Kompetenzraster

Kompetenz

Die Lernenden kennen die Grundlagen des Programmierens, deren Herkunft, ihrer Voraussetzungen und sind in der Lage im Beruflichen Umfeld Probleme zu verstehen und Lösungen dafür zu entwickeln.

Handlungsziele und typische Handlungssituationen

1. Erfasst Problemstellungen, entwickelt strukturiert Lösungsansätze und übersetzt sie für die Stakeholder.

Paul Muster erhält von seinem Software-Team den Auftrag, eine Applikation zu programmieren. Stakeholder ist das Business-Team, welches das Problem beschrieben hat. Er kann mittels Rückfragen ans Business-Team das Problem eingrenzen und beschreiben. Er achtet darauf, dass er die Anforderungen genau und verständlich beschreibt, damit sie von den Business-Analysten verstanden werden. Er achtet auch darauf, dass er keine technischen Lösungen in den Anforderungen erwähnt. (HZ 1.1, 1.2)

2. Erstellt eine geeignete visuelle Darstellung für die Programmierung von Anforderungen.

Aus den Anforderungen kann Paul Muster die ungefähre Logik des Programms ableiten. Er kann den Ablauf visuell darstellen mit einem Activity Diagramm, damit er seinen Vorschlag auch den Business-Analysten zeigen kann. (HZ 3.1, 3.2)

3. Leitet aufgrund der Vorgaben die erforderlichen Daten (Eingabe, Verarbeitung, Ausgabe und ihre Datentypen) ab.

Paul Muster kann aus den beschriebenen Anforderungen die nötigen Eingaben vom Benutzer ableiten, sowie was die Ausgabe des Programms sein soll. Er kann daraus auch die notwendigen Variablen und ihre Datentypen bestimmen.(HZ 2.2) Mithilfe der Logikbeschreibung (Activity Diagramm) kann er weitere nötige Daten für die Verarbeitung und Ausgabe bestimmen.(HZ 2.3)

4. Implementiert die Applikation mit Hilfe von Kontrollstrukturen und selbst erstellten Funktionen.

Paul Muster kann in seinem Code die notwendingen Kontrollstrukturen (if-statemtents, loops) für die Logik einsetzen. (HZ 4.2, 4.4) Er achtet darauf, dass redundante Code-Abschnitte in Funktionen ausgegliedert werden. (HZ 4.6) Er gliedert seinen Code in wiederverwendbare Abschnitte (Funktionen). Er findet sich in der IDE zurecht und kann Fehler und Warnmeldungen korrekt

interpretieren - z.Bsp.wenn er Kontrollstrukturen an einer falschen Stelle im Code einsetzt oder schreibt. (HZ 4.5)

5. Hält vorgegebene Konventionen ein, kommentiert den Code und achtet dabei auf die Wartbarkeit.

Das Software-Team erklärt Paul Muster die Code-Conventions im Team. So muss z.Bsp. stets der Autor und eine kurze Beschreibung in jedem Programm stehen. Funktionen werden vorgängig kommentiert. (HZ 5.1) Paul Muster schreibt klare Kommentare, damit sein Code auch später verstanden wird. (HZ 5.2)

6. Interpretiert Mängel (Fehler) in der Software und korrigiert diese.

Paul Muster kann Fehler beim Ausführen des Programms ausfindig machen, indem er einen Debugger einsetzt. (HZ 6.1) Paul Muster fragt in seinem Team nach einem Code-Review, damit er eine Meinung zu seinem Code erhält. