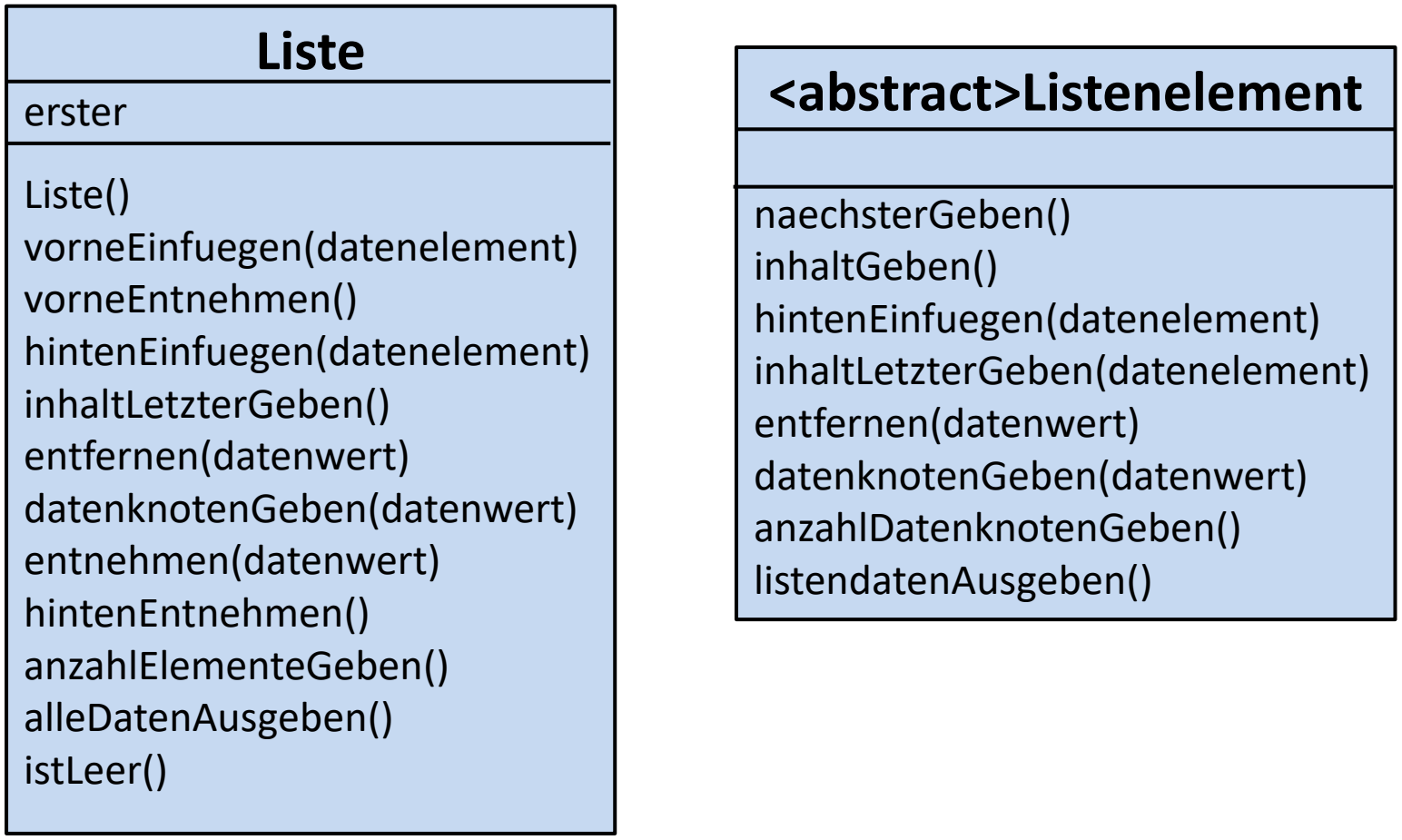
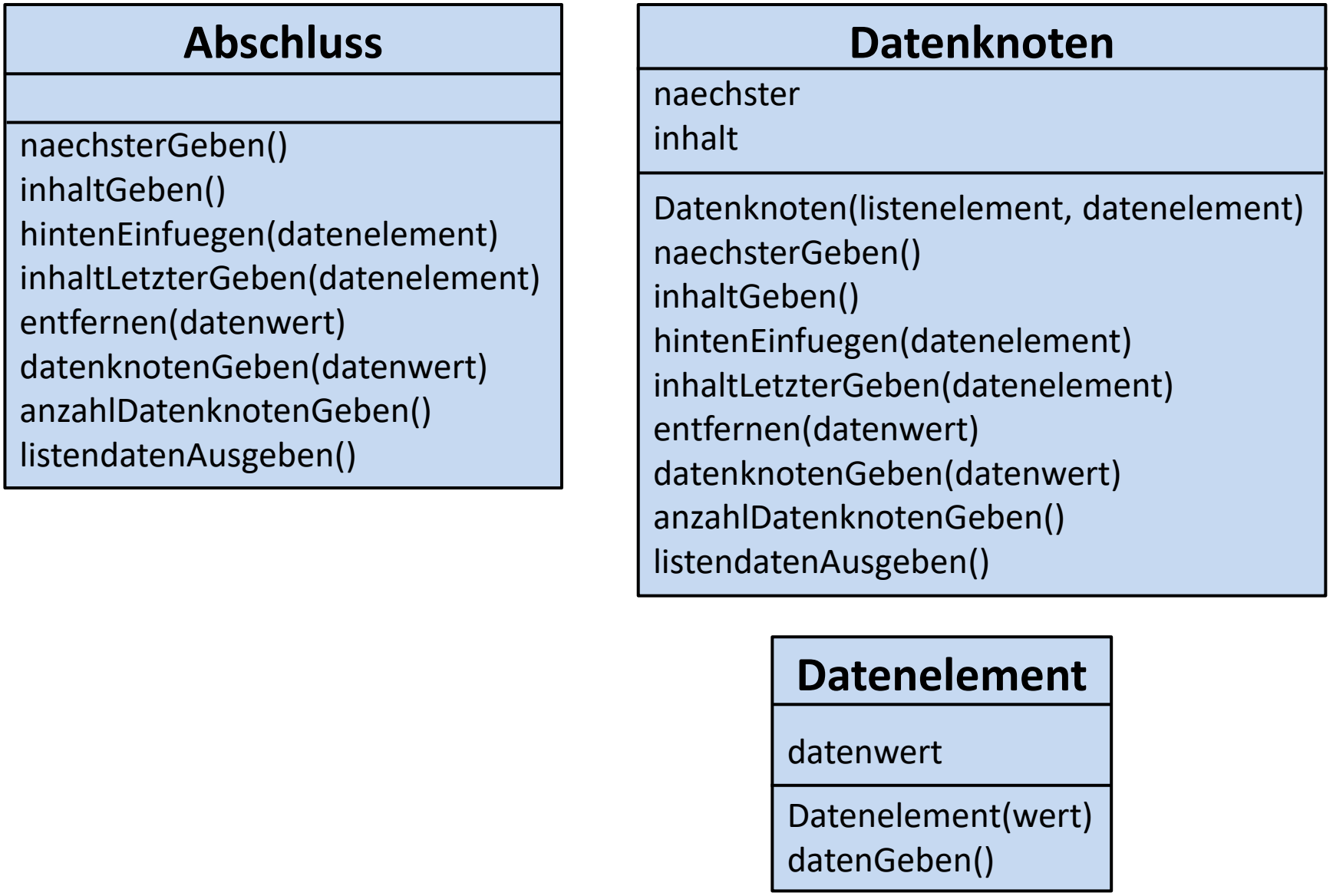

