Woche 2

Modul 404

Agenda

- 🧠 Repetition Fachklassen
- ⌨️ Repetition Arrays (PDF)
- 🧠 Konzept Static
  - ⌨️ Aufgabe Starterklasse
  - ⌨️ Aufgabe Temperaturkonvertierer
- 🧠 Einstieg in Swing
  - ⌨️ Aufgaben JFrame
  - ⌨️ Aufgabe JFrame Komponenten

🧠 Repetition Fachklassen

🔗

- **Fachklassen** beinhalten die **generalisierte Logik** für ein Fachproblem
  - *In unserem Fall wäre das Fachproblem die Kontoverwaltung*
- Mit **Fachklassen** lassen sich **Programme entkoppeln**
  - **einfacherer Wartbarkiet**
  - **besserer Testbarkeit**
  - **besserer Qualität**
- Nennen wir es 🍣 **Sushi-Code**, das Gegenteil von 🍝 *Spaghetti-Code*

⌨️ Repetition Arrays (20min) 🔗

Im folgenden PDF ist beschrieben wie man in Java mit Arrays, also Listen von Datentypen, arbeitet.

- Repetition Arrays (📖 PDF) bitte genau studieren
- Diese **Aufgaben sind Optional** und müssen im **Selbststudium** gemacht werden.
- Am Ende der Sildes wird auf eine weitere Art von Arrays (Listen) eingeganen.
    - Diese ist **nicht Pflicht** jedoch häufig einfacher.

🦸 Dies ist eine Grundlage, welche Ihr im allgemeinen begreifen müsst um die beiden LBs mit einer `6` zu bestehen

Was kann `static`? 🪨 <- in Stein gemeisselt

- Kann ohne `new` aufgerufen werden.
- Kann wiederum andere `static` Methoden aufrufen
- Kann `static` Variablen verwenden (diese können nicht geändert werden!)
- Kann mit `new` ein `Objekt` / `Instanz` einer beliebigen Klasse erstellen.

Für was sind `static` Methoden gut?

- Die Java `main` Methode ist immer `static` (Entrypoint)
- Generelle/Universelle Helfermethoden 🧰 **ohne Datenstand**
  - Z.B. die Java Klasse `Math` ist komplett statisch `Math.sqrt(64);`

`Instanz-Methoden` ? 🐣 <- Es lebt!

- Können auf **Instanz-Variablen zugreifen**
- Können andere **Instanz-Methoden ausführen**
- Es können von einer **Klasse mehrere Instanzen** erstellt werden
  - Diese sind **gekapselt** und **besitzen einen individuellen Datenstand**
- Können nur über ein `Obejkt` / `Instanz` ausgeführt werden (`new`)

```
// Alle Objekte haben ihren eigen "gekapselten" Datenstand
Account accountObject1 = new Account();
Account accountObject2 = new Account();
Account accountObject3 = new Account();
```

🧠 Static vs Instanz-Methoden

Eine `static` Methode einer `Klasse` kann direkt aufgerufen werden, ohne dass ein `Object` / `Instanz` der Klasse erstellt werden muss.

```
public class MixedExample {
    private static final double PI = 3.14; // Konstante, kann nicht geändert werden!
    private String greeting = "Hello";     // Instanz-Variablen, kann geändert werden

    public static double staticCircle(double radiant) {
        return radiant * radiant * PI; // Kann auf `PI` zugreifen nicht aber auf `greeting`
    }

    public String instanceGreeting(String name) {
        return greeting + " " + name;  // Kann auf `greeting` zugreifen
                                       // Könnte theoretisch auch auf `PI` zugreifen
    }

    public void setGreeting(String greeting) { this.greeting = greeting; }
}
```

🧠 Verwenden von `MixedExample`

```java
public class Starter {
    // Startpunkt des Programms, ist immer static!
    public static void main(String[] args) {

        // Statische Methoden können ohne new ausgeführt werden!
        double circle = MixedExample.staticCircle(1.5d);

        // Um instanceMethoden aufzurufen, muss zuerst eine Instanz erstellt werden
        MixedExample mixedExampleInstance = new MixedExample();
        String greeting = mixedExampleInstance.instanceGreeting("Lukas");
        // Wert ist "Hallo Lukas";

        mixedExampleInstance.setGreeting("Ciao") // Objekt ändern
        greeting = mixedExampleInstance.instanceGreeting("Lukas");
        // Wert ist "Ciao Lukas";
    }
}
```

⌨ Starterklasse (20min) 🔗

🔗 Aufgabe Starterklasse öffnen

- Es ist eine *gute Praxis* in der `main` Methode nicht viel Logik zu implementieren.
- Bestenfalls besteht die **Methode** `main` nur mit der **Instanzisierung einer Applikations Klasse** welche die eigentliche App verwaltet.

```
public class Starter {
    public static void main(String[] args) { // Startpunkt des Programms, ist immer static!

    MyNewShinyApp app = new MyNewShinyApp(); // `new` ist innerhalb von `static` erlaubt
    app.start(); // starten der eigentlichen App
}
```

⌨ Refactoring Temperaturkonvertierer

🔗 Aufgabe Temperaturkonvertierer

- Es existiert ein Temperaturkonvertierer der **als Objekt** implementiert wurde.
- Ist dies hier sinnvoll oder gibt es eine bessere Art diesen zu implementieren?
- Dies sollt Ihr euch in der Aufgabe überlegen. **Könnte hier `static` was vereinfachen**?

