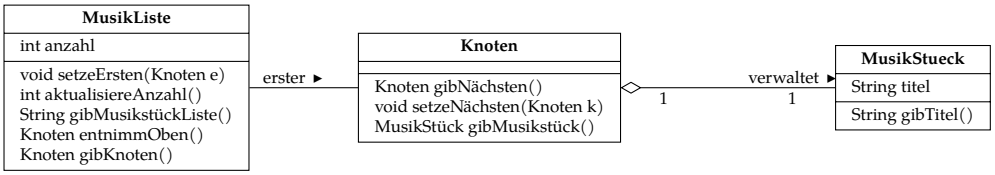
Musikliste

Aufgabe 1



Das Klassendiagramm zeigt den Aufbau einer Playlist.

(a) Implementieren Sie das Klassendiagramm.

Klasse „MusikListe“

```
3  public class MusikListe {
4    private Knoten erster;
5    private int anzahl;
6
7    public MusikListe() {
8      erster = null;
9      anzahl = 0;
10   }
11
12   public void setzeErsten(Knoten e) {
13     erster = e;
14     aktualisiereAnzahl();
15   }
16
17   public int aktualisiereAnzahl() {
18     if (erster == null) {
19       anzahl = 0;
20     } else {
21       int n = 1;
22       Knoten k = erster;
23       while (!(k.gibNächsten() == null)) {
24         k = k.gibNächsten();
25         n = n + 1;
26       }
27       anzahl = n;
28     }
29     return anzahl;
30   }
31
32   public String gibMusikstückListe() {
33     String ausgabe = " ";
34     if (anzahl >= 1) {
35       Knoten k = erster;
36       ausgabe = k.gibMusikstück().gibTitel();
37       for (int i = 1; i <= anzahl - 1; i++) {
38         k = k.gibNächsten();
39         ausgabe = ausgabe + " | " + k.gibMusikstück().gibTitel();
40       }
41     }
42     return ausgabe;
```

```
  }

  public Knoten entnimmOben() {
    if (erster == null) {
      return erster;
    }
    Knoten alterKnoten = erster;
    erster = erster.gibNächsten();
    aktualisiereAnzahl();
    return alterKnoten;
  }

  public Knoten gibKnoten(int position) {
    if ((position < 1) || (position > anzahl)) {
      System.out.println(" FEHLER ! ");
      return null;
    } else {
      Knoten k = erster;
      for (int i = 1; i <= position - 1; i++) {
        k = k.gibNächsten();
      }
      return k;
    }
  }
```

Klasse „Knoten“

```
public class Knoten {
  private Knoten naechster;
  private Knoten vorheriger;
  private MusikStueck lied;

  public Knoten(MusikStueck lied) {
    naechster = null;
    this.lied = lied;
    vorheriger = null;
  }

  public Knoten gibNächsten() {
    return naechster;
  }

  public void setzeNächsten(Knoten naechsterKnoten) {
    naechster = naechsterKnoten;
  }

  public MusikStueck gibMusikstück() {
    return lied;
  }
```

Klasse „MusikStueck“

```
public class MusikStueck {
  private String titel;

  public MusikStueck(String t) {
    titel = t;
  }

```

```
10    public String gibTitel() {
11      return titel;
12    }
13  }
```

(b) Schreiben Sie eine Testklasse, in der Sie eine Playlist mit mindestens vier Liedern erstellen.

```
3   public class TestKlasse {
4
5     public static void main(String[] args) {
6       MusikListe test = new MusikListe();
7
8       MusikStueck stueck1 = new MusikStueck("Hangover");
9       MusikStueck stueck2 = new MusikStueck("Roar");
10      MusikStueck stueck3 = new MusikStueck("On the Floor");
11      MusikStueck stueck4 = new MusikStueck("Whistle");
12
13      Knoten platz1 = new Knoten(stueck1);
14      Knoten platz2 = new Knoten(stueck2);
15      Knoten platz3 = new Knoten(stueck3);
16      Knoten platz4 = new Knoten(stueck4);
```

Aufgabe 2

Die Playlist aus Aufgabe 1 soll nun erweitert werden. Aktualisieren Sie Ihren Code entsprechend!

(a) Bisher wurde das erste Element der Musikliste in einer öffentlich sichtbaren Variable gespeichert, dies ist jedoch nicht sinnvoll. Erstellen Sie eine Methode `setzeErsten()`, mit der anstatt dessen die Liste der erstellten Musikstücke angesprochen werden kann.

```
12    public void setzeErsten(Knoten e) {
13      erster = e;
```

(b) Außerdem wird ein Attribut `anzahl` benötigt, dass mit Hilfe der Methode `aktualisiereAnzahl()` auf dem aktuellen Stand gehalten werden kann.

```
14      aktualisiereAnzahl();
15    }
```

(c) Eine weitere Methode `gibMusikstückListe()` soll die Titel aller Lieder in der Liste als `String` zurückgeben.

```
32    public String gibMusikstückListe() {
33      String ausgabe = " ";
34      if (anzahl >= 1) {
35        Knoten k = erster;
36        ausgabe = k.gibMusikstück().gibTitel();
37        for (int i = 1; i <= anzahl - 1; i++) {
38          k = k.gibNächsten();
39          ausgabe = ausgabe + " | " + k.gibMusikstück().gibTitel();
40        }
41      }
42      return ausgabe;
```

```
43    }
```

(d) Mit `entnimmOben()` soll der erste Titel aus der Liste entnommen werden können.

```
45    public Knoten entnimmOben() {
46      if (erster == null) {
47        return erster;
48      }
49      Knoten alterKnoten = erster;
50      erster = erster.gibNächsten();
51      aktualisiereAnzahl();
52      return alterKnoten;
53    }
```

(e) Es soll der Titel des Musikstücks ermittelt werden, das an einer bestimmten Position in der Musikliste abgespeichert ist. Implementieren Sie dazu die Methode `gibKnoten()`.

```
55    public Knoten gibKnoten(int position) {
56      if ((position < 1) || (position > anzahl)) {
57        System.out.println(" FEHLER ! ");
58        return null;
59      } else {
60        Knoten k = erster;
61        for (int i = 1; i <= position - 1; i++) {
62          k = k.gibNächsten();
63        }
64        return k;
65      }
66    }
```

(f) Die Musikliste soll nun in eine doppelt verkettete Liste umgebaut werden. Fügen Sie entsprechende Attribute, getter- und setter-Methoden hinzu.

```
5     private Knoten vorheriger;

22    public MusikStueck gibMusikstück() {
23      return lied;
24    }
25
26    public Knoten gibVorherigen() {
27      return vorheriger;
28    }
```

(g) Testen Sie die Funktionalität der neuen Methoden in Ihrer Testklasse.

```
17
18      test.setzeErsten(platz1);
19      platz1.setzeNächsten(platz2);
20      platz2.setzenVorherigen(platz1);
21      platz2.setzeNächsten(platz3);
22      platz3.setzenVorherigen(platz2);
23      platz3.setzeNächsten(platz4);
24      platz4.setzenVorherigen(platz3);
25      test.aktualisiereAnzahl();
26      System.out.println(test.gibMusikstückListe());
```

```
27        System.out.println("Der erste Knoten der Liste wird entnommen: " +
          ↪  test.entnimmOben().gibMusikstück().gibTitel());
28        System.out.println(test.gibMusikstückListe());
29        System.out.println("Der zweite Knoten der Liste wird gegeben: " +
          ↪  test.gibKnoten(2).gibMusikstück().gibTitel());
```

Rekursion

Die Anzahl der Titel in der Musikliste aus Aufgabe 1 kann auch unter Verwendung einer rekursiven Methode ermittelt werden. Implementieren Sie eine Methode `zaehleEintraege()`, die analog zu `aktualisiereAnzahl()` angibt, wie viele Titel in der Musikliste gespeichert sind, dies aber rekursiv ermittelt! Testen Sie diese Methode in der Testklasse. Hinweis: Um für die gesamte Musikliste aufgerufen werden zu können, muss diese Methode in der Musikliste selbst und auch in der Klasse Knoten existieren!

Klasse „MusikListe"

```
68    public int zähleEinträge() {
69      if (erster == null) {
70        return 0;
71      } else {
72        return erster.zähleEinträge();
73      }
74    }
```

Klasse „Knoten"

```
34    public int zähleEinträge() {
35      if (this.gibNächsten() == null) {
36        return 1;
37      } else {
38        return this.gibNächsten().zähleEinträge() + 1;
39      }
40    }
```

Klasse „TestKlasse"

```
30
        ↪  System.out.println("Die Anzahl der Listeneinträge wird rekursiv ermittelt: "
        ↪  + test.zähleEinträge());
```