Kompetenzraster

Name:_____ Klasse:_____ Note:_____

Handlungsziel	Kompetenzen				
	1	2	3	4	5
Ein objektorientiertes Design nachvollziehen und durch Einsatz der Vererbung erweitern. **HZ1**	Ich kann Situationen mit redundanten Informationen in mehreren Klassen durch Einführung einer Basisklasse vermeiden. 1.2.18	Ich kann überprüfen, ob in einer gegebenen Situation Vererbung angebracht ist, oder ob es besser wäre mit Komposition zu arbeiten. 1.2.18	Ich kenne den Unterschied zwischen technischen und fachlichen Klassen und kann in einem Design die vorhandenen Klassen diesen Kategorien zuordnen.	Ich kann eine switch-artige Struktur durch Anwendung von polymorphen Variablen (Delegation und Methoden überschreiben) eliminieren und fördere so die Erweiterbarkeit meines Codes. (22.3.18)	Ich kann ein einfaches Design-Patterns, wie Composite, Iterator, Observer, Template Methode etc. in einem eigenen Programm anwenden. (22.3.18)
Die Notation dynamischer und statischer Strukturen einer Anwendung mittels Unified Modeling Language (UML) nachvollziehen. **HZ2**	Ich kann die statische Struktur eines objektorientierten Programms mit einigen Klassen in einem UML-Klassendiagramm dokumentieren.	Ich kann die Objektsituation eines objektorientierten Programms zu einem bestimmten Zeitpunkt mit einem Speicherdiagramm erklären. 15.2.18	Ich kann in einem UML-Sequenzdiagramm den Aufruf von überschriebenen Methoden bei Objekten mit unterschiedlicher Klasse, aber identischer Basisklasse, illustrieren. (Illustration des dynamischen Bindens bei polymorphen Variablen)	Ich kenne die Bedeutung von weiteren UML-Elementen, wie Interfaces und Pakete und kann damit meine Klassendiagramme noch aussagekräftiger machen.	Ich kann mit einem UML-Tool ein objektorientiertes Modell für ein Programm erstellen und daraus Code erzeugen. Den Code kann ich von Hand zu einem lauffähigen Programm vervollständigen und schliesslich das Modell mit dem fertigen Code synchronisieren.
Objektorientiertes Design implementieren. **HZ3**	Ich kann mit Java eine Basisklasse und davon abgeleitete Klassen mit Instanzvariablen, und Konstruktoren implementieren. Ich vermeide dabei Redundanz im Code. 15.2.18	Ich kann Methoden in Basisklassen deklarieren / implementieren und in abgeleiteten Klassen implementieren / überschreiben. Ich verwende dann die Basisklasse zur Deklaration von polymorphen Variablen.1.3.18	Ich kann ein Listener-Interface implementieren oder eine Adapterklasse erweitern und damit Ereignisse in Java-Programmen behandeln.	Ich kann ein lauffähiges objektorientiertes Java-Programm mit mehreren Klassen und einer minimalen grafischen Benutzeroberfläche implementieren.	Ich mache ausgiebigen Gebrauch von weiteren Java-Sprachmitteln und den in der Java-Bibliothek verfügbaren Klassen. (22.3.18)
Fortgeschrittene Testfälle für funktionale Einheiten implementieren, welche durch geeignete Techniken von anderen Systemteilen unabhängig sind. **HZ4**	Ich kann für eine gegebene Klasse ein Interface definieren und dieses in einer anderen Klasse an Stelle der ersten Klasse einsetzen. Auf diese Weise kann ich die beiden Klassen entkoppeln.	Ich kann für ein gegebenes Interface eine Stub-Klasse implementieren, welche beim Aufruf der vom Interface definierten Methoden fest programmierte Resultate liefert.	Ich kann eine Stub-Klasse in einem Unit-Test einsetzen und so die zu testende Klasse unabhängig von anderen Klassen testen.	Ich kann fortgeschrittene Stub-Klassen implementieren, welche einen Zustand haben. Dieser wird durch die aufgerufenen Methoden verändert und kann im Unit-Test überprüft werden.	Ich kann in meinen Unit-Tests eine Mocking-Bibliothek, wie EasyMock oder Mockito, einsetzen und bei damit erzeugten Mock-Objekten die Einhaltung von Testbedingungen überprüfen.

Massstab: Alle Kompetenzen links der roten Linie erfüllt: Note 4.0. Pro fehlender / zusätzlicher Kompetenz - / + 0.25