Unterrichtsentwurf für den 1. Besuch der Schulleitung

Vor- und Nachname Lorenz Bung		
Schulanschrift (mit Telefonnummer) Walther-Rathenau-Gewerbeschule, Friedrichstr. 51, 79098 Freiburg. 0761/201-7942 Schulleiter/-in Renate Storm		
Mentor/-in Leonie Feldbusch	Ausbilder/-in Jochen Pogrzeba	
Datum 23.05.2025	Uhrzeit 11:30 – 12:15	
Klasse und Schulart E2FI3 – Berufsschule: Fachinformatiker, 2. Lehrjahr		Raum 025
Fach Software- und Anwendungsentwicklung (SAE)		

Thema des Unterrichts

Objektorientierte Programmierung: Polymorphie

1. Überblick und zentrales Anliegen

Thema	Objektorientierte Programmierung: Umsetzung des Konzepts der Polymorphie in einer objektorientierten Programmiersprache (Python)
Lehrplanbezug	Lernfeld 8: **Daten systemübergreifend bereitstellen** "Die Schülerinnen und Schüler ermitteln für einen Kundenauftrag Datenquellen und **analysieren** diese hinsichtlich ihrer Struktur […]." "Sie **entwickeln** Konzepte zur Bereitstellung der gewählten Datenquellen für die weitere Verarbeitung […]." "Die Schülerinnen und Schüler **implementieren** […] ihr Konzept mit vorhandenen sowie dazu passenden Entwicklungswerkzeugen und Produkten."
Zentrales Anliegen	Die SuS können das Konzept der Polymorphie mithilfe der objektorientierten Programmiersprache Python umsetzen und konkrete Beispiele implementieren. Weiterhin können die SuS mit UML-Diagrammen modellierte Sachverhalte, in denen Polymorphie zum Einsatz kommt, in Python-Code überführen.
Lehr-Lernarrangement	Eine Datei mit verschiedenen System-Log-Einträgen soll problemorientiert an das Thema der Polymorphie heranführen. Hierbei sollen die SuS sich eine Möglichkeit zur besseren Formatierung der Nachrichten überlegen. In der anschließenden Erarbeitungsphase wird die dafür benötigte Syntax in Python anhand des präsentierten Beispiels eingeführt und durch Programmierung am Lehrer-PC demonstriert. Es folgt eine Übungsphase, in der die SuS selbstständig eine Situation programmieren sollen, in welcher Polymorphie zum Einsatz kommt. Weiterhin sollen sie den entwickelten Code testen. Abschließend werden die Ergebnisse durch eine Präsentation durch die SuS vorgestellt, welche durch die anderen SuS sowie die Lehrkraft bei Bedarf ergänzt werden.

2. Lernziele und Kompetenzentwicklung

Die Lernziele der Unterrichtsstunde sind die Kenntnis des Prinzips der Polymorphie und deren Umsetzung in einer objektorientierten Programmiersprache (Python):

1. TZ: Die SuS *kommentieren* Code, in dem Polymorphie verwendet wird. (AFB I)

2. TZ: Die SuS *implementieren* Sachverhalte, bei denen Polymorphie zum Einsatz kommt, in einer objektorientierten Programmiersprache. (AFB II)

3. TZ: Die SuS *begründen* die Sinnhaftigkeit der Nutzung von Polymorphie (z.B. Strukturiertheit, Wartbarkeit…). (AFB III)

3. Unterrichtsverlaufsplan

Phase	Unterrichtsstruktur (mit Zeitplanung)	Lehrerhandeln	Schülerhandeln	Lernziele (fachliche und überfachliche)	Medien
Unterrichtseinstieg (11:30 – 11:35)	LSG, fragend-entwickelnd Begrüßung, Einstiegsproblem "System-Logs" 5 min	Begrüßung der SuS Präsentation der Problemsituation Sammeln von Ideen der SuS	Begrüßung Hineinversetzung in die Problemstellung	Motivationsaufbau Vorwissensaktivierung Herstellung eines geeigneten Lernklimas	Textdatei "Logs" digitale Tafel evtl. Kreidetafel (abhängig von Schülerbeiträgen)
Erarbeitung (11:35 – 11:43)	Lehrervortrag, darbietend Syntax Polymorphie in Python 8 min	"Vorprogrammieren" Erklären des Codes Beantwortung von Fragen Aufforderung an SuS: "Bitte beschreiben Sie in eigenen Worten, was wir gerade gemacht haben"	Zuhören Fragen stellen Beschreibung der Funktionalität des demonstrierten Codes	Syntax kennenlernen 1. TZ: Die SuS *kommentieren* Code, in dem Polymorphie verwendet wird. (AFB I)	Lehrer-PC: Code-Editor digitale Tafel
Übung und Anwendung (11:43 – 12:05)	Einzel- bzw. Partnerarbeit Aufgabe 1 "Formen" 22 min	Freischalten des Arbeitsblatts bzw. Upload-Ordners in Moodle Beantwortung von Fragen technische Hilfestellung	Bearbeitung der Aufgabe Verständnisfragen stellen ggf. gemeinsame Diskussion in Partnerarbeit nach Beendigung der Aufgabe: Upload der eigenen Lösung in Moodle	2. TZ: Die SuS *implementieren* Sachverhalte, bei denen Polymorphie zum Einsatz kommt, in einer objektorientierten Programmiersprache. (AFB II)	Arbeitsblatt "OOP: Polymorphie", Aufgabe 1 Merkzettel Polymorphie
Ergebnissicherung (12:05 – 12:15)	Schülerpräsentation, LSG Besprechung Aufgabe 1 "Formen" 10 min	Moderation von Meldungen Ergänzung der vorgestellten Lösung Beantwortung von Fragen	Präsentation der eigenen Lösung Ergänzung der vorgestellten Lösung Verständnisfragen stellen	3. TZ: Die SuS *begründen* die Sinnhaftigkeit der Nutzung von Polymorphie (z.B. Strukturiertheit, Wartbarkeit…). (AFB III) Stärkung der Präsentationsfähigkeiten	Schülerlösung zur Aufgabe 1 "Formen" via Moodle digitale Tafel

