Universität Bremen

Praktische Informatik 1
Objektsammlungen

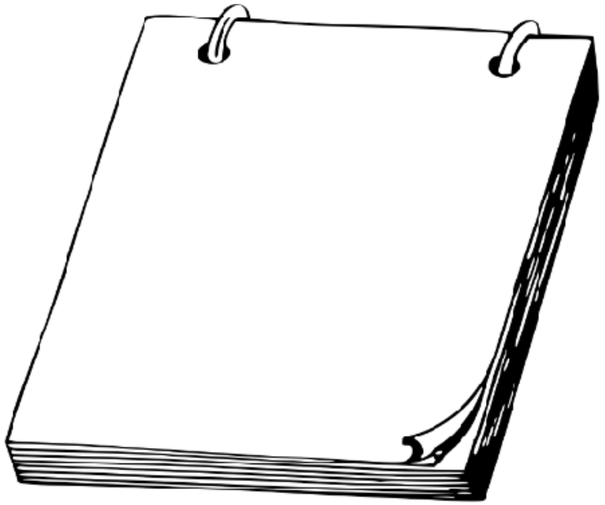
Thomas Röfer

Cyber-Physical Systems
Deutsches Forschungszentrum für Künstliche Intelligenz

Multisensorische Interaktive Systeme
Fachbereich 3, Universität Bremen

Bild: ©Michael Ihle

Projekt: Notizbuch



- Notizen können gespeichert werden
- Anzahl der Notizen unbegrenzt
- Einzelne Notizen können angezeigt werden
- Anzahl der Notizen kann abgefragt werden
- Analog zum Projekt „Musiksammlung“ in aktueller Auflage des BlueJ-Buchs

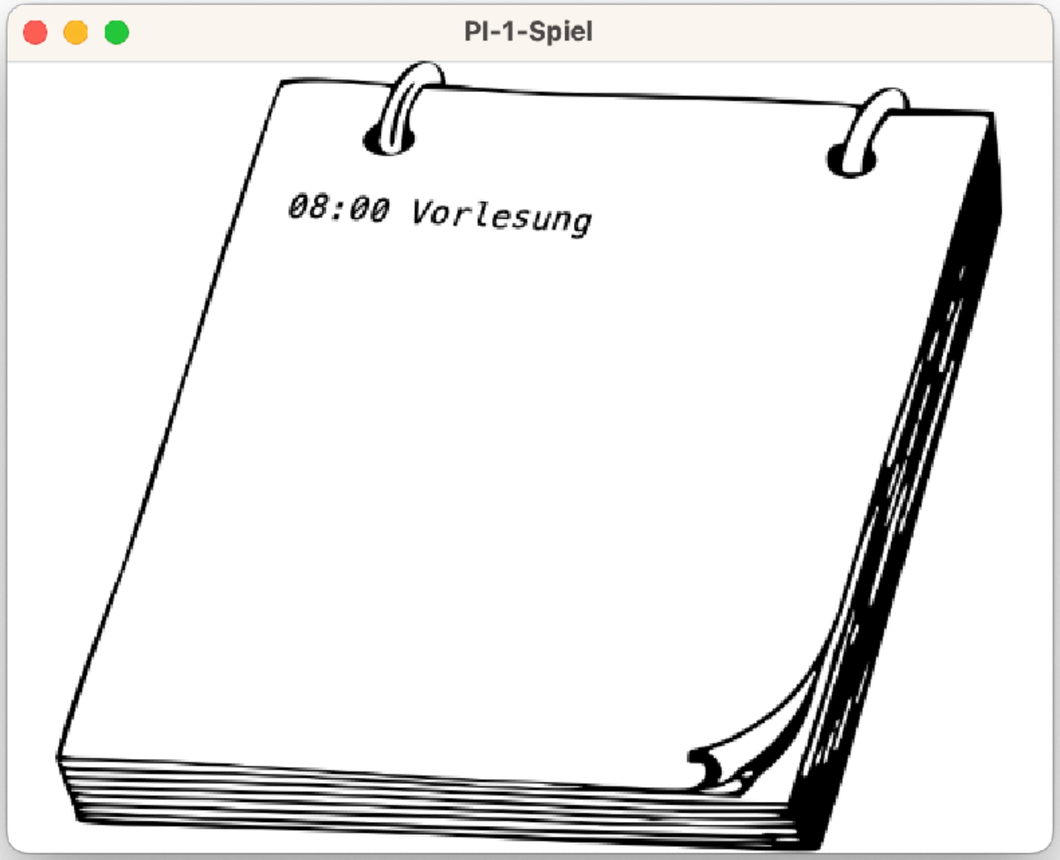
Universität Bremen

Klassenbibliothek

- Bibliothek nützlicher Klassen
  - Rad nicht neu erfinden!
- Bibliotheken in Java: **Pakete** (Packages)
  - Darüber gibt es seit Java 9 noch **Module**, hier interessiert uns nur **java.base** (und später noch **java.desktop**)
- **Sammlungen** sind Klassen von Objekten, die eine **beliebige Anzahl** anderer Objekte **enthalten** können
- Sammlungen finden sich in Java im Paket **java.util**

Universität Bremen

Notizbuch: Demo



Universität Bremen

Pakete (Packages)

- **Klassen** fassen **Methoden** und **Attribute** zusammen
- **Pakete** fassen **Klassen** und weitere **Pakete** zusammen
- Vermeidung von **Namenskonflikten**
- Mit **import** kann man **einzelne Klassen** oder **alle Klassen** aus einem Paket in den aktuellen **Namensraum** holen

```
class Notizbuch
{
    private final java.util.ArrayList<String> notizen
            = new java.util.ArrayList<>();
```

```
import java.util.*;
```

```
import java.util.ArrayList;

class Notizbuch
{
    private final ArrayList<String> notizen
            = new ArrayList<>();
```

Universität Bremen

Sammlungen (Collections)

- Typ der Sammlung festlegen: **ArrayList**
- Typ der enthaltenen Elemente festlegen: **<String>**
  - Wir sagen: **Array-Liste von Strings**
- Typ

```
ArrayList<String> notizen;
```

