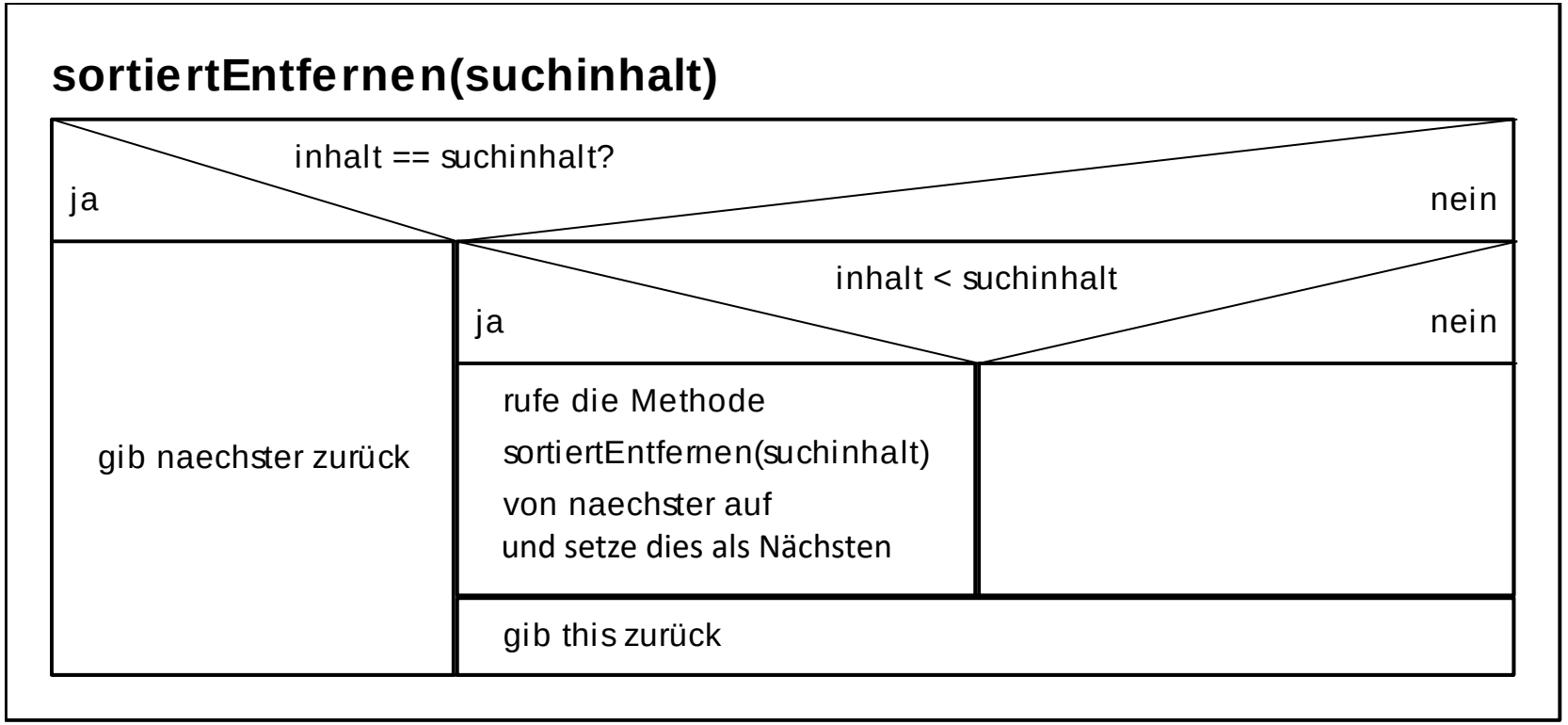
Entfernen eines Knotens mit einem bestimmten Inhalt

Entwirf ein passendes Struktogramm für die Klasse Datenknoten!

Beachte:

Wie weit muss die Liste durchlaufen werden, wenn der geforderte Inhalt nicht gefunden wird?



Implementiere zunächst den hellgrau hinterlegten Bereich!
Benutze hierzu S45-A1-Vorlage-1!

einfuegen(datenelement):

In SortierteListe:

```
public void einfuegen(Datenelement de){
    erster = erster.sortiertEinfuegen(de);
}
```

In *Listenelement*:

```
public abstract Datenknoten
                    sortiertEinfuegen(Datenelement de);
```

In Abschluss:

```
public Datenknoten sortiertEinfuegen(Datenelement de){
        Datenknoten neuerKnoten = new Datenknoten(this, de);
        return neuerKnoten;
    }
```

In Datenknoten:

```
public Datenknoten sortiertEinfuegen(Datenelement de){
        if (inhalt.istKleiner(de)) {
            naechster = naechster.sortiertEinfuegen(de);
            return this;
        }
        else {
            Datenknoten neuerKnoten = new Datenknoten(this, de);
            return neuerKnoten;
        }
    }
```

In *Datenelement*:

```
public abstract boolean istKleiner(Datenelement de);
```

In Eintrag:

```
public  boolean istKleiner(Datenelement de){
    return name.compareTo(((Eintrag) de).name)<0  ;
  }
```

In der Klasse Datenelement gibt es kein Attribut name. Es muss explizit mitgeteilt werden, dass es sich hier um ein Objekt der Subklasse Eintrag handelt (explizite Typum-wandlung, typecast).
Dazu wird die Subklasse in Klammern der Variablen voran gestellt.