Einstieg `Swing` 🏞️

`Swing` ist eine **Bibliothek für grafische Oberflächen**. Dadurch ist es möglich die bis jetzt textuelle Benutzerinteraktion mit grafischen Elemente, auch `GUI - Graphical User Interface` genannt, umzusetzen.

🚨 Achtung!

- Das Verständis davon ist essenziell um im **LB1** eine gute Note zu erhalten!
- Auf dieser Grundlage wird auch das Projekt (**LB2**) umgesetzt!

☝️ Nehmt euch also die *Zeit* dies zu verstehen!

Hilfe im Web! 🚀

Es ist immer gut sich im Internet weiterzubilden (🤔 zumindest für IT-Themen)

www.java-tutorial.org ist eine Webseite die Java, sowie `Swing` im detail erklärt:

- Deutsches `Swing`-Tutorial als Zusatzinfo für Interessierte
  - **Relevante Themen**: JFrame, JLabel, JButton, JTextField, JPanel
  - Auch wichtig, wird nächste Woche behandelt Event-Handling

🦸 **Ich würde alle in den Aufgaben verwendeten Klassen hier nachschlagen**

Swing: `JFrame`, JFrame auf Java Tutorial

Möchte man eine Klasse als `Fenster` erstellen, muss man der Klasse die Java Klasse `JFrame` **vererben**. Dies geht mit dem Ausdruck `extends JFrame`

```
public class PureWindow extends JFrame {
    public void start() {                           // oder auch showDialog, oder was gefällt
        setLayout(null);                            // Standard Layout deaktivieren
        setDefaultCloseOperation(EXIT_ON_CLOSE);  // Beim schliessen des Fensters, das ganze Programm beenden
        setSize(300, 300);                          // Grösse vom Fenster festlegen
        setTitle("Ich bin der Fenster Titel");    // Titel des Fensters festlegen
        setVisible(true);                           // Fenster sichtbar machen
    }
}
```

ℹ️ *Was genau Vererbung ist, und wie man es selbst verwendet ist Teil eines späteren Moduls.* ***Hier wenden wir es einfach Mal blindlings an.***

`JFrame` Cheat Cheet 💡

```
setLayout(null); // Standard Layout deaktivieren
setDefaultCloseOperation(EXIT_ON_CLOSE); // Beim schliessen des Fensters, das ganze Programm beenden
setSize(300, 300); // Grösse vom Fenster festlegen
setTitle("Ich bin der Fenster Titel"); // Titel des Fensters festlegen
setVisible(true); // Fenster sichtbar machen

JLabel label = new JLabel("Beschriftung"); // Ein Label
label.setBounds(x, y, wigh, height) // Bestimmen wo sich das Label befindet
add(label) // Label hinzufügen

JTextField textfield = new JTextField(); // Ein Textfeld
textfield.setBounds(x, y, wigh, height) // Bestimmen wo sich das Textfeld befindet
add(textfield) // Textfeld hinzufügen

JButton button = new JButton("press me"); // Ein Button
button.setBounds(x, y, wigh, height) // Bestimmen wo sich der Button befindet
add(button) // Textfeld hinzufügen
```

`Swing` Aufgaben 📝

Lest die beiden PDFs auf der Modulwebseite genau durch und bearbeiten Sie die **Aufgaben 1 - 4**.

- Einstiegs Aufgaben PDF 📖
- Einstiegs Musterlösungen 📖

🦸 Versucht es immer zuerst ohne Musterlösung!

Abschluss / Lernjournal

🦸 Die Klasse `List` oder auch `ArrayList`

Primitive Java Arrays sind umständlich und können häufig durch die Klasse `List<Datetyp>` oder `ArrayList<Datentype>` abgelöst werden.

Hier ein Beispiel für eine `List<String>`, also eine Liste von Wörtern:

```
List<String> list = Arrays.asList("Element 1", "Element 2");
list.add("element 3"); // es können Elemente dynamisch hinzugefügt werden

// über die gesamte Liste iterieren ist viel einfacher als bei "primitiven" Arrays
for (String element : list) {
  System.out.println(element);
}

// Auf ein Element zugreifen mit `.get` (es startet bei 0, nicht bei 1)
System.out.println(list.get(0));
```