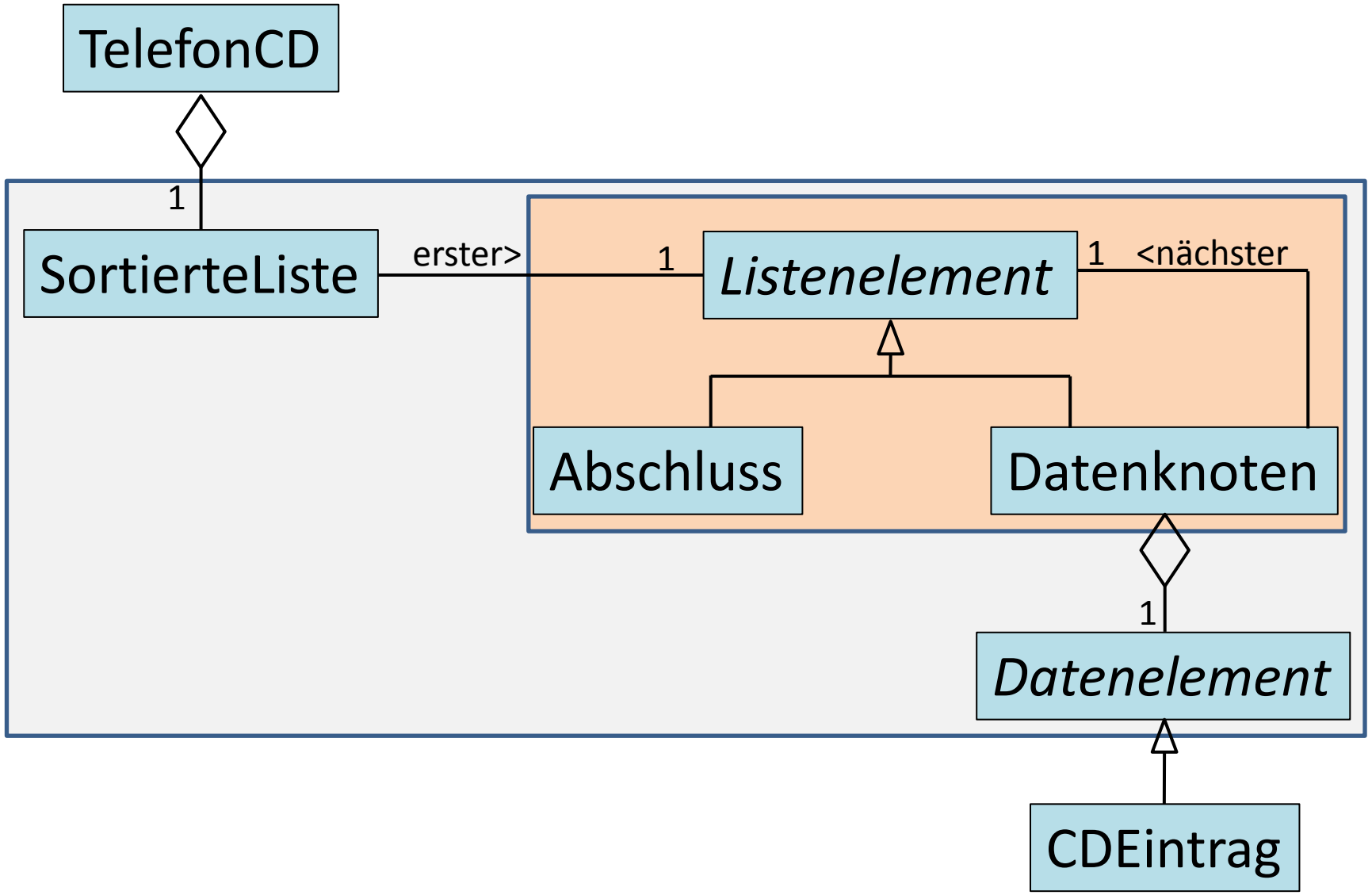
S. 130/2 Suchen in sortierten Adresslisten



Öffne S130-A2-Vorlage, beginne in der Klasse Test.
ablaufen(int n, int s) erzeugt eine neue TelefonCD mit n zufälligen Einträgen nach Nummer sortiert. Dabei werden s Nummern ausgewählt, deren mittlere Suchzeit bestimmt werden soll. Die Suchdauer wird bestimmt von der Anzahl der besuchten Datenknoten.

Methode suchdauer in Test:

- Deklariere Integer-Variable dauer und initialisiere sie mit 0.
- Durchlaufe mit Hilfe einer For-Schleife alle Nummer der Suchliste und ermittle und summiere jeweils die besuchten Datenknoten. Verwende dazu eine Methode zaehlen(int nr) der Klasse TelefonCD.
- Gib den Durchschnittswert für die Suchdauer zurück.