Liste
erster
Liste()
vorneEinfuegen(datenelement)
vorneEntnehmen()
hintenEinfuegen(datenelement)
inhaltLetzterGeben()
entfernen(datenwert)
datenknotenGeben(datenwert)
entnehmen(datenwert)
hintenEntnehmen()
anzahlElementeGeben()
alleDatenAusgeben()
istLeer()
<abstract>Listenelement
naechsterGeben()
inhaltGeben()
hintenEinfuegen(datenelement)
inhaltLetzterGeben(datenelement)
entfernen(datenwert)
datenknotenGeben(datenwert)
anzahlDatenknotenGeben()
listendatenAusgeben()


Abschluss
naechsterGeben()
inhaltGeben()
hintenEinfuegen(datenelement)
inhaltLetzterGeben(datenelement)
entfernen(datenwert)
datenknotenGeben(datenwert)
anzahlDatenknotenGeben()
listendatenAusgeben()
Datenknoten
naechster
inhalt
Datenknoten(listenelement, datenelement)
naechsterGeben()
inhaltGeben()
hintenEinfuegen(datenelement)
inhaltLetzterGeben(datenelement)
entfernen(datenwert)
datenknotenGeben(datenwert)
anzahlDatenknotenGeben()
listendatenAusgeben()
Datenelement
datenwert
Datenelement(wert)
datenGeben()

