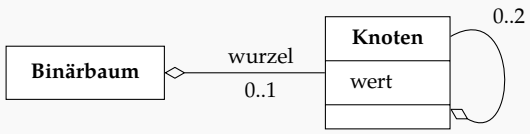
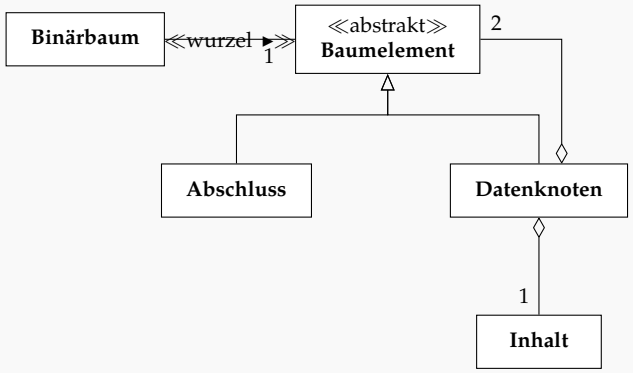
Klassendiagramm und Implementierung

(a) Erstellen Sie ein Klassendiagramm für einen Binärbaum.

einfacher Binärbaum



Binärbaum mit Kompositum



(b) Entwerfen Sie eine mögliche Implementierung zur Erzeugung eines binären Baumes in Java.

einfacher Binärbaum

```java
public class Baum {

  public Knoten wurzel;
  public int anzahl;

  public Baum(int anzahl) {
    this.anzahl = anzahl;

    if (anzahl == 0) {
      return;
    } else {
      wurzel = new Knoten();
      // wurzel.wert = anzahl;
      Baum linkBaum = new Baum(anzahl / 2);
       // Zeiger auf linken Teilbaum
      wurzel.links = linkBaum.wurzel;
      Baum rechtBaum = new Baum(anzahl - 1 - anzahl / 2);
      // Zeiger auf rechten Teilbaum
      wurzel.rechts = rechtBaum.wurzel;
    }
  }
```

```
24
25  }
```

```
3   public class Knoten {
4
5     public Knoten links;
6     public Knoten rechts;
7     public int wert;
8
9     public Knoten() {
10    }
11
12  }
```

