**Konzept für den Kompetenznachweis des Moduls 226B**

Die Nachhaltigkeit der im Vorgängermodul verlangten Kompetenzen wird laufend in Gesprächen überprüft.

Die Kompetenzentwicklung wird im Modul 226B mit einzelnen Aufgabenstellungen nachgewiesen. Als Grundlage der Bewertung dient ein Kompetenzraster, welches einer Matrix von 4 Zeilen und 3 Spalten entspricht. Es findet eine **Zwischenprüfung** zu dem angeeigneten Wissen statt.

Jeder Aufgabenstellung ist ein Kompetenzprofil zugeordnet. Die Lehrperson übernimmt dabei die Rolle des Kunden und stellt den Bezug zu den Handlungszielen sicher.

**Nachweis einer Kompetenz:** Der einzelne Lernende hat die beschriebenen Kompetenzen im Verlaufe des Moduls nachzuweisen. Ist die entsprechende Kompetenz nachgewiesen worden, so wird dies durch das Visum des Coachs und des Lernenden im dazugehörigen Feld bestätigt.   
**Bewertet wird grundsätzlich nur die vom Lernenden erbrachte Eigenleistung. Fremde Quellen sind zu kennzeichnen und zu kommentieren.**

**Notengebung:** Prüfung 33% und Kompetenzraster( jedes Kompetenzfeld 0.25)

Folgende Formen sind für den Nachweis einer Kompetenz denkbar:

* **Fachgespräch**In einem Gespräch zeigt der Lernende, dass er über die Kompetenz verfügt.
* **Konkretes Produkt**Der Lernende zeigt die Kompetenz in Form eines entsprechenden Produkts (Programm, Dokumentation, erstelltes Diagramm, usw.)
* **Bericht Lernprozess**Der Lernende hält seine Erkenntnisse in einem selbst erstellten Bericht fest, in Form eines Blogs.

Der persönliche Lernfortschritt und die eingebrachten Nachweise sind:

* in einem **ePortfolio** zu *sammeln*, zu *strukturieren* und *zugänglich* zu machen
* im **Kompetenzjournal** festzuhalten, in Absprache mit der Lehrperson
* vorausschauend zu **planen** indem im Blog die nächsten Schritte geplant werden

**Kompetenzraster für das Modul 226B Name:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

|  |  |  |  |
| --- | --- | --- | --- |
| **Handlungsziel** | **A 3.0** | **B 4.0** | **C 5.0 bis 6.0\*** |
| **1 OO Design &  UML-Notation** | *Hz 1.1; 2.1* | *Hz 1.2; 2.2* | *Hz 1.3* |
| Der Lernende kann Anforderungen in ein Design umsetzen, welches auch Vererbung abbildet. Dabei kann er im Klassendiagramm die IST- und HAT-Beziehungen aufzeigen. Er kann das Design mittels UML-Werkzeugen umsetzen. | Die Anforderungen werden mit einem sinnvollen Design umgesetzt. Dabei wird beim Erstellen von Vererbungshierarchien das Prinzip der Spezialisierung / Generalisierung inklusive abstrakte Klassen gezeigt. Das gewählte Design wird mit geeigneten UML-Diagrammen dargestellt | Die eigene Anwendung hat ein ausgereiftes Design mit geeigneten UML-Diagrammen um eine Applikation umzusetzen (Klassendiagramm / Sequenzdiagramm). Das detaillierte Design ist Teil eines Projekts und wird in der Implementation widerspiegelt. |
| **2 Implementierung** | *Hz 3.1* | *Hz 3.2, 3.3* | *Hz 3.2* |
| Der Lernende kann das Prinzip der Vererbung anhand von Beispielen aufzeigen und dabei auch die Erweiterbarkeit der Applikation erklären und anwenden. Der Unterschied zwischen dem Überladen und Überschreiben von Methoden ist bekannt und kann gezeigt werden. Polymorphie kann an den Beispielen gezeigt werden. | Der Lernende zeigt anhand von erweiterten, eigenen Beispielen Kenntnisse über abstrakte Klassen im Zusammenhang mit Vererbung. Die Trennung von Spezifikation und Implementierung kann durch Interfaces realisiert werden. Die Beispiele zeigen auch die Konzepte Polymorphie. Basis kann bspw. Simulation eines sozialen Netzwerks, Auktionssystems, Flughafenplaner etc. sein. | Die eigene Anwendung zeigt ein vertieftes Verständnis der OO-Konzepte (Polymorphismus, Vererbung, Komposition) und beinhaltet auch Abstraktion oder Schnittstellen (Interfaces) – oder es wird aufgezeigt / erklärt, weshalb diese Konzepte nicht verwendet wurden. Delegation vs. Vererbung ist ersichtlich. |
| **3 Testen & Dokumentieren** | *Hz 4.1, 4.3* | *Hz 4* | *Hz 4.1;4.3* |
| Der Lernende kann Unit-Tests für seine Beispiele erstellen und ausführen. Er kann dabei auch aufzeigen, wie Testfälle angepasst werden können, damit die Erweiterbarkeit einer Applikation widerspiegelt wird. | Der Lernende setzt sinnvolle Unit-Tests zu den eigenen Beispielen um. Er kann auch zeigen, wieso Mockings sinnvoll sein können. Testprotokolle und ein Testfazit werden für die Beispiele aufgeführt. | Die eigene Anwendung umfasst Testfälle (pro Bereich) sowie automatisierte Tests (Unit-Suites). Es können SecurityTests (Grenzwerte, Pfade, Wertebereichte) umgesetzt werden. Testfälle sind in der Projekt-Dokumentation als Testprotokoll enthalten. |
| **4 Methoden- Sozialkompetenz & Selbstlernkompetenz** |  |  | *Hz 5.1* |
| Der Lernende zeigt, dass er das Gelernte auch anderen Lernenden mitteilen kann. Dies kann er mittels einer Präsentation oder durch Veröffentlichung von Tipps umsetzen (Anleitung, Code-Beispiele, etc.). Er kann eigene Verantwortung für das Lernen aufbringen und zeigt das in Gesprächen. | Der eigene Lernfortschritt wird selber geplant und auch geprüft. Das wird entsprechend dokumentiert. Ergebnisse werden kritisch hinterfragt. Der Austausch im Tandem / Gruppen ist ebenfalls nachvollziehbar (mittels Journal, Gesprächen, gemeinsames Repository, etc.). | Das Dokument zeigt den Entwicklungsprozess der eigenen Anwendung vom Design bis zur Implementation. (UML Diagramme) Gemachte Erfahrungen im Projekt werden in einem Bericht zusammengefasst, als Teil der Projektdokumentation. Im Fazit sind Vorteile (oder auch Nachteile) der Zusammenarbeit erkennbar. (Termine, produktiver Code, etc.). |

Hz = entspricht den Handlungszielen in der Modulidentifikation.

\*Grad der Benotung ist abhängig vom Grad des Zustands der Applikation (abgeschlossen), dessen Komplexität und Dokumentation. Siehe sep. Checkliste.