muss dieser von unten nach oben aufgebaut werden. Beginnend mit den Blättern (z. B. mit dem Inhalt „Jacqueline Bouvier") arbeitet man sich also hoch bis zur Wurzel des Baumes (Inhalt „Lisa Simpson").

```
BinaryTree<Ahne> b1 = new BinaryTree<Ahne>( Jacqueline Bouvier );
BinaryTree<Ahne> b2 = new BinaryTree<Ahne>( Clancy Bouvier );
BinaryTree<Ahne> b3 = new BinaryTree<Ahne>( Marge Simpson , b1, b2);
BinaryTree<Ahne> b4 = new BinaryTree<Ahne>( Mona Simpson );
BinaryTree<Ahne> b5 = new BinaryTree<Ahne>( Abraham J. Simpson );
Bin  Tree<Ahne> b6 = new BinaryTree<Ahne>( Homer Simpson , b4, b5);
lis  nenbaum = new BinaryTree<Ahne>( Lisa Simpson , b3, b6);
```

Aufgaben

1. Ein Ahnenbaum stellt die direkten Vorfahren einer Person dar. In einem Stammbaum hingegen werden alle Nachfahren einer Person oder eines Elternpaares aufgeführt.

   a) Erstellen Sie zunächst Ihren eigenen Ahnenbaum und anschließend den Stammbaum eines Vorfahren, z. B. Ihrer Großeltern. Beschreiben Sie Unterschiede und Gemeinsamkeiten der beiden Strukturen mithilfe entsprechender Fachbegriffe.

   b) Geben Sie die Aufrufe der unterschiedlichen Konstruktoren der Klasse *BinaryTree<ContentType>* an, die nötig sind, um Ihren Ahnenbaum aus a) zu erzeugen.

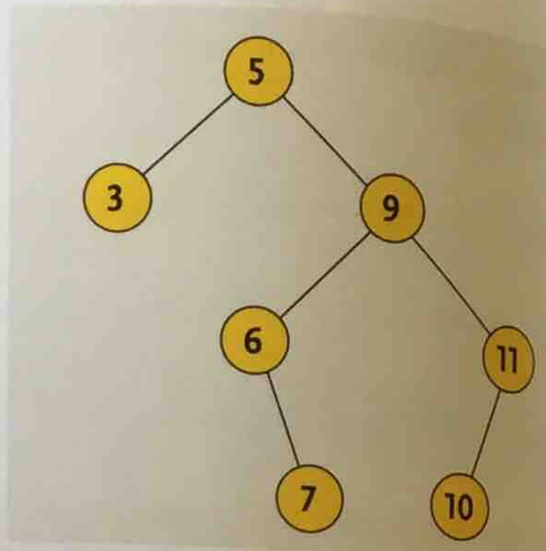
2. a) Implementieren Sie Lisas Ahnenbaum mit Hilfe der generischen Klasse *BinaryTree<ContentType>*.

   b) Implementieren Sie eine Methode *preorderAusgabe()*, die die Namen aller Ahnen in Lisas Ahnenbaum in der Reihenfolge der Preorder-Traversierung auf dem Bildschirm ausgibt.

   Tipp: Schachteln Sie den rekursiven Aufruf in eine weitere Methode, die den jeweiligen Teilbaum als Parameter erhält.

```
public void preorderAusgabe()
{
    besuche (lisasAhnenbaum);
}
private void besuche(BinaryTree<Ahne> b)
{
...
```

3. Gegeben ist der folgende binäre Baum:



a) Bestimmen Sie jeweils die Durchlaufreihenfolge für die drei Traversierungsvarianten Preorder, Inorder und Postorder.

b) Gegeben sind die Preorder- und die Inordertraversierung eines binären Baumes.
Preorder: 1, 2, 4, 3, 5, 6, 7
Inorder: 4, 2, 1, 6, 5, 7, 3
Rekonstruieren Sie den Aufbau des binären Baums und stellen Sie ihn mit Knoten und Kanten dar.

c) Implementieren Sie rekursiv die drei Traversierungsmethoden. Das Bearbeiten eines Knotens soll dabei darin bestehen, dass sein Inhalt auf dem Bildschirm ausgegeben wird.

d) Gegeben ist die folgende Methode:

```
public void wasTueIch(BinaryTree<Integer> b)
{
  Stack<BinaryTree<Integer>> stapel =
                          new Stack<BinaryTree<Integer>>();
  while (!stapel.isEmpty() || !b.isEmpty())
  {
    if (!b.isEmpty())
    {
      stapel.push(b);
      b = b.getLeftTree();
    }
    else
    {
      b = stapel.top();
      stapel.pop();
      System.out.println(b.getContent());
      b = b.getRightTree();
    }
  }
}
```

Analysieren Sie die Arbeitsweise der Methode *wasTuelch*, indem Sie die Belegungen des Stapels sowie die Ausgabe auf dem Bildschirm für den abgebildeten Baum notieren. Erläutern Sie, was die Methode leistet.

e) Entwickeln Sie eine Strategie für eine Methode, die die Anzahl aller Knoten eines binären Baums bestimmt. Implementieren Sie Ihre Methode.