Speichere das eben erstellte Projekt zur rekursiven Liste unter einem neuen Namen, z.B. Praxis.
Verändere die Klasse Datenelement, sie soll nun Patienten verwalten. Gespeichert werden sollen Name, Alter und Grund für den Arztbesuch.
Erzeuge in der Testklasse mehrere Patienten, die du in die Liste einfügst.
Ergänze nun eine Methode der Klasse Liste, die das Durchschnittsalter der Patienten berechnet. Gib auch alle weiteren dazu nötigen Methoden an.
Lass die Namen aller Patienten ausgeben, die wegen Husten da sind (die wegen einer bestimmten frei wählbaren Krankheit da sind). Ergänze hier ebenso die entsprechenden Methoden.

```
public class Datenelement {
    private String name;
    private int alter;
    private String anlass;

    public Datenelement(String n, int a, String anl){
        name = n;
        alter = a;
        anlass = anl;    }

    public String datenGeben(){
        return name + ", "+alter + ",  " + anlass;    }

    public String nameGeben(){
        return name;    }

    public int alterGeben(){
        return alter;    }

    public String anlassGeben(){
        return anlass;    }

}
```

```
public class Test
{
   private Liste wartezimmer;

   public Test(){
      wartezimmer = new Liste();
   }

   public void ablaufen(){
      wartezimmer.hintenEinfuegen(new Datenelement("Sabine", 23, "Husten"));
      wartezimmer.hintenEinfuegen(new Datenelement("Susi", 45, "Impfung"));
      wartezimmer.hintenEinfuegen(new Datenelement("Jacob", 32, "Kopfschmerzen"));
      wartezimmer.hintenEinfuegen(new Datenelement("Armin", 49, "Husten"));
      wartezimmer.alleDatenAusgeben();
      System.out.println("Im Wartezimmer sind " +wartezimmer.anzahlElementeGeben()+ "
                                                    Patienten!");
      System.out.println("Durchschnittsalter: "+ wartezimmer.durchschnittsalterBerechnen());
      wartezimmer.namenslisteGeben("Husten");
   }

}
```

In Liste:

```
public double durchschnittsalterBerechnen(){
    int gesamtalter = erster.gesamtalterBestimmen();
    return (double) gesamtalter/anzahlElementeGeben();
}
```

In Listenelement:

```
public abstract int gesamtalterBestimmen();
```

In Datenknoten:

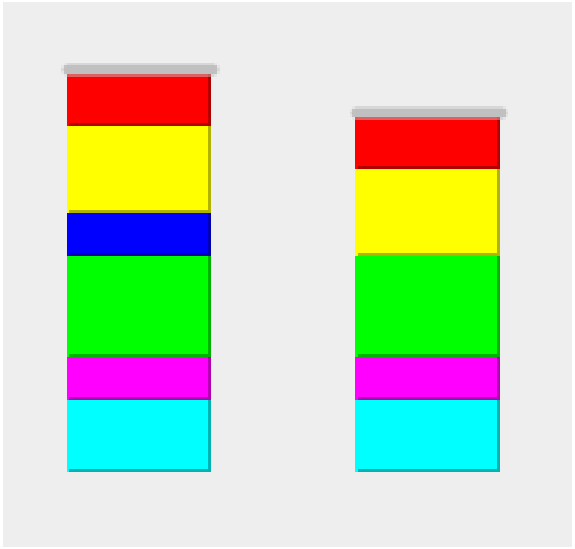
```
public String namenslisteGeben(String anlass){
    if (anlass.equals(inhalt.anlassGeben())){
        return inhalt.nameGeben()+ "\n"
                    +naechster.namenslisteGeben(anlass); }
    else {
        return naechster.namenslisteGeben(anlass);
    }
}
```

In Abschluss:

```
public String namenslisteGeben(String anlass){
    return "";
}
```

In Liste:
```
public void namenslisteGeben(String anlass){
    System.out.println(erster.namenslisteGeben(anlass));
}
```
In Listenelement:
```
public abstract String namenslisteGeben(String anlass);
```
In Datenknoten:
```
public String namenslisteGeben(String anlass){
    if (anlass.equals(inhalt.anlassGeben())){
        return inhalt.nameGeben()+ "\n"
                    +naechster.namenslisteGeben(anlass);
    }
    else {
        return naechster.namenslisteGeben(anlass);
    }
}
```
In Abschluss:
```
public int gesamtalterBestimmen(){
    return 0;
}
```