- Konstruktoraufruf

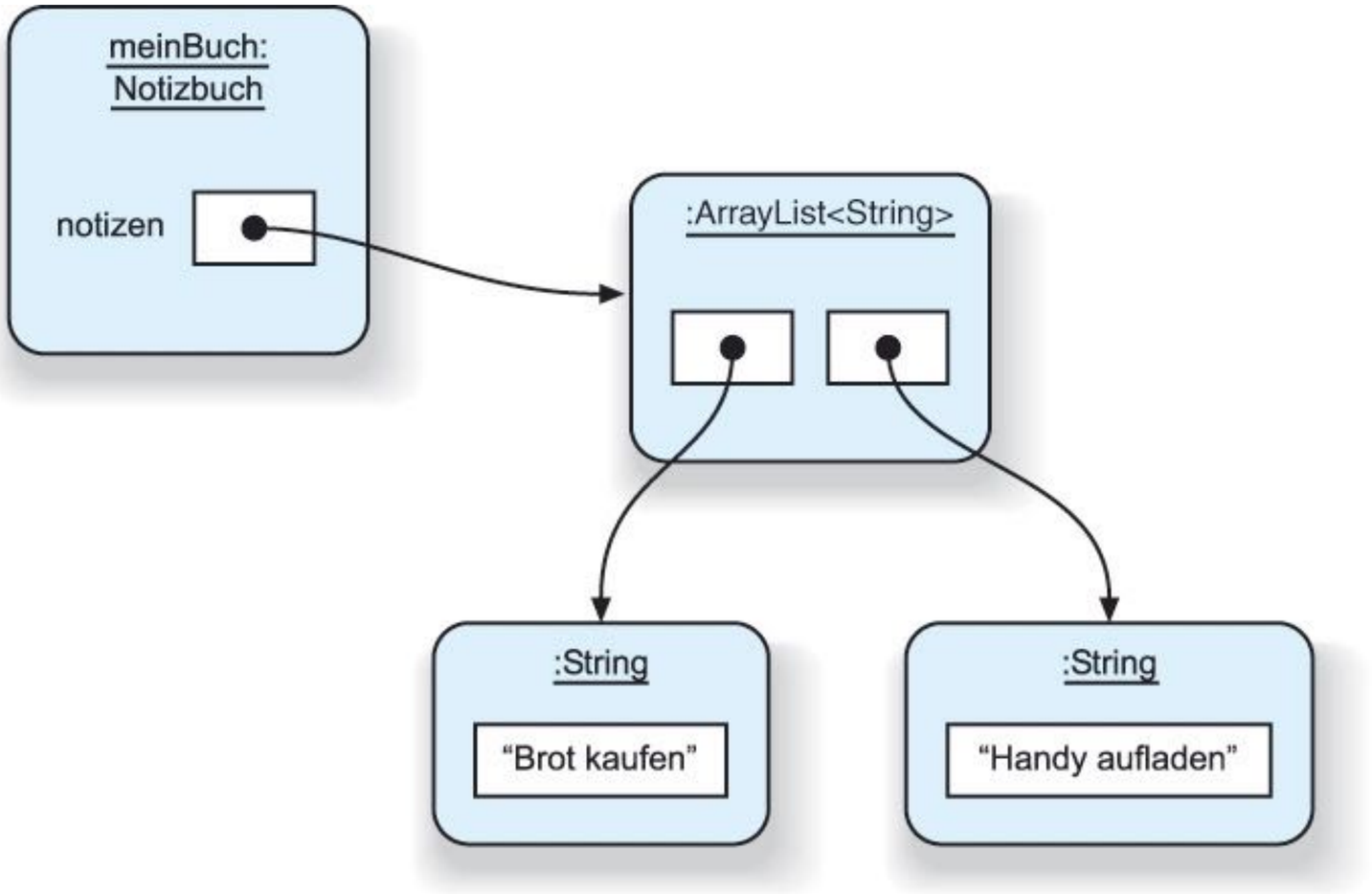
```
notizen = new ArrayList<String>();
```

  - Oder kürzer

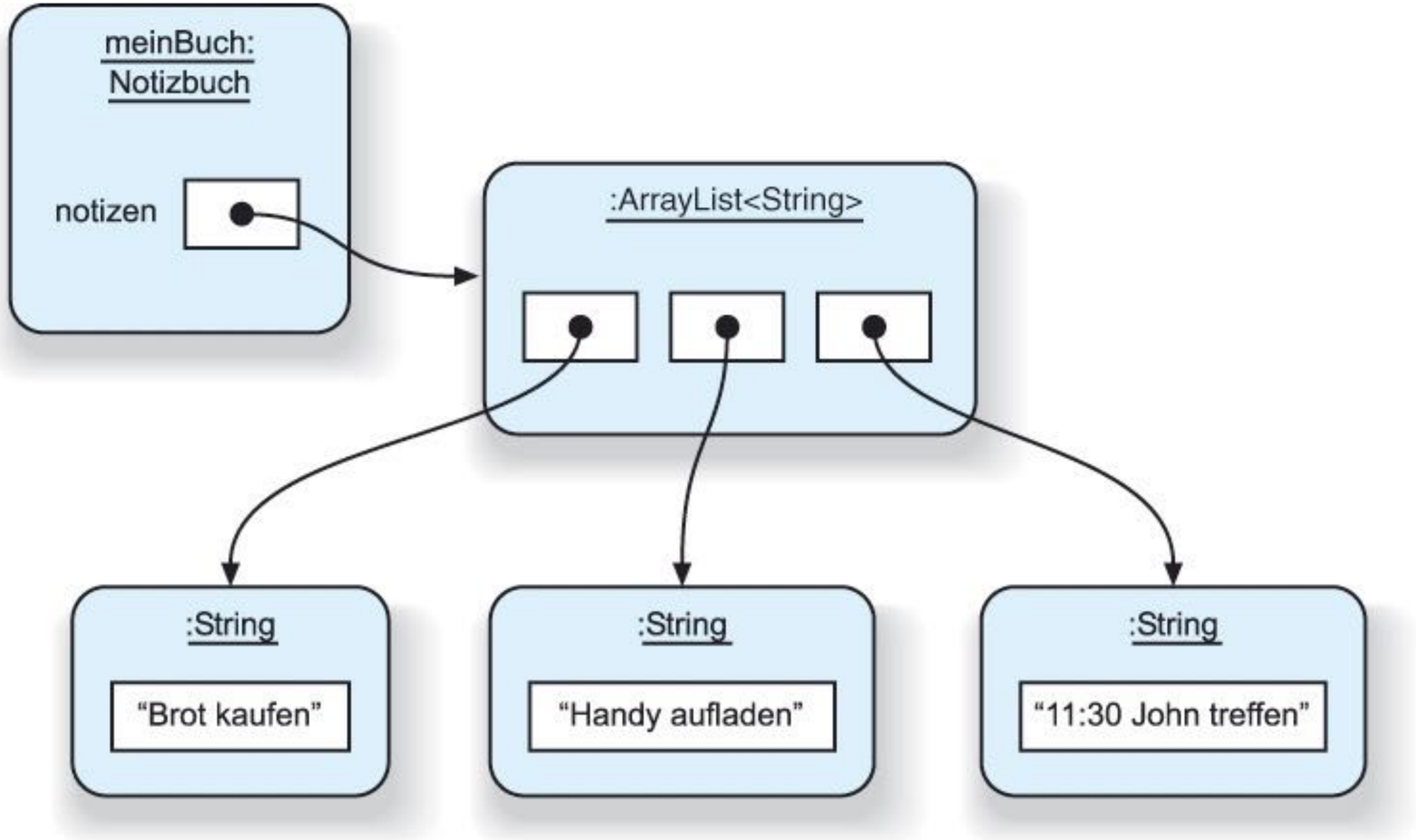
```
notizen = new ArrayList<>();
```

Universität Bremen

Dynamische Sicht



Notiz hinzufügen



Universität Bremen

ArrayList: Merkmale

- Kapazität bei Bedarf vergrößerbar
- Zählt Anzahl der Elemente: **size()**
- Reihenfolge der Elemente wird beibehalten, sind über Index ansprechbar
- Details verborgen
  - Ist das ein Problem?
  - Behindern fehlende Details die Nutzung?