Methode suchdauer in Test:

- Deklariere Integer-Variable dauer und initialisiere sie mit 0.
- Durchlaufe mit Hilfe einer For-Schleife alle Nummer der Suchliste und ermittle und summiere jeweils die besuchten Datenknoten. Verwende dazu eine Methode zaehlen(int nr) der Klasse TelefonCD.
- Gib den Durchschnittswert für die Suchdauer zurück

```
public int suchdauer(){
        int dauer = 0;
        for (int i = 0; i< suchliste.length;i++){
                dauer = dauer + cd.zaehlen(suchliste[i]);
        }
        return (int) (dauer/suchliste.length);
    }
```

Methode zaehlen(int nr) in TelefonCD:

- Erzeuge einen Dummy-CDEintrag (Konstruktor siehe dort).
- Rufe die Methode datenknotenZaehlen der sortierten Liste telefonCD mit diesem Sucheintrag auf und gib dies zurück.

In TelefonCD:

```
public int zaehlen(int nr){
    CDEintrag sucheintrag = new CDEintrag("", "", nr);
    return telefonCD.datenknotenZaehlen(sucheintrag);
}
```

Methode datenknotenZaehlen(Datenelement suchinhalt) in den folgenden Klassen (durchschleifen)

- SortierteListe: Reiche an erster weiter und gib dies zurück.
- Listenelement: Formuliere abstrakte Methode.
- Datenknoten: Hier wird die Arbeit erledigt, es muss gezählt und nötigenfalls weitergereicht werden.
- Abschluss: Zählen wir mit, kostet ja auch Zeit.

In SortierteListe:

```
public int datenknotenZaehlen(Datenelement suchinhalt){
    return erster.datenknotenZaehlen(suchinhalt);
}
```

In <<abstract>> Listenelement:

```
public abstract int datenknotenZaehlen(Datenelement
                                        suchinhalt);
```

- Datenknoten: Hier wird die Arbeit erledigt, es muss gezählt und nötigenfalls weitergereicht werden.
- Abschluss: Zählen wir mit, kostet ja auch Zeit.

In Datenknoten:

```
public int datenknotenZaehlen(Datenelement suchinhalt){
        if (inhalt.istGleich(suchinhalt)){
                return 1;
        }
        else {
                return 1 + naechster.datenknotenZaehlen(suchinhalt);
        }
 }
```

In Abschluss:

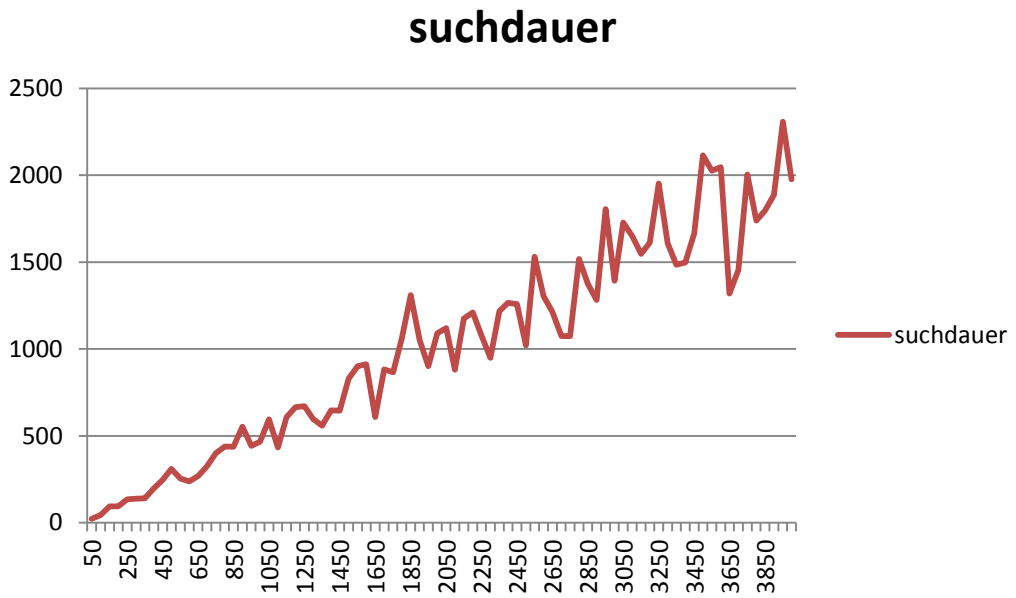
```
   public int datenknotenZaehlen(Datenelement suchinhalt){
        return 1;
   }
```

Variiere n und miss die jeweilige Suchdauer für s Einträge.
Erhöhe z.B. die Listenlänge in 50er-Schritten bis zu einer Obergrenze n und suche jeweils 20 Einträge (in Test).
Trage die Ergebnisse in eine Tabelle ein (for-Schleife!)

```
public void schleife(int n){
        for (int i =50; i<=n; i=i+50){
            System.out.println(i + ": " + ablaufen(i, 20));
        }
    }
```

Welche Laufzeitordnung ergibt sich?
Welche Laufzeitordnung ergäbe sich für eine unsortierte Liste?

n	suchdauer
50	23
100	45
150	94
200	94
250	136
300	138
350	141
400	197
450	248
500	311
550	255
600	238
650	270
700	325
750	401
800	439
850	437
900	554
950	443
1000	467
1050	596
1100	434



Laufzeitordnung: O(n) (linear)
falls unsortiert, ebenfalls O(n).