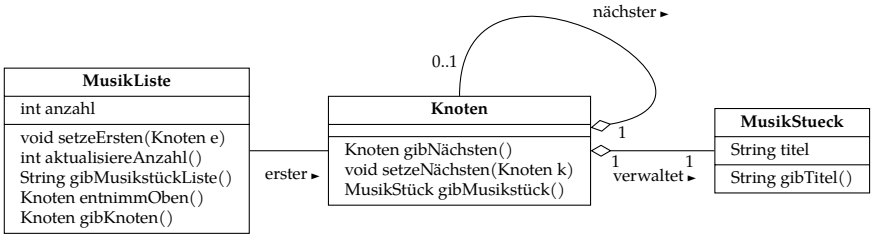
Musikliste

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 knoten) {
13     erster = knoten;
14     aktualisiereAnzahl();
15   }
16
17   public int aktualisiereAnzahl() {
18     if (erster == null) {
19       anzahl = 0;
20     } else {
21       int zähler = 1;
22       Knoten knoten = erster;
23       while (!(knoten.gibNächsten() == null)) {
24         knoten = knoten.gibNächsten();
25         zähler = zähler + 1;
26       }
27       anzahl = zähler;
28     }
29     return anzahl;
30   }
31
32   public String gibMusikstückListe() {
33     String ausgabe = " ";
34     if (anzahl >= 1) {
35       Knoten knoten = erster;
36       ausgabe = knoten.gibMusikstück().gibTitel();
37       for (int i = 1; i <= anzahl - 1; i++) {
```

```
        knoten = knoten.gibNächsten();
        ausgabe = ausgabe + " | " + knoten.gibMusikstück().gibTitel();
      }
    }
    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;
    }
    Knoten knoten = erster;
    for (int i = 1; i <= position - 1; i++) {
      knoten = knoten.gibNächsten();
    }
    return knoten;
  }

```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/aud/listen/musikliste/MusikListe.java`

Klasse „Knoten“

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

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

  public Knoten gibNächsten() {
    return nächster;
  }

  public void setzeNächsten(Knoten nächsterKnoten) {
    nächster = nächsterKnoten;
  }

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

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/aud/listen/musikliste/Knoten.java`

Klasse „MusikStueck"

```java
public class MusikStueck {
  private String titel;

  public MusikStueck(String titel) {
    this.titel = titel;
  }

  public String gibTitel() {
    return titel;
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/aud/listen/musikliste/MusikStueck.java`

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

```java
  private MusikListe macheListe() {
    MusikListe liste = new MusikListe();

    MusikStueck stueck1 = new MusikStueck("Hangover");
    MusikStueck stueck2 = new MusikStueck("Roar");
    MusikStueck stueck3 = new MusikStueck("On the Floor");
    MusikStueck stueck4 = new MusikStueck("Whistle");

    Knoten platz1 = new Knoten(stueck1);
    Knoten platz2 = new Knoten(stueck2);
    Knoten platz3 = new Knoten(stueck3);
    Knoten platz4 = new Knoten(stueck4);

    liste.setzeErsten(platz1);
    platz1.setzeNächsten(platz2);
    platz2.setzenVorherigen(platz1);
    platz2.setzeNächsten(platz3);
    platz3.setzenVorherigen(platz2);
    platz3.setzeNächsten(platz4);
    platz4.setzenVorherigen(platz3);
    liste.aktualisiereAnzahl();
    return liste;
  }
```

Code-Beispiel auf Github ansehen: `src/test/java/org/bschlangaul/aufgaben/aud/listen/musikliste/MusikListeTest.java`

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.

```java
  public void setzeErsten(Knoten knoten) {
    erster = knoten;
    aktualisiereAnzahl();
```

```
15    }
```

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/aufgaben/aud/listen/musikliste/MusikListe.java

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

```
5     private int anzahl;
6
```

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/aufgaben/aud/listen/musikliste/MusikListe.java

(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 knoten = erster;
36        ausgabe = knoten.gibMusikstück().gibTitel();
37        for (int i = 1; i <= anzahl - 1; i++) {
38          knoten = knoten.gibNächsten();
39          ausgabe = ausgabe + " | " + knoten.gibMusikstück().gibTitel();
40        }
41      }
42      return ausgabe;
43    }
```

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/aufgaben/aud/listen/musikliste/MusikListe.java

(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    }
```

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/aufgaben/aud/listen/musikliste/MusikListe.java

(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      }
60      Knoten knoten = erster;
61      for (int i = 1; i <= position - 1; i++) {
62        knoten = knoten.gibNächsten();
```

```
63      }
64      return knoten;
65    }
66
```

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/aufgaben/aud/listen/musikliste/MusikListe.java

(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;
```

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/aufgaben/aud/listen/musikliste/Knoten.java

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

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/aufgaben/aud/listen/musikliste/Knoten.java

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

```
3   import static org.junit.Assert.*;
4   import org.junit.Test;
5
6   public class MusikListeTest {
7
8     private MusikListe macheListe() {
9       MusikListe liste = new MusikListe();
10
11      MusikStueck stueck1 = new MusikStueck("Hangover");
12      MusikStueck stueck2 = new MusikStueck("Roar");
13      MusikStueck stueck3 = new MusikStueck("On the Floor");
14      MusikStueck stueck4 = new MusikStueck("Whistle");
15
16      Knoten platz1 = new Knoten(stueck1);
17      Knoten platz2 = new Knoten(stueck2);
18      Knoten platz3 = new Knoten(stueck3);
19      Knoten platz4 = new Knoten(stueck4);
20
21      liste.setzeErsten(platz1);
22      platz1.setzeNächsten(platz2);
23      platz2.setzenVorherigen(platz1);
24      platz2.setzeNächsten(platz3);
25      platz3.setzenVorherigen(platz2);
26      platz3.setzeNächsten(platz4);
27      platz4.setzenVorherigen(platz3);
28      liste.aktualisiereAnzahl();
29      return liste;
30    }
31
32    @Test
33    public void methodeGibMusikstückListe() {
34      MusikListe liste = macheListe();
```

```
35      assertEquals("Hangover | Roar | On the Floor | Whistle",
          ↪ liste.gibMusikstückListe());
36    }
37
38    @Test
39    public void methodeEntnimmOben() {
40      MusikListe liste = macheListe();
41      assertEquals("Hangover",
          ↪ liste.entnimmOben().gibMusikstück().gibTitel());
42      assertEquals("Roar | On the Floor | Whistle",
          ↪ liste.gibMusikstückListe());
43    }
44
45    @Test
46    public void methodeZähleEinträge() {
47      MusikListe liste = macheListe();
48      assertEquals(4, liste.zähleEinträge());
49    }
50  }
```

Code-Beispiel auf Github ansehen: `src/test/java/org/bschlangaul/aufgaben/aud/listen/musikliste/MusikListeTest.java`

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“

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

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/aud/listen/musikliste/MusikListe.java`

Klasse „Knoten“

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

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/aud/listen/musikliste/Knoten.java`

Klasse „TestKlasse“

```
45    @Test
46    public void methodeZähleEinträge() {
47      MusikListe liste = macheListe();
48      assertEquals(4, liste.zähleEinträge());
49    }
50  }
```

Code-Beispiel auf Github ansehen: `src/test/java/org/bschlangaul/aufgaben/aud/listen/musikliste/MusikListeTest.java`