Universität Bremen

Generische Klassen

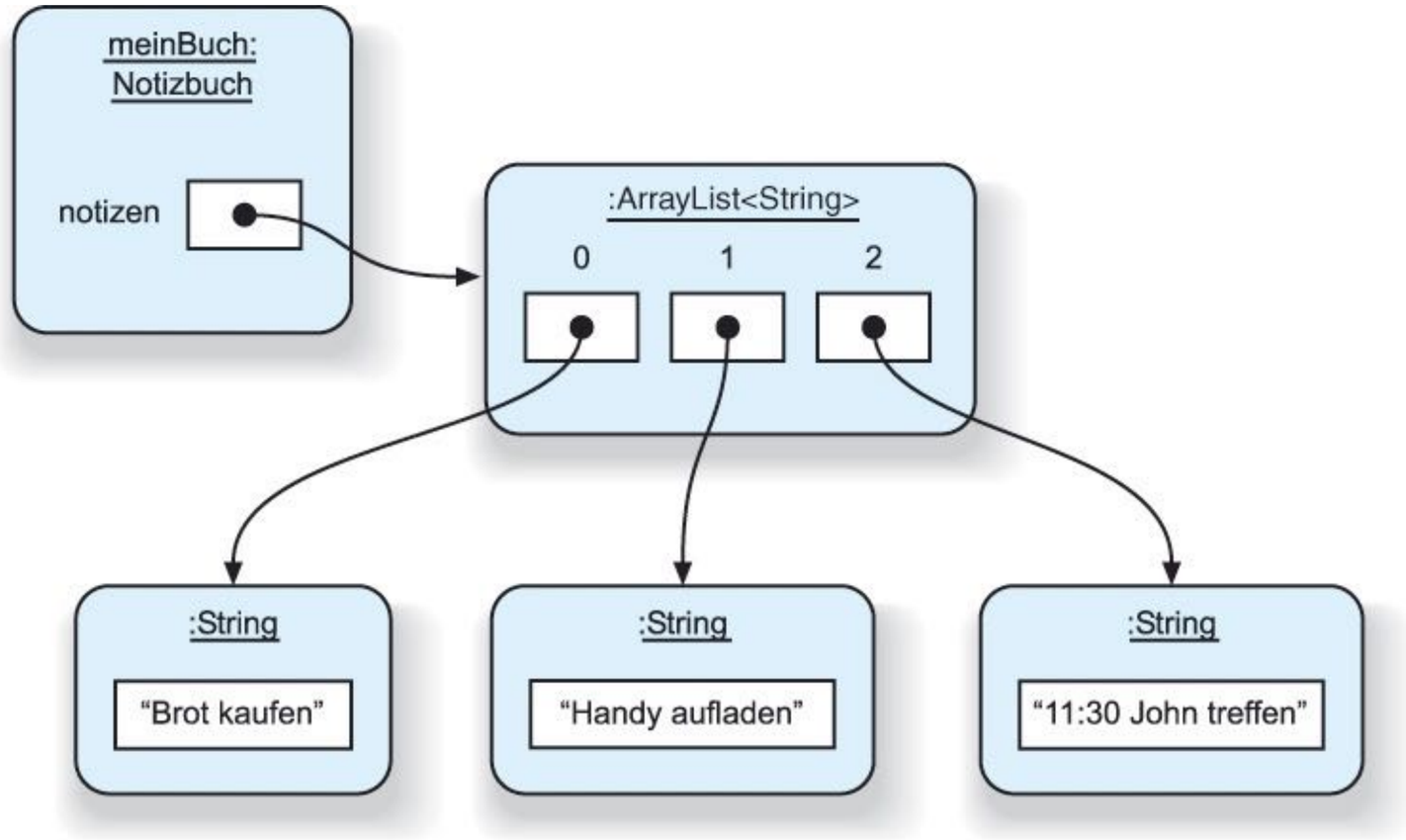
- **ArrayList** ist eine **parametrisierte** bzw. **generische** Klasse
- Ein **Typparameter** spezifiziert den Typ (Klasse) der Elemente
  - **ArrayList<GameObject>** ...
  - **ArrayList<String>** ...
- Achtung: Java erlaubt auch die Nutzung des Namens der generischen Klasse ohne Typparameter

```
ArrayList a = new ArrayList();
a.add("String"); // Ok
String s = a.get(0); // Fehler
```

  - Dies führt zu Fehlern bei der Nutzung der von diesem Typ deklarierten Variablen
  - Manchmal zeigt BlueJ auch die Warnung "... uses unchecked or unsafe operations" an