Einfaches Beispiel zur Typumwandlung:

int x=3.0/3; liefert Fehlermeldung:
possible loss of precision

Richtig: int x= (int) (3.0/3);

entfernen(datenelement):

In SortierteListe:

```
public void entfernen(Datenelement suchinhalt){
    erster = erster.sortiertEntfernen(suchinhalt);
}
```

In *Listenelement*:

```
public abstract Listenelement
    sortiertEntfernen(Datenelement suchinhalt);
```

In Abschluss:

```
public Listenelement
sortiertEntfernen(Datenelement suchinhalt){
    return this;
}
```

In Datenknoten:

```
public Listenelement sortiertEntfernen(Datenelement
suchinhalt){
        if (inhalt.istGleich(suchinhalt)) {
            return naechster;
        }
        else {
            if (inhalt.istKleiner(suchinhalt)){
                naechster = naechster.sortiertEntfernen(suchinhalt);
            }
            return this;
        }
    }
```

In *Datenelement*:

```
public abstract boolean istGleich(Datenelement de);
```

In Eintrag:

```
public  boolean istGleich (Datenelement de){
      return name.compareTo(((Eintrag) de).name)==0  ;
   }
```

datenknotenGeben(datenelement):

In SortierteListe:

```
private Datenknoten datenknotenGeben
                    (Datenelement suchinhalt){
      return erster.datenknotenGeben(suchinhalt);
   }
```

In *Listenelement*:

```
public abstract Datenknoten
          datenknotenGeben(Datenelement suchinhalt);
```

In Abschluss:

```
public Datenknoten
datenknotenGeben(Datenelement suchinhalt){
       return null;
    }
```

In Datenknoten:

```
public Datenknoten datenknotenGeben(Datenelement
suchinhalt){
        if (inhalt.istGleich(suchinhalt)){
            return this;
        }
        else {
            if(inhalt.istKleiner(suchinhalt)){
                return naechster.datenknotenGeben(suchinhalt);
            }
            else return null;
        }
    }
```

Zusätzlich in SortierteListe:

```
public Datenelement datenelementGeben(Datenelement suchinhalt){
    Datenknoten gesuchterDk = datenknotenGeben(suchinhalt);
    if (gesuchterDk != null) {
        return datenknotenGeben(suchinhalt).inhaltGeben();
    }
    else return null;
}
```

entnehmen(datenelement):

In SortierteListe (analog Liste):

```
public Datenelement entnehmen(Datenelement suchinhalt){
      Datenknoten gesuchterDk = datenknotenGeben(suchinhalt);
      if (gesuchterDk != null) {
         Datenelement gesuchtesDe = gesuchterDk.inhaltGeben();
         entfernen(suchinhalt);
         return gesuchtesDe;
      }
      else return null;
  }
```

Telefonbuch:

```
public void ausgeben(){
        System.out.println("Telefonverzeichnis: ");
        telefonverzeichnis.alleDatenAusgeben();
    }

public void eintragen(String nn, String adr, int nr){
        Eintrag neuerEintrag = new Eintrag(nn, adr, nr);
        telefonverzeichnis.einfuegen(neuerEintrag);
    }

public void eintragEntfernen(String name){
        Eintrag sucheintrag = new Eintrag(name, "", 0);
        telefonverzeichnis.entfernen(sucheintrag);
    }
```

```
public Eintrag eintragEntnehmen(String name){
        Eintrag sucheintrag = new Eintrag(name, "", 0);
        return (Eintrag) telefonverzeichnis.entnehmen(sucheintrag);
    }


public Eintrag eintragGeben(String name){
        Eintrag sucheintrag = new Eintrag(name, "", 0);
        Eintrag e = (Eintrag) telefonverzeichnis.
                                        datenelementGeben(sucheintrag);
        if (e != null) return e;
        else
           return new Eintrag("nicht gefunden", "nicht gefunden", 0);
    }
```

Teste das Programm!

Zu Implementieren:
Klasse TelefonCD analog Telefonbuch, die nach der Telefonnummer sortiert.

Ordnungsrelationen (istKleiner, istGleich) müssen anders ausgeführt werden.

Setze Eintrag als abstrakte Klasse und überschreibe die abstrakten Methoden istKleiner und istGleich in den Subklassen Telefoneintrag und CDEintrag.

Implementiere zuerst die Klassen Eintrag, Telefoneintrag und Telefonbuch und ergänze dann TelefonCD und CDEintrag.

```
public abstract class Eintrag extends Datenelement {

   protected String name;
   protected String adresse;
   protected int telnr;

    public abstract boolean istKleiner(Datenelement de);

    public abstract boolean istGleich (Datenelement de);

    public String datenGeben(){
        return name + " " + adresse + " , Tel: " + telnr;
    }
}
```

```
public class Telefoneintrag extends Eintrag {

    public Telefoneintrag(String n, String a, int tn) {
        name = n;
        adresse = a;
        telnr = tn;
    }

    public  boolean istKleiner(Datenelement de){
      return name.compareTo(((Eintrag) de).name)<0  ;
    }

    public  boolean istGleich (Datenelement de){
        return name.compareTo(((Eintrag) de).name)==0  ;
    }
}
```

Oder Konstruktor in Eintrag und hier nur Aufruf mit super(…), dann in CDEintrag genauso!

```
public class Telefonbuch{


…
public void eintragen(String nn, String adr, int nr){
      Telefoneintrag neuerEintrag = new Telefoneintrag(nn, adr, nr);
      telefonverzeichnis.einfuegen(neuerEintrag);
   }

public void eintragEntfernen(String name){
      Telefoneintrag sucheintrag = new Telefoneintrag(name, "", 0);
      telefonverzeichnis.entfernen(sucheintrag);
   }
  …
```

```
public class TelefonCD{

    private SortierteListe telefonCD;

    public TelefonCD(){
        telefonCD = new SortierteListe();
    }

    public void ausgeben(){
        System.out.println("TelefonCD: ");
        telefonCD.alleDatenAusgeben();
    }

    public void eintragen(String nn, String adr, int nr){
        CDEintrag neuerEintrag = new CDEintrag(nn, adr, nr);
        telefonCD.einfuegen(neuerEintrag);
    }
```

```
public void eintragEntfernen(int nr){
    CDEintrag sucheintrag = new CDEintrag("", "", nr);
    telefonCD.entfernen(sucheintrag);
  }

  public Eintrag eintragEntnehmen(int nr){
    CDEintrag sucheintrag = new CDEintrag("", "", nr);
    return (Eintrag) telefonCD.entnehmen(sucheintrag);
  }

  public Eintrag eintragGeben(int nr){
     CDEintrag sucheintrag = new CDEintrag("", "", nr);
     Eintrag e = (Eintrag) telefonCD.datenelementGeben(sucheintrag);
     if (e != null) return e;
     else return new CDEintrag("nicht gefunden", "nicht gefunden", 0);
  }}
```

1c

```
public class Telefondaten{

    private Telefonbuch tb;
    private TelefonCD tCD;

    public Telefondaten(){
        tb = new Telefonbuch();
        tCD = new TelefonCD();
    }

    public void eintragen(String nn, String adr, int nr){
      tb.eintragen(nn, adr, nr);
      tCD.eintragen(nn, adr, nr);
    }

    public void TBausgeben(){
        tb.ausgeben();
    }
```

1c

```
    public void CDausgeben(){
        tCD.ausgeben();
    }

    public void allesAusgeben(){
        TBausgeben();
        CDausgeben();
    }

    public Eintrag suchen(String name){
        return tb.eintragGeben(name);
    }

    public Eintrag reverssuchen(int nr){
        return tCD.eintragGeben(nr);
    }

    public void loeschen_nr(int nr){
        tb.eintragEntfernen(tCD.eintragEntnehmen(nr).name);
    }
}
```

Teste das Programm!

Benutze bei der Implementierung die bereits vorhandenen Klassen SortierteListe, Listenelement, Datenknoten, Abschluss und Datenelement.

```
public class Eintrag extends Datenelement {
   private String name;
   private String adresse;
   private String email;
   private int gebmonat;
   private int gebtag;

   public Eintrag(String n, String adr, String eml, int gebt, int gebm){
      name=n;
      adresse=adr;
      email=eml;
      gebmonat=gebm;
      gebtag=gebt;
    }
```

```
public boolean istKleiner(Datenelement de){
      if(gebmonat<((Eintrag) de).gebmonat)return true;
      else {
         if (gebmonat>((Eintrag) de).gebmonat) return false;
         else {
            if(gebtag<((Eintrag) de).gebtag) return true;
            else return false;
         }
      }
   }
public  boolean istGleich (Datenelement de){
      return (gebmonat == ((Eintrag) de).gebmonat && gebtag ==
                                                   ((Eintrag) de).gebtag);
   }
public String datenGeben(){
      return name + ", " + adresse + ", " + email +", Geburtstag: " +
                              gebtag + "." + gebmonat+".";
   }
}
```

```
public class Adressheft{

   private SortierteListe geburtstagsliste;

   public Adressheft(){
      geburtstagsliste = new SortierteListe();
   }

   public void ausgeben(){
      System.out.println("Geburtstage: ");
      geburtstagsliste.alleDatenAusgeben();
   }

   public void eintragen(String nn, String adr, String eml, int gebt,
                                                    int gebm){
      Eintrag neuerEintrag = new Eintrag(nn, adr, eml, gebt, gebm);
      geburtstagsliste.einfuegen(neuerEintrag);
   }
```