(Hinweise zur Ergebnissicherung werden in den Spalten Lehrer- bzw. Schülerhandeln eingetragen)

SAE	**Merkzettel**	Klasse:
L. Bung	**Polymorphie in Python**	Datum:

Problemsituation: Informationen und Fehlermeldungen werden gleich formatiert angezeigt.

```
System started
Disk not found
Connection established
Permission denied
```

Lösung: Wir schreiben zwei Klassen `InfoLog` und `ErrorLog`, die von der gemeinsamen Superklasse `LogEntry` erben. Die Formatierung (durch die Methode `display()`) wird durch die beiden Klassen unterschiedlich implementiert.

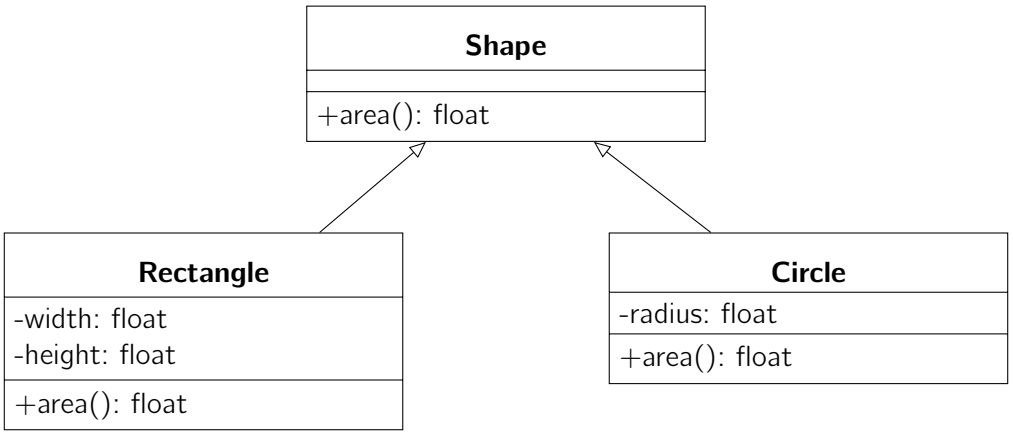
> **Polymorphie**
>
> Verschiedene Subklassen, die eine Methode von der Superklasse erben, können unterschiedliche Implementierungen der Methode haben. Damit ist es möglich, in verschiedenen Subklassen unterschiedliche Funktionalitäten bei Aufruf der Methode zu erreichen. Dieses Konzept nennt man in der objektorientierten Programmierung **Polymorphie**.

```python
class LogEntry:
  def __init__(self, message):
    self.message = message
  def display(self):
    print(self.message)

class InfoLog(LogEntry):
  def display(self):
    print(f"[INFO] {self.message}")

class ErrorLog(LogEntry):
  def display(self):
    print(f"[ERROR] {self.message} (!!!)")

log1 = InfoLog("System started")
log2 = ErrorLog("Disk not found")
log3 = InfoLog("Connection established")
log4 = ErrorLog("Permission denied")

log1.display()
log2.display()
log3.display()
log4.display()
```

SAE	**OOP: Polymorphie**	Klasse:
L. Bung		Datum:

Aufgabe 1: Formen in Python

a) Setzen Sie folgendes UML-Diagramm in Python um. Die `area()`-Methode der Klasse `Shape` soll immer den Standardwert 0 zurückgeben.



b) Testen Sie Ihre Implementierung: Erstellen Sie jeweils ein Rechteck und einen Kreis und lassen Sie den Flächeninhalt berechnen.

c) Ergänzen Sie die drei Klassen um eine Methode `circumference()`, die den Umfang der jeweiligen Form berechnet.

d) Erweitern Sie Ihre Tests aus Teilaufgabe b): Erstellen Sie mehrere Objekte der beiden Klassen, speichern Sie sie in einer Liste und nutzen Sie anschließend eine Schleife, um die Methoden `area()` und `circumference()` aufzurufen.