Binärbaum mit Kompositum

```
3   class Ahnenbaum {
4     private Baumelement wurzel;
5
6     public Ahnenbaum() {
7       wurzel = new Abschluss();
8     }
9
10    public void wurzelSetzen(Baumelement w) {
11      wurzel = w;
12    }
13
14    public Baumelement wurzelGeben() {
15      return wurzel;
16    }
17
18    public int anzahlDatenknotenGeben() {
19      return wurzel.anzahlDatenknotenGeben();
20    }
21
22    public void alleDatenAusgeben() {
23      wurzel.baumdatenAusgeben();
24    }
25
26    public static void main(String[] args) {
27      Ahnenbaum abaum = new Ahnenbaum();
28      Person[] person = new Person[16];
29      Datenknoten[] datenknoten = new Datenknoten[16];
30      person[0] = new Person("Händel", "Georg,Friedrich", "1685",
        ↪  "1759", "Komponist");
31
32      person[1] = new Person("Taust", "Dorothea", "1651", "1730", "");
33      person[2] = new Person("Händel", "Georg", "1622", "1697",
        ↪  "Amtschirurg");
34
35      person[3] = new Person("Cuno", "Dorothea", "1618", "1682", "");
36      person[4] = new Person("Taust", "Georg", "1606", "1685",
        ↪  "Pfarrer");
37      person[5] = new Person("Beichling", "Anna", "1587", "1670", "");
38      person[6] = new Person("Händel", "Valentin", "1582", "1636",
        ↪  "Kupferschmied");
39
40      person[7] = new Person("Olearius", "Catharina", "1595", "1672",
        ↪  "");
```

```
41      person[8] = new Person("Cuno", "Christoph", "1590", "?",
        ↪  "Gerichtsschr.");
42      person[9] = new Person("Taust", "Joh.", "?", "?", "Prediger");
43      person[10] = new Person("Beichling", "Samuel", "?", "?",
        ↪  "Kupferschmied");
44      person[11] = new Person("Händel", "Valentin", "?", "?",
        ↪  "Röhrmeister?");
45
46      person[12] = new Person("Heshusius", "Anna", "1566", "1600", "");
47      person[13] = new Person("Olearius", "Johannes", "1546", "1623",
        ↪  "Prof. theol.");
48      person[14] = new Person("Becker", "Dorothea", "?", "1631", "");
49      person[15] = new Person("Cuno", "Samuel", "um 1555", "1615",
        ↪  "Hospitalprediger");
50
51      for (int i = 0; i < 16; i++) {
52        datenknoten[i] = new Datenknoten(new Abschluss(), new
          ↪  Abschluss(), person[i]);
53      }
54      abaum.wurzelSetzen(datenknoten[0]);
55
56      datenknoten[0].mutterSetzen(datenknoten[1]);
57      datenknoten[0].vaterSetzen(datenknoten[2]);
58
59      datenknoten[1].mutterSetzen(datenknoten[3]);
60      datenknoten[1].vaterSetzen(datenknoten[4]);
61
62      datenknoten[2].mutterSetzen(datenknoten[5]);
63      datenknoten[2].vaterSetzen(datenknoten[6]);
64
65      datenknoten[3].mutterSetzen(datenknoten[7]);
66      datenknoten[3].vaterSetzen(datenknoten[8]);
67
68      datenknoten[4].vaterSetzen(datenknoten[9]);
69
70      datenknoten[6].vaterSetzen(datenknoten[10]);
71
72      datenknoten[7].vaterSetzen(datenknoten[11]);
73
74      datenknoten[8].mutterSetzen(datenknoten[12]);
75      datenknoten[8].vaterSetzen(datenknoten[13]);
76
77      datenknoten[9].mutterSetzen(datenknoten[14]);
78      datenknoten[9].vaterSetzen(datenknoten[15]);
79
80      System.out.println("Anzahl: " + abaum.anzahlDatenknotenGeben());
81      abaum.alleDatenAusgeben();
82      System.out.println();
83
84      System.out.println("Vater der Mutter der Mutter von Händel: ");
85
        ↪  abaum.wurzelGeben().mutterGeben().mutterGeben().vaterGeben().inhaltGeben().datenAusgeben();
86    }
87  }
```

```
3   abstract class Baumelement {
4     public abstract void mutterSetzen(Baumelement nl);
5
6     public abstract void vaterSetzen(Baumelement nr);
7
8     public abstract Baumelement mutterGeben();
```

```
9
10    public abstract Baumelement vaterGeben();
11
12    public abstract Datenelement inhaltGeben();
13
14    public abstract int anzahlDatenknotenGeben();
15
16    public abstract void baumdatenAusgeben();
17  }
```

```
3   class Abschluss extends Baumelement {
4
5     public void mutterSetzen(Baumelement nl) {
6       System.out.println("Ein Abschluss hat keine Mutter!");
7     }
8
9     public void vaterSetzen(Baumelement nr) {
10      System.out.println("Ein Abschluss hat keinen Vater!");
11    }
12
13    public Baumelement mutterGeben() {
14      System.out.println("Mutter nicht bekannt!");
15      return this;
16    }
17
18    public Baumelement vaterGeben() {
19      System.out.println("Vater nicht bekannt!");
20      return this;
21    }
22
23    public Datenelement inhaltGeben() {
24      return null;
25    }
26
27    public int anzahlDatenknotenGeben() {
28      return 0;
29    }
30
31    public void baumdatenAusgeben() {
32    }
33  }
```

```
3   class Datenknoten extends Baumelement {
4     private Baumelement mutter, vater;
5     private Datenelement inhalt;
6
7     public Datenknoten(Baumelement nl, Baumelement nr, Datenelement i) {
8       mutter = nl;
9       vater = nr;
10      inhalt = i;
11    }
12
13    public void mutterSetzen(Baumelement nl) {
14      mutter = nl;
15    }
16
17    public void vaterSetzen(Baumelement nr) {
18      vater = nr;
19    }
20
21    public void inhaltSetzen(Datenelement i) {
```

```
22      inhalt = i;
23    }
24
25    public Baumelement mutterGeben() {
26      return mutter;
27    }
28
29    public Baumelement vaterGeben() {
30      return vater;
31    }
32
33    public Datenelement inhaltGeben() {
34      return inhalt;
35    }
36
37    /////////// rekursive Baummethoden/////////////////
38    public int anzahlDatenknotenGeben() {
39      return 1 + mutter.anzahlDatenknotenGeben() +
        ↪  vater.anzahlDatenknotenGeben();
40    }
41
42    // InOrder
43    public void baumdatenAusgeben() {
44      System.out.print(" [");
45      mutter.baumdatenAusgeben();
46      inhalt.datenAusgeben();
47      vater.baumdatenAusgeben();
48      System.out.print("] ");
49    }
50  }
```

```
3   class Person extends Datenelement {
4     private String geburtsdatum;
5     private String sterbedatum;
6     private String vorname;
7     private String name;
8     private String beruf;
9
10    public Person(String nn) {
11      name = nn;
12    }
13
14    public Person(String nn, String vn, String gebdat, String stedat,
      ↪  String ber) {
15      name = nn;
16      vorname = vn;
17      geburtsdatum = gebdat;
18      sterbedatum = stedat;
19      beruf = ber;
20    }
21
22    public void datenAusgeben() {
23      System.out.print(name + " " + vorname + " (" + geburtsdatum + "-"
        ↪  + sterbedatum + ") " + beruf);
24    }
25  }
```