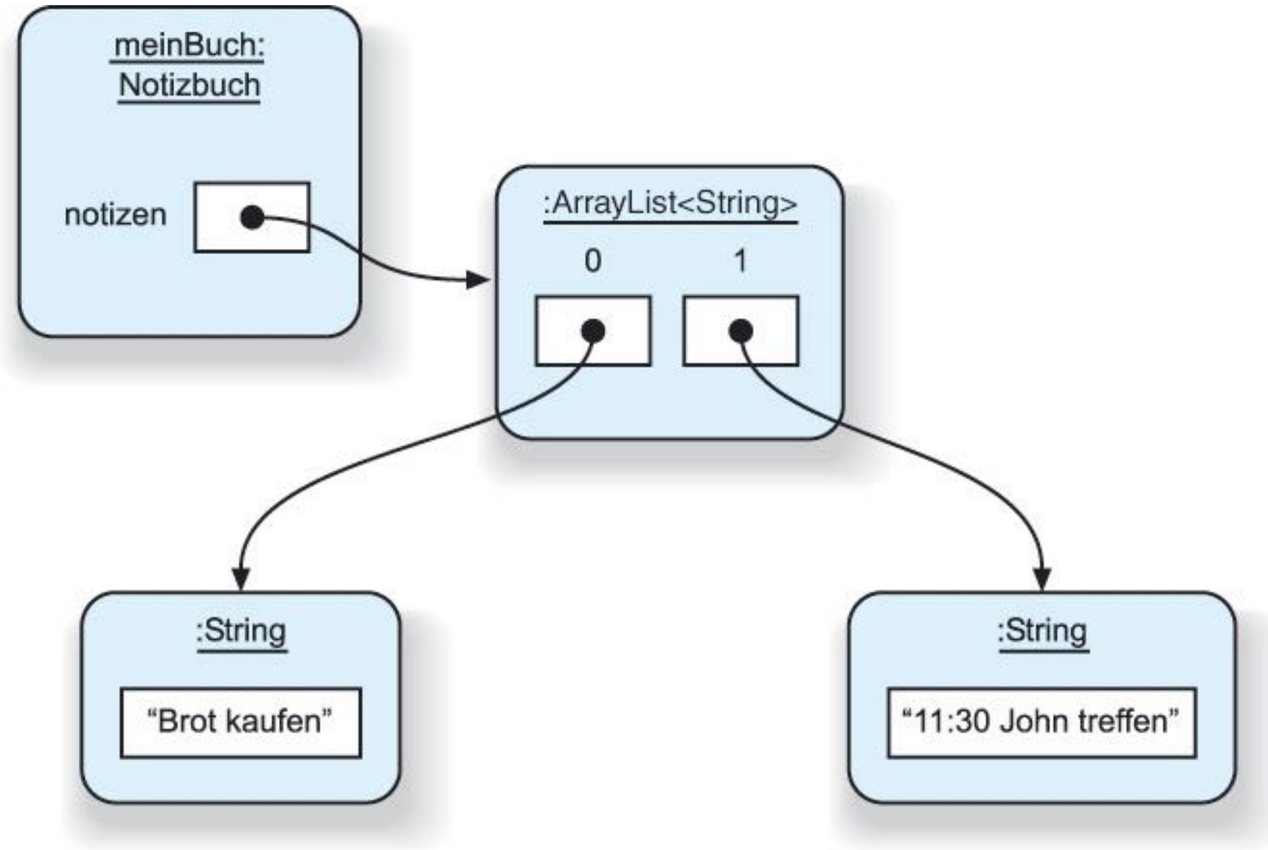
ArrayList: Indexnummerierung

- Elemente werden über Index adressiert
  - Index von **0** bis **size() - 1**
  - Schreiben: **set(***index*, *wert***)**
  - Lesen: **get(***index***)**
  - Index außerhalb des gültigen Bereichs → Fehler
- Anfügen: **add(**wert**)**