S.31 / 4 Turm aus Bausteinen



Turm
erster
obenEinfuegen(datenelement) obenEntnehmen(datenelement) anzahlSpielsteineGeben() listendatenAusgeben() turmhoeheGeben() turmZeichnen(xpos,ypos)

Datenelement
farbe breite hoehe
Datenelement(f, h) datenGeben() rechteckZeichnen(xpos, ypos) hoeheGeben() farbeGeben()

(Moodle: S31-A4-Vorlage);
entspricht im Wesentlichen der rekursiven Liste mit TODO-Hinweisen;

Datenelement:

```
public class Datenelement {

    private String farbe;
    private int breite;
    private int hoehe;

    public Datenelement(String f, int h){
        farbe = f;
        breite = 50;
        hoehe = h;
    }

    public String datenGeben(){
        return "Farbe: " + farbe + ", Höhe: " + hoehe;
    }
```

gültige Farben: siehe Klasse Rechteck

Datenelement:

```
    public void rechteckZeichnen(int xpos, int ypos){
        Rechteck r = new Rechteck(xpos, ypos, breite, hoehe);
        r.setzeFarbe(farbe);
        r.sichtbarMachen();
    }

    public int hoeheGeben(){
        return hoehe;
    }

    public String farbeGeben(){
        return farbe;
    }

}
```

In Turm:

- evtl. datenwert durch farbe ersetzen

- ```
  public int turmhoeheGeben(){
        return erster.turmhoeheGeben();
  }
  ```

- ```
  public void turmZeichnen(int xpos,int ypos){
        erster.turmZeichnen(xpos, ypos);
  }
  ```

In Listenelement:

- evtl. datenwert durch farbe ersetzen

- public abstract int turmhoeheGeben();

- public abstract void turmZeichnen(int xpos, int ypos);

In Datenknoten:

- evtl. datenwert durch farbe ersetzen:
  if (farbe.equals(inhalt.farbeGeben()))…

- ```
  public int turmhoeheGeben(){
        return inhalt.hoeheGeben()
                              +naechster.turmhoeheGeben();
  }
  ```
  ```

In Datenknoten:

