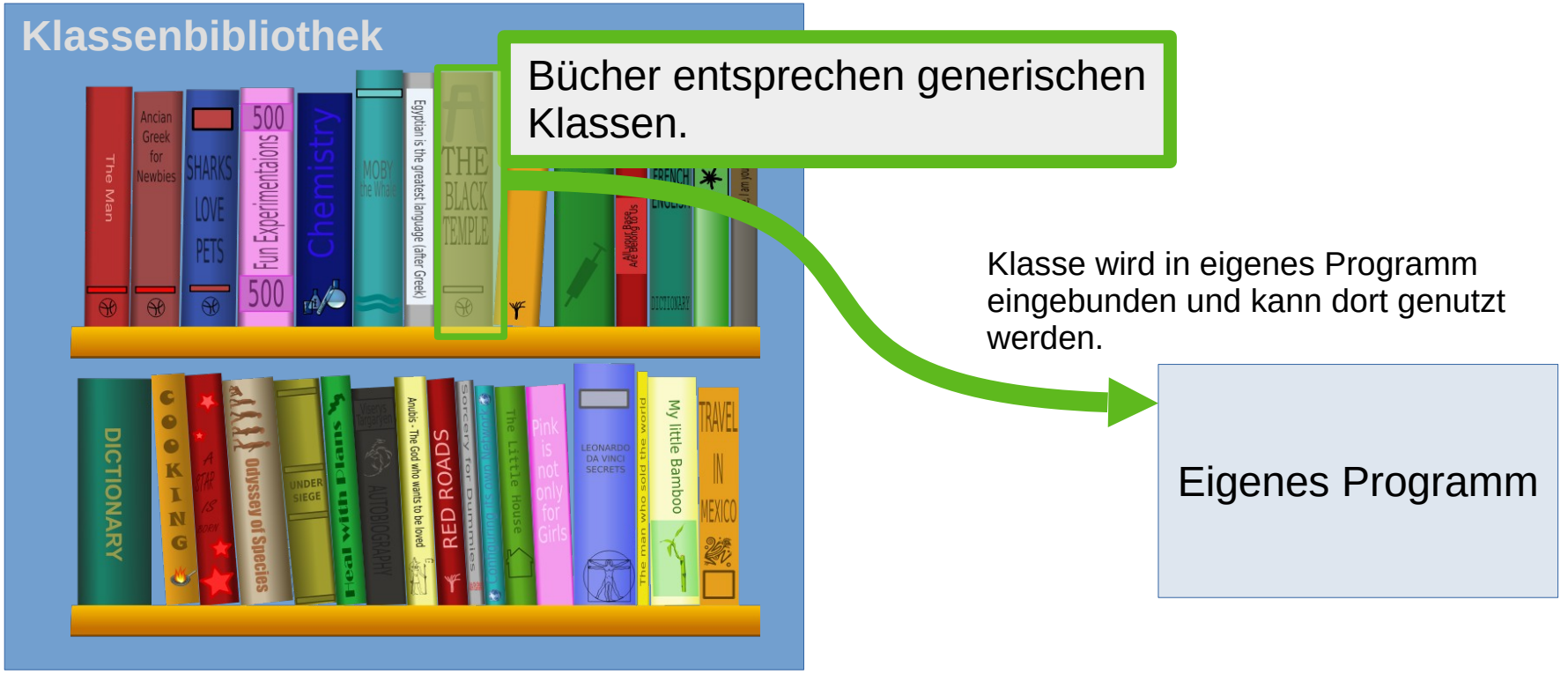
Abstrakte Datentypen aus der Java Klassenbibliothek am Beispiel ArrayList()

Klassenbibliothek?