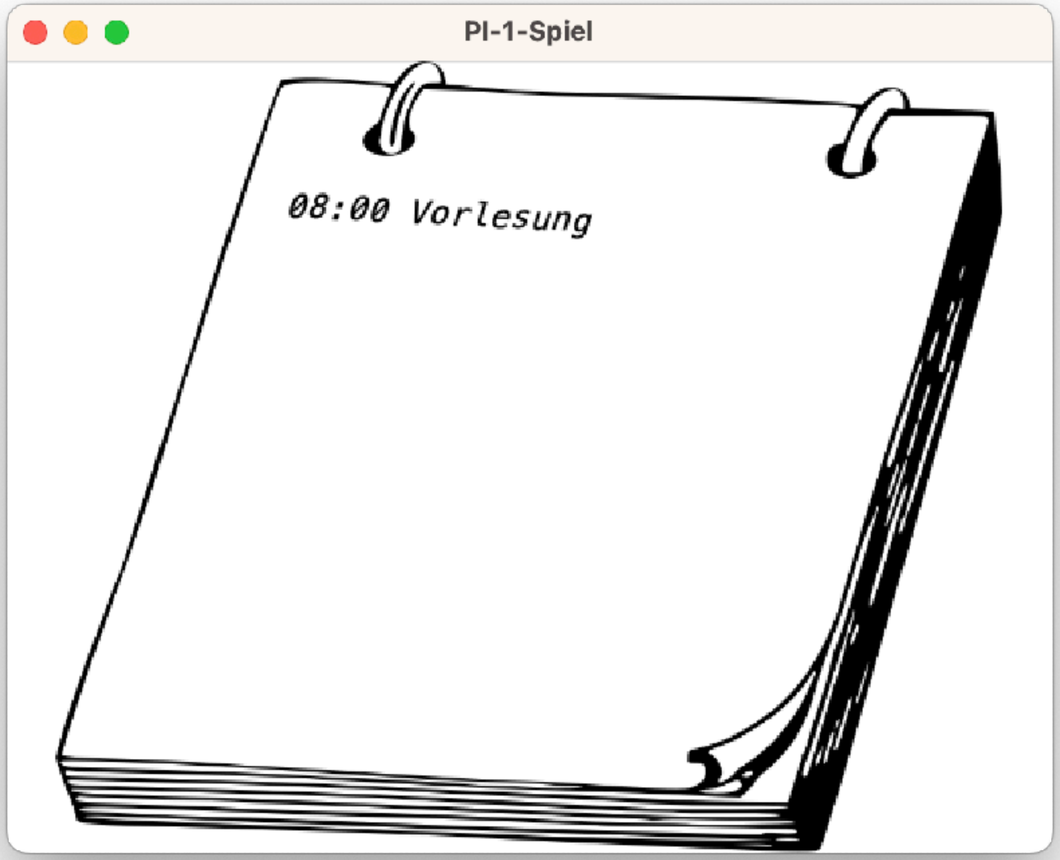
ArrayList: Elemente entfernen

- **remove(***index***)**: Entfernen
  - Alle Elemente hinter dem gelöschten ändern ihren Index!
  - **remove** gibt das gelöschte Element zurück



Universität Bremen

Notizbuch: Demo 2



Universität Bremen

Aufzählungsschleife

- Führe eine Folge von Anweisungen für jedes Element einer Sammlung aus
- Analogie zu Mengen in der Mathematik, z.B. $\sum_{n \in N} n^2$
- Führt Schleifenrumpf **size()**-mal aus
- In jedem Durchlauf enthält die Elementvariable ein anderes Element der Sammlung
- Das Durchlaufen folgt der Ordnung in der Sammlung

```
void notizenAusgeben()
{
    for (final String notiz : notizen) {
        schreibe(notiz);
    }
}
```

```
for (Elementtyp element : sammlung) {
    Schleifenrumpf
}
```