```
public void turmZeichnen(int xpos, int ypos){
    inhalt.rechteckZeichnen(xpos, ypos-inhalt.hoeheGeben());
    naechster.turmZeichnen(xpos, ypos-inhalt.hoeheGeben());
}
```

In Abschluss:

- evtl. datenwert durch farbe ersetzen
-
```
public int turmhoeheGeben(){
    return 0;
}
```
-
```
public void turmZeichnen(int xpos, int ypos){
    Linie lin = new Linie(xpos,ypos,xpos+50,ypos);
    lin.sichtbarMachen();
}
```

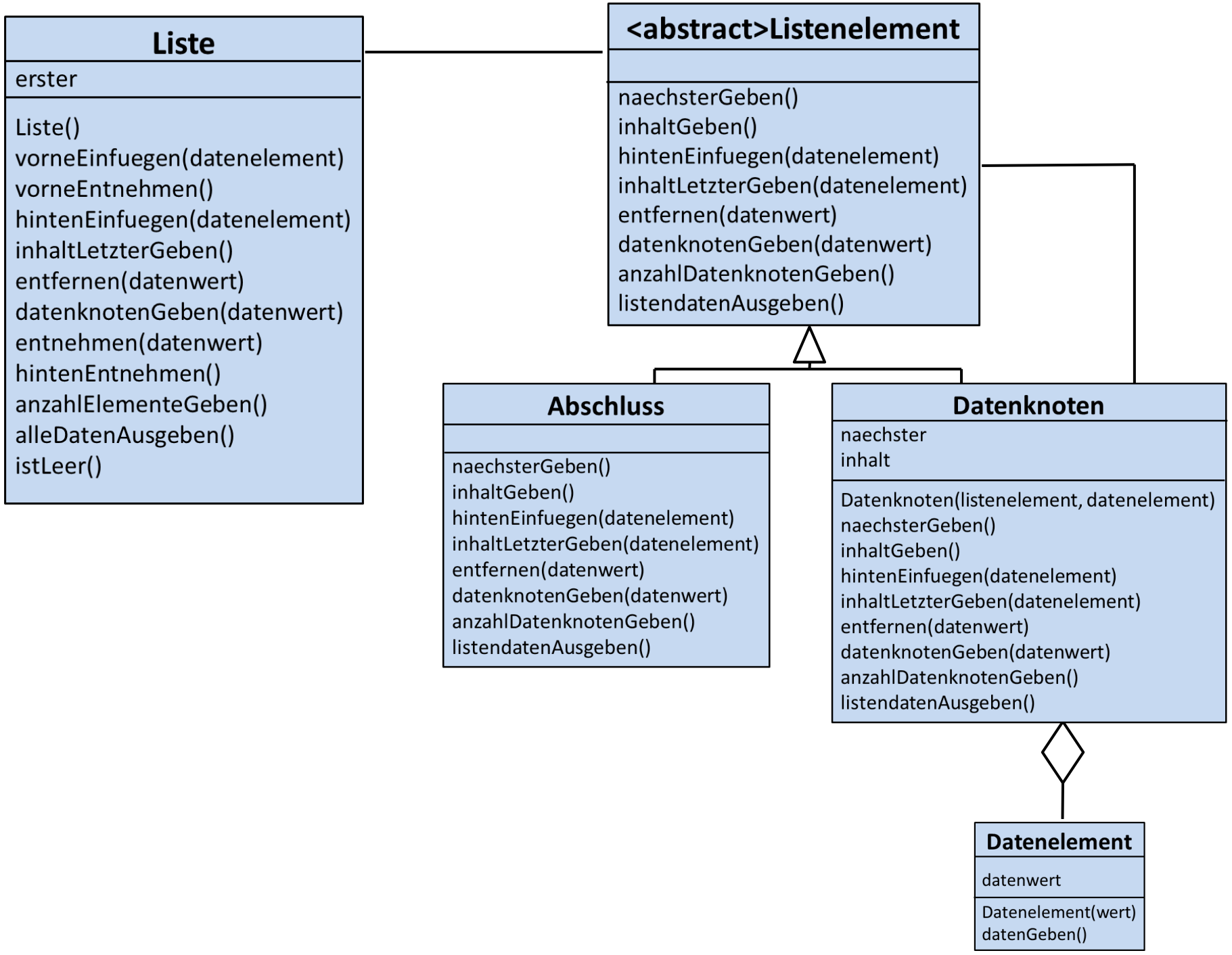
(beim Konstruktor von Linie werden die Koordinaten von Anfangs-und Endpunkt übergeben.)

```
public class Test{
  private Datenelement d1,d2,d3;
  private Turm t;

  public void bausteineErzeugen(){
     d1=new Datenelement("rot", 20);
     d2=new Datenelement("blau", 20);
     d3=new Datenelement("cyan", 30);
   }

   public void ablaufen(){
      bausteineErzeugen();
      t = new Turm();
      t.vorneEinfuegen(d1);
      t.vorneEinfuegen(d2);
      t.vorneEinfuegen(d3);
      t.vorneEinfuegen(d1);
      t.alleDatenAusgeben();
      t.turmZeichnen(200,200);
      t.vorneEntnehmen();
      t.turmZeichnen(400,200);
   }
}
```

```
try {
          Thread.sleep(1500);
        }
    catch (InterruptedException e){}
```


Liste
erster
Liste()
vorneEinfuegen(datenelement)
vorneEntnehmen()
hintenEinfuegen(datenelement)
inhaltLetzterGeben()
entfernen(datenwert)
datenknotenGeben(datenwert)
entnehmen(datenwert)
hintenEntnehmen()
anzahlElementeGeben()
alleDatenAusgeben()
istLeer()
<abstract>Listenelement
naechsterGeben()
inhaltGeben()
hintenEinfuegen(datenelement)
inhaltLetzterGeben(datenelement)
entfernen(datenwert)
datenknotenGeben(datenwert)
anzahlDatenknotenGeben()
listendatenAusgeben()
Abschluss
naechsterGeben()
inhaltGeben()
hintenEinfuegen(datenelement)
inhaltLetzterGeben(datenelement)
entfernen(datenwert)
datenknotenGeben(datenwert)
anzahlDatenknotenGeben()
listendatenAusgeben()
Datenknoten
naechster
inhalt
Datenknoten(listenelement, datenelement)
naechsterGeben()
inhaltGeben()
hintenEinfuegen(datenelement)
inhaltLetzterGeben(datenelement)
entfernen(datenwert)
datenknotenGeben(datenwert)
anzahlDatenknotenGeben()
listendatenAusgeben()
Datenelement
datenwert
Datenelement(wert)
datenGeben()