- Klassenbibliothek: Eine Sammlung nützlicher generischer Klassen
- In Java heißen die Bibliotheken „**Packages**“
- ADTs zur Verwaltung von Objektsammlungen werden oft benötigt, das Package **java.util** enthält mehrere Klassen um derartige Aufgaben zu lösen



Nutzen einer Klassenbibliothe/Package

ArrayList Klasse aus dem Package java.util laden

```
import java.util.ArrayList;

public class MusikSammlung
{
    // Eine ArrayList, in der die Namen
    // von Audiodateien gespeichert werden können.

    private ArrayList<String> dateien;

    /**
     * Konstruktor: Erzeuge eine MusikSammlung.
     */
    public MusikSammlung()
    {
        dateien = new ArrayList<>();
    }
[...]
```

dateien ist eine Variable vom Typ ArrayList, sie kann String Objekte speichern. Sichtbarkeit ist „private“

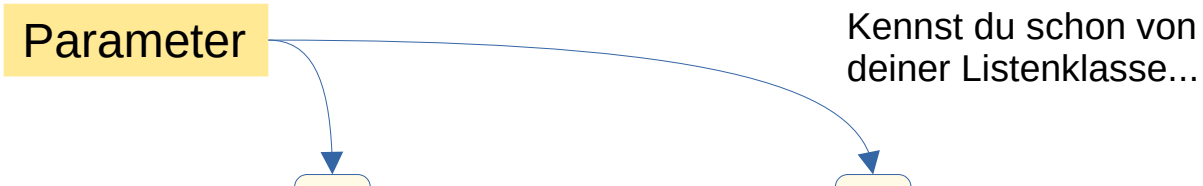
Ein Objekt des Typs ArrayList wird instanziiert und eine Referenz auf dieses Objekt in **dateien** gespeichert.
Der Typ der ArrayList wird aus der Deklaration der Objektvariablen abgeleitet.
new ArrayList <>() ist also eine Abkürzung für
new ArrayList<String>()

Wir sagen:
„Eine ArrayList von Strings“
„Eine ArrayList von Integers“
...

Generische Klassen

Generische Klassen (oder parametrisierte Klassen)

```
private ArrayList<String> dateien;
```




```
public class ListeIterativ<T> extends Liste<T> {
[...]
```

Der Typ-Parameter gibt an, welche Art von Objekten die ArrayList speichern soll. Auch eigene Klassen sind erlaubt.

```
ArrayList<Person>
ArrayList<TicketAutomat>
ArrayList<Int>
...
```

Liste for free...

Die Bibliotheksklasse **ArrayList** implementiert Listenfunktionalität.
Sie hat Methoden wie:

- add()
- get()
- size()
- …

RTFM: https://docs.oracle.com/javase/8/docs/api/java/util/package-summary.html

AbstractSet<E>	This class provides a skeletal implementation of the `Set` interface to minimize the effort required to implement this interface.
ArrayDeque<E>	Resizable-array implementation of the **`Deque`** interface.
ArrayList<E>	Resizable-array implementation of the `List` interface.
Arrays	This class contains various methods for manipulating arrays (such as sorting and searching).
Base64	This class consists exclusively of static methods for obtaining encoders and decoders for the Base64 encoding scheme.

Einschub: Die Sache mit den Buchstaben

Type Parameter Naming Conventions

By convention, type parameter names are single, uppercase letters. This stands in sharp contrast to the variable naming conventions that you already know about, and with good reason: Without this convention, it would be difficult to tell the difference between a type variable and an ordinary class or interface name.

The most commonly used type parameter names are:

- E - Element (used extensively by the Java Collections Framework)
- K - Key
- N - Number
- T - Type
- V - Value
- S,U,V etc. - 2nd, 3rd, 4th types

Als Parameter nimmt man für gewöhnlich einzelne Grossbuchstaben.
Reihenfolge matters, manche Buchstaben haben eine Art „Bedeutung“: Bei den Collections nimmt man oft „E“ für „Element“ – unsere Liste hatte da ein „T“ wie „Type“, gegen die Konvention – letztlich ist es nur eine Ersetzung, man könnte auich XYZ nehmen.

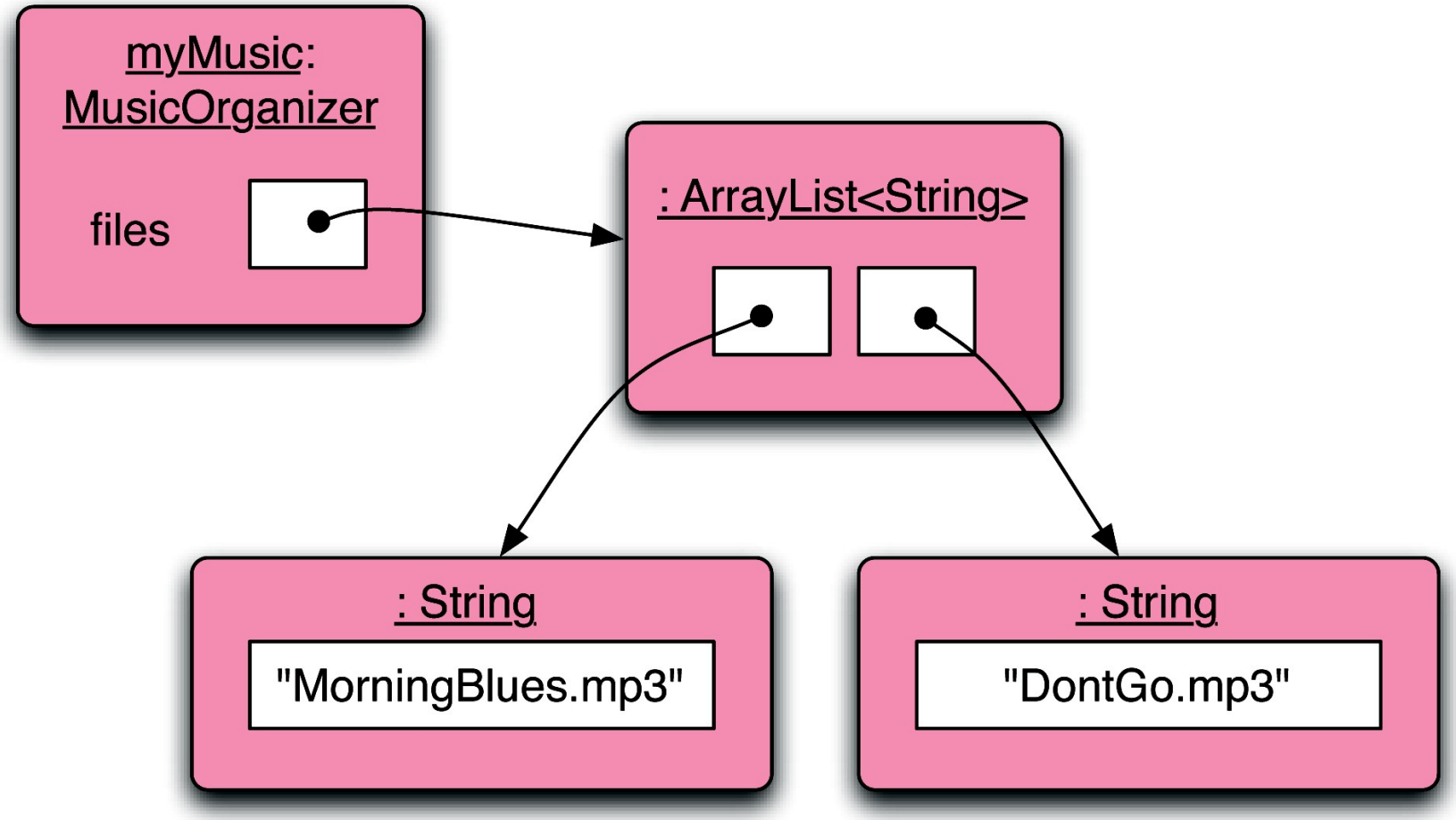
Die Musikbibliothek v1

- Einzelne Musikdateien können verwaltet werden
- Es gibt keine Grenze für die Zahl der Titel
- Man kann die Zahl der Titel abfragen
- Man kann eine Liste aller Titel ausgeben

V1 testen. Wiki/Git

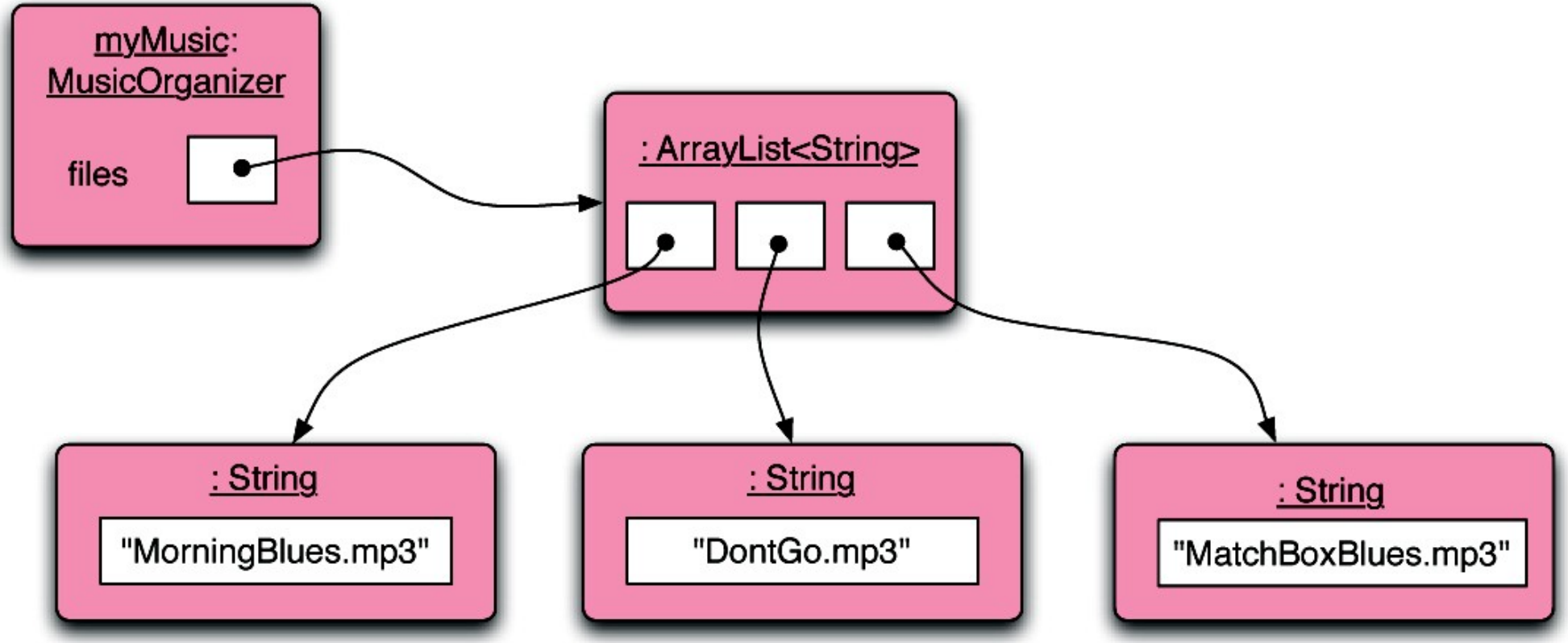
Hinter den Kulissen I

Musikverwaltung mit zwei Titeln:



Hinter den Kulissen II

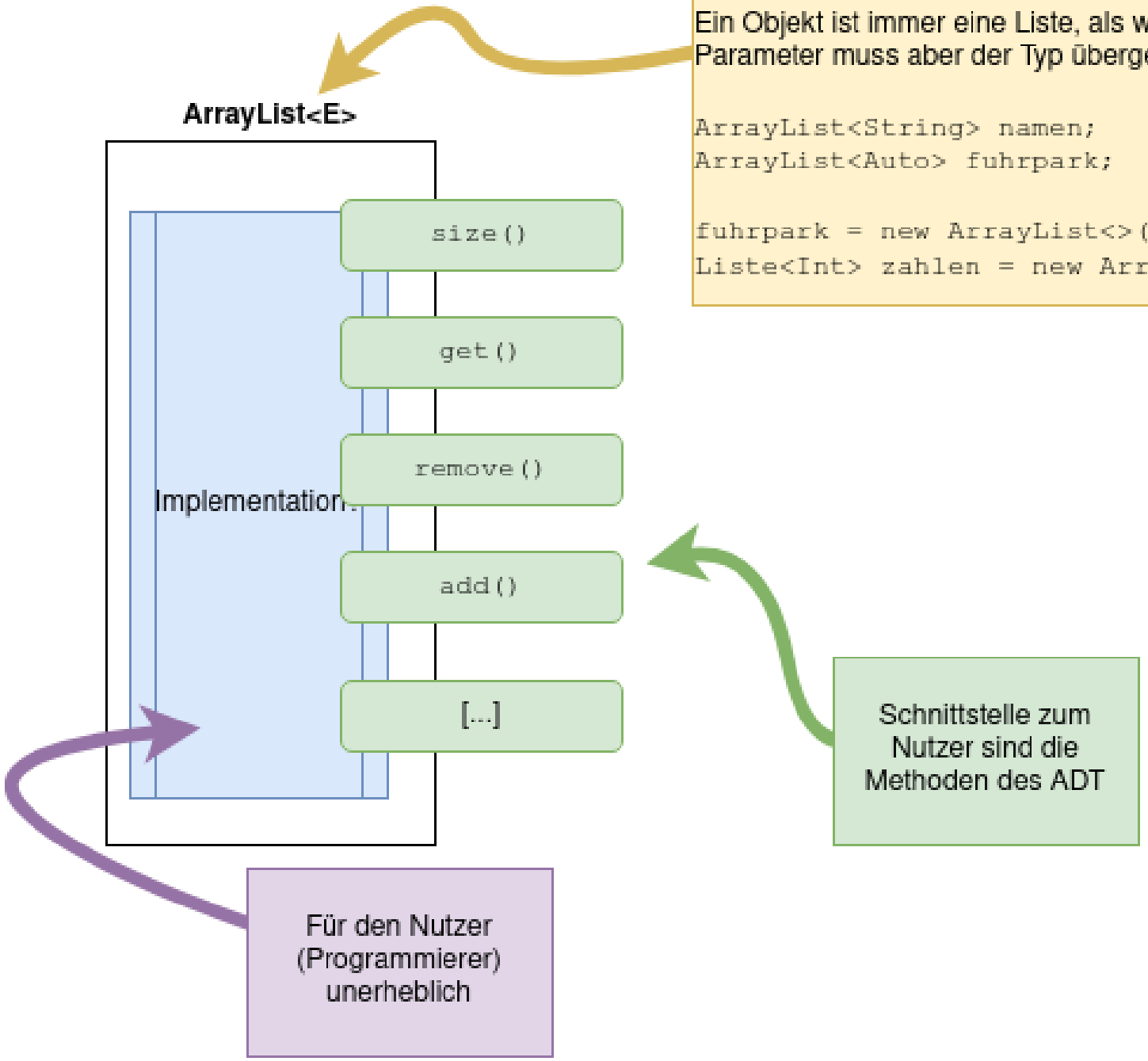
Neuen Titel einfügen….



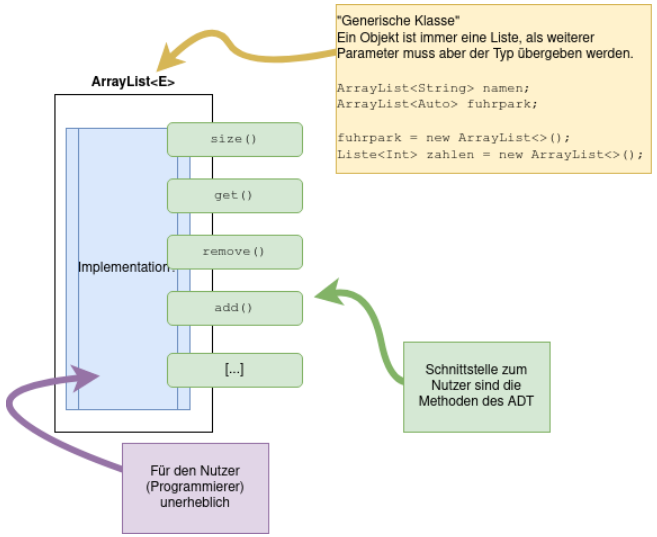
Erinnerung

"Generische Klasse"
Ein Objekt ist immer eine Liste, als weiterer Parameter muss aber der Typ übergeben werden.

```
ArrayList<String> namen;
ArrayList<Auto> fuhrpark;

fuhrpark = new ArrayList<>();
Liste<Int> zahlen = new ArrayList<>();
```



Benutzung

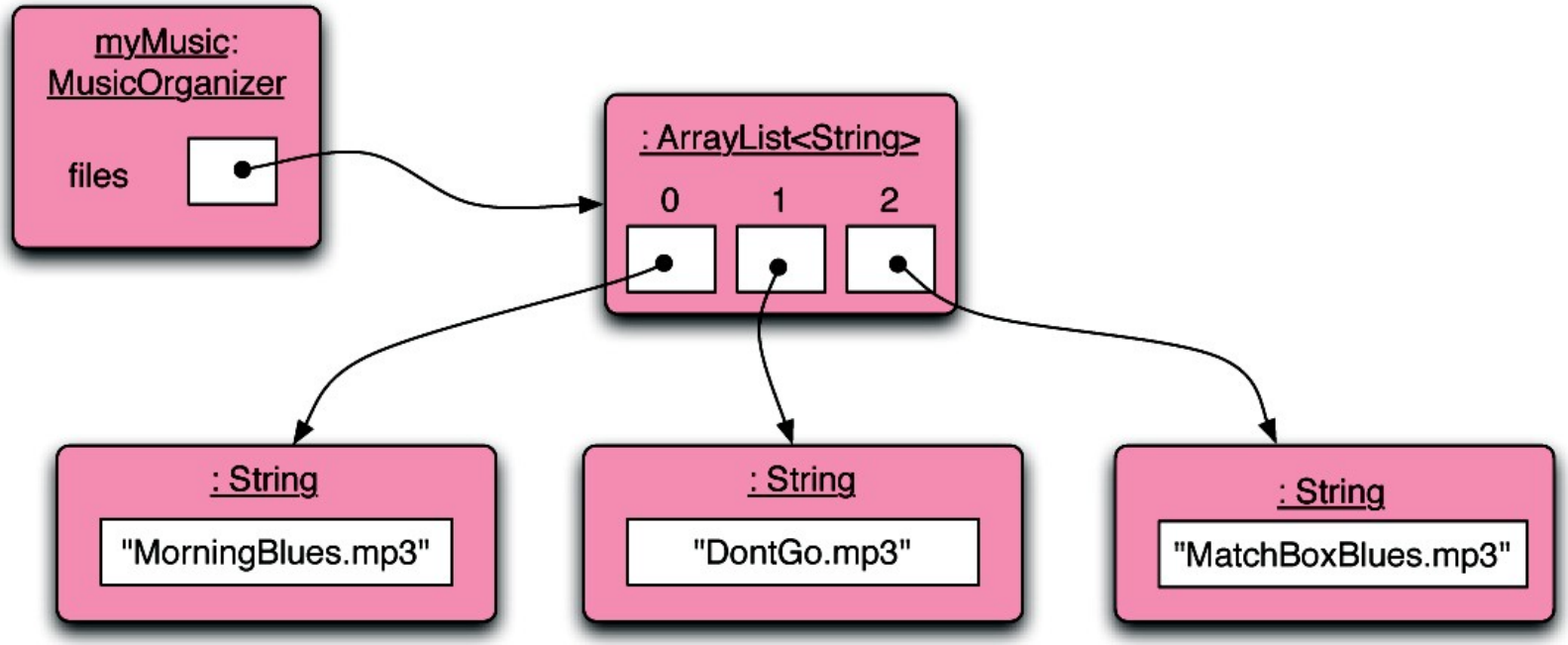


```
import java.util.ArrayList;

public class MusikSammlung
{
    // Eine ArrayList, in der die Namen
    // von Audiodateien gespeichert
    //werden können.

      private ArrayList<String> dateien;

    /**
     * Konstruktor: Erzeuge eine
     * MusikSammlung.
     */
    public MusikSammlung()
    {
        dateien = new ArrayList<>();
    }
[...]
```

```
[...]

/**
* Füge der Sammlung eine Datei hinzu.
* @param dateiname die hinzuzufügende Datei
*/

public void dateiHinzufuegen(String
dateiname)
{
    dateien.add(dateiname);
}

/**
* Liefere die Anzahl der Dateien.
* @return die Anzahl der Dateien in dieser
Sammlung
*/
public int gibAnzahlDateien()
{
    return dateien.size();
}

[...]
```

Hinter den Kulissen III

Indexnummern



Ausgeben

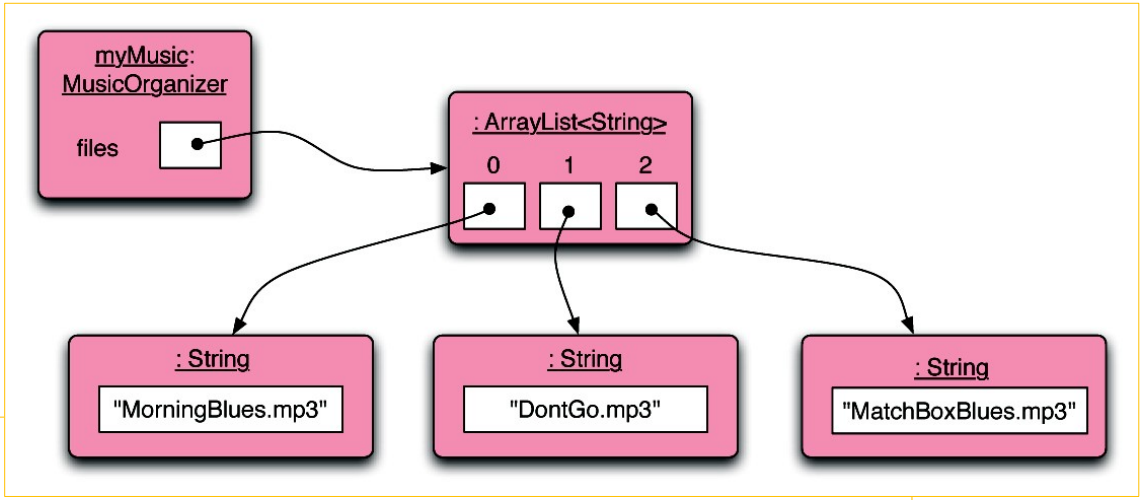
```
/**
* Gib eine Datei aus der Sammlung auf die Konsole aus.
* @param index der Index der Datei, deren Name ausgegeben
* werden soll
*/
public void dateiAusgeben(int index)
{
    if(index >= 0 && index < dateien.size()) {
        String dateiname = dateien.get(index);
        System.out.println(dateiname);

    }

}
```

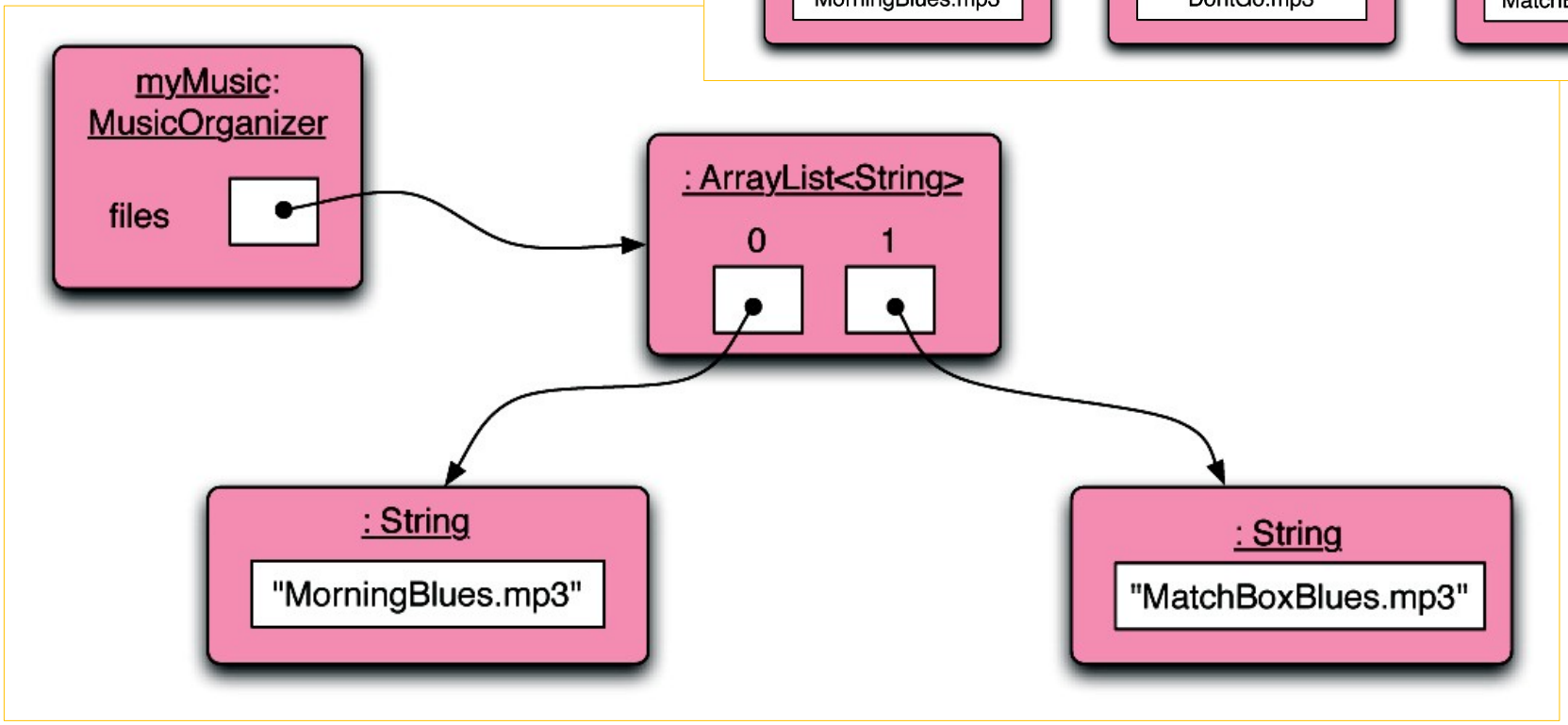
Warum?

Macht ein „else“-Zweig Sinn?

Hinter den Kulissen III

Indexnummern werden durch Entfernen/Einfügen u.U. verändert.





Algemeines zu Indizes

- Nächster: `aktueller_index + 1`
- Voriger: `aktueller_index - 1`
- Letzter: `list.size() - 1`
- Die ersten drei: `0,1,2`