Universität Bremen

Suchschleife

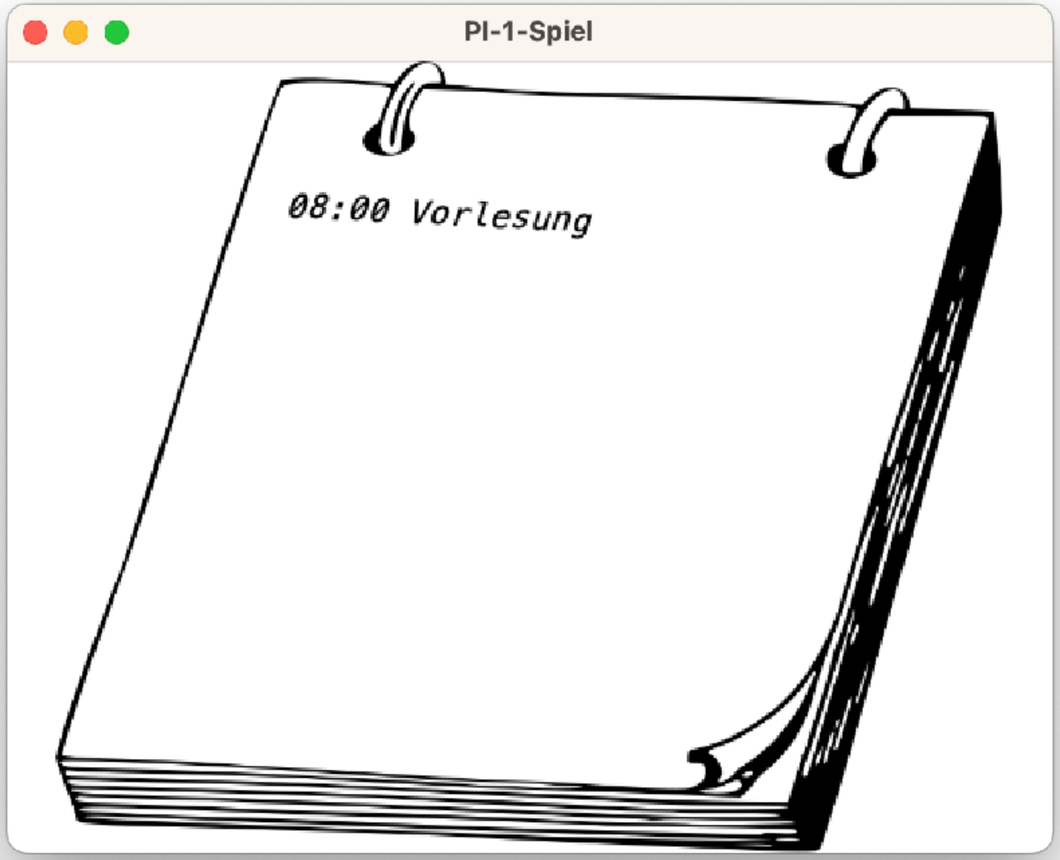
- Durchsucht eine Sammlung nach dem ersten Element, das eine bestimmte Bedingung erfüllt
- **Erfolgsfall**: Ein Element erfüllt die Bedingung
- **Misserfolgsfall**: Schleife läuft ohne Eintreten des Erfolgsfalls durch

```
String gibNotizMit(final String suchtext)
{
    for (final String notiz : notizen) {
        if (notiz.contains(suchtext)) {
            return notiz;
        }
    }
    return null;
}
```

```
for (Elementtyp element : sammlung) {
    if (element erfüllt Bedingung) {
        return Erfolg;
    }
}
// Kein Element erfüllte Bedingung
return Misserfolg;
```

Universität Bremen

Notizbuch: Demo 3



Universität Bremen

Iteratoren

- Dienen zum effizienten Durchlaufen von Sammlungen
- Haben als Parameter denselben Typ wie die Sammlung, die sie durchlaufen
- Werden von der Methode **iterator()** einer Sammlung zurückgeliefert
- Bieten u.a. die drei Methoden
  - **hasNext()**: Gibt es ein weiteres Element?
  - **next()**: Gib nächstes Element
  - **remove()**: Entferne das Element, das von **next()** zurückgeliefert wurde

```
void entferneNotizenMit(final String suchtext)
{
    final Iterator<String> i = notizen.iterator();
    while (i.hasNext()) {
        if (i.next().contains(suchtext)) {
            i.remove();
        }
    }
}
```

Universität Bremen

Zusammenfassung der Konzepte

- **Klassenbibliothek** und **Paket**
- **Sammlung**
- **Aufzählungsschleife** und **Suchschleife**
- **Iterator**