Woche 4

Modul 404

Agenda

Input

- Example 10 Normal Nor
- **L** Formatierung
- 🖋 UML Klassendiagram
- Konstruktor

Aufgaben

- Account UML
- 👺 DiceGame
- Konstruktor
- 🙎 Weitere Swing-Aufgaben

Kontoverwaltung in Swing



Musterlösung erläutern

Formatierung

- 9 Wichtig um die Lesbarkeit zu erhöhen
- 🙎 Ist ein Zeichen von Professionalität
- Pilft Flüchtigkeitsfehlern vorzubeugen

📐 Formatierung / Unsere Regeln! 🚨

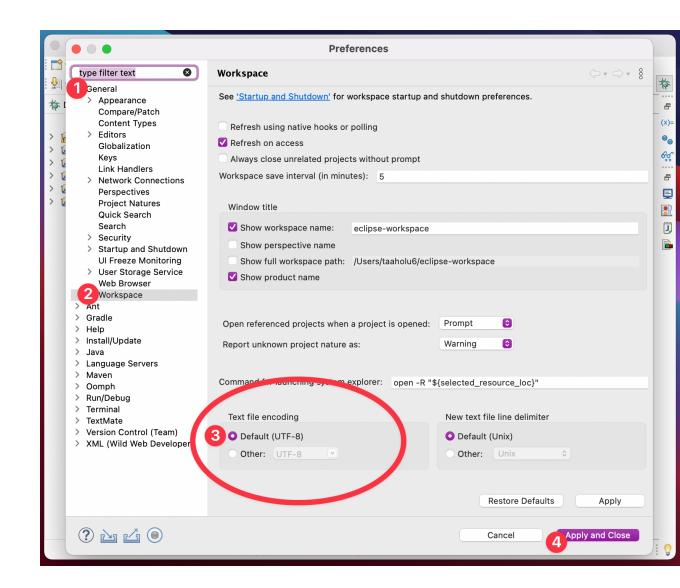
- Wir verwenden keine Umlaute im Code (Ausnahmen sind Kommentare)
- Jede Klasse
 - beginnt mit einem Grossbuchstaben
 - ∘ hat einen AusdrucksstarkenNamen in 🐪 UpperCamelCase 🔗
- Jede Methode
 - beginnt mit einem Kleinbuchstaben
 - ∘ hat einen ausdrucksstarkenNamen in 🐪 lowerCamelCase 🔗
- Blöcke {} werden eingerückt (Ctrl-Shift-F Command-Shift-F)
- Standard-Encoding UTF-8: Preferences > General > Workspace -> UTF-8



Eclipse Preferences öffnen:

- 1. General auswählen
- 2. Workspace auswählen
- 3. Default (UTF-8) setzen
- 4. Speichern

Sonst compiliert euer Code auf meinem Mac nicht!



Formatierung / Auftrag



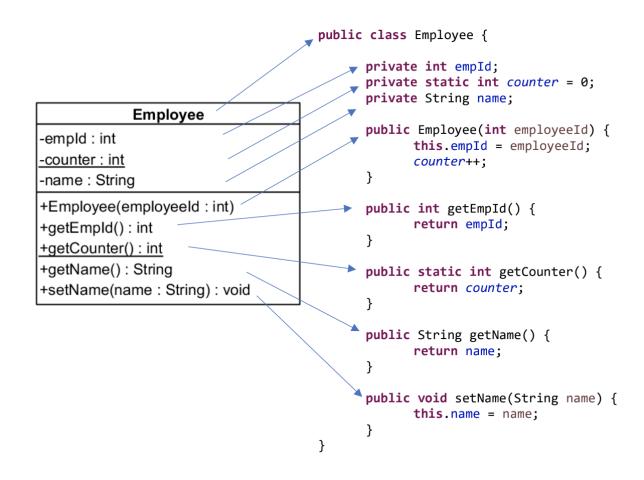
- 1. Lesen Sie das Konzept Formatierung gut durch!
- 2. Lösen Sie folgende Aufgabe Formatierung

UML - Unified Modeling Language

- 🖋 Visualisierung von Code und Abläufen
- Z Sollte grafisches programmieren ermöglichen UML -> Code
- - 。Code ist konfortabler zu schreiben als UML zu malen 🧐

UML Klassendiagram

- Eine Klasse ist ein Rechteck
- Klassenname ist zentrierter Titel
- Sichtbarkeit
 - ∘ ist private
 - o + ist public
- Obenhalb: Instanzvariablen
- Unterhalb: Instanzmethoden
- <u>Unterstrichen</u>: static



UML Klassendiagram / Methoden

UZ-

+|-methodennamen(variablenNamen: DatenTyp) : returnDatenTyp

UML	Java Signatur
+setName(name : String)	<pre>public void setName(String name)</pre>
+getName() : String	<pre>public String getName()</pre>
<u>+sum(a : int, b: int) : int</u>	<pre>public static int sum(int a, int b)</pre>
-secret(key : String) : String	private String secret(String key)

UML Klassendiagram / Variablen

uel

+|- variablenNamen : Datentyp

UML	Java
-name : String	private String name;
+year : int	public int year;
+PI : double	public static double PI;



Klasse verwendet ein new Objekt

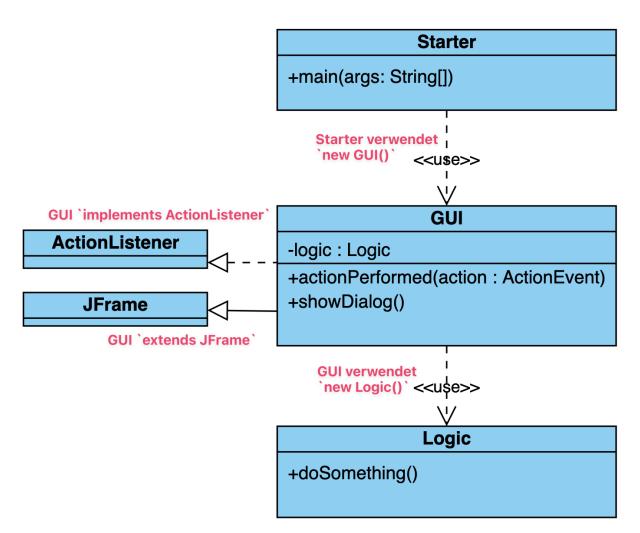
• gestrichelter Pfeil

Klasse implements ein Interface

gestrichelter Pfeil mit
 Dreiecksspitze

Klasse extends eine Klasse

durchgezogener Pfeil mit
 Dreiecksspitze





- diagrams.net früher draw.io
- Visual Paradigm Online
 - Visual Paradigm finden Sie auf unseren Windows VMs
- Mermaid 🕹
 - wird auf dieser Seite verwendet
 - Mermaid Dokumentation
 - Mermaid Live im Browser

Automatisches Generieren

- IntelliJ Diagrams jedoch nicht 100% UML Standard!!
- ObjektAid for Eclipse
 - Video mit Installationsanleitung

UML-Klassendiagram / Auftrag

- 1. lesen Sie das Konzept UML
- 2. Erstellen Sie ein UML-Klassendiagram der Fachklasse Account
- 3. Implementieren Sie das DiceGame
 - o Diese Aufgabe würde ich besonders gut anschauen! 😉
- 🙎 Weitere Swing-Aufgaben

Konstruktor

- Methodenname ist immer gleich wie die Klasse
- ist nicht explizit aufrufbar
- wird ausgeführt wenn ein Objekt erstellt wird (💡 in Verbindung mit 🛛 new)
- hat keinen Rückgabewert
- es können mehrere Konstruktoren bestehen (💡 andere Anzahl Parameter)
- werden keine Parameter angegeben, nennt man ihn Standardkonstruktor
- dient dazu das Objekt mit gültigen Werten zu initialisieren
- Ŭ Video auf Youtube über Konstruktoren

Konstruktor / Beispiel

```
public class MyClass {
private String name; // Instanzvariable die Inizialisiert werden muss!
private int year = 2000; // Instanzvariable mit Standardwert
public MyClass() { // Standardkonstruktor (ohne Parameter)
  this.name = "Startwert"; // `name = "Startwert"` ohne `this` ist auch gültig
public MyClass(String name) { // konstruktor mit gleichnamigem parameter
  this.name = name; // `this` ist notwendig da gleichnamig
public MyClass(String aName, int year) { // Konstruktor mit zwei Variablen
  name = aName;  // `this` darf weggelassen werden (muss aber nicht!)
  this.year = year; // `this` ist notwendig da gleichnamig
```

Konstruktor / Verwendung

```
public class Starter {
public static void main(String[] args) {
  // Standardkonstruktor wird ausgeführt!
  MyClass myClass = new MyClass();
  // Konstruktor mit einem Parameter wird ausgeführt
  MyClass myClass2 = new MyClass("Neuer Startwert");
  // Konstruktor mit zwei Parameter wird ausgeführt
  MyClass myClass3 = new MyClass("Neuer Startwert", 2022);
```

2 Das nennt sich auch Methoden überladen und geht auch für normale Methoden

Konstrutkor / Auftrag

- 1. Lesen Sie das Konzept Konstruktor gut durch!
- 2. Lösen Sie folgende Aufgabe Konstruktor

Nächste Woche gibts einen Test!

- Poetails auf der Modulwebseite
- **2** Geht **alle Aufgaben** nochmals durch und **versteht Sie**!
- 🙎 Lernt ein UML-Klassendiagram in Java umzuwandeln 😌
- 🖢 Arrays sollte man anwenden können
- 🙎 Schaut euch die **DiceGame** Würfel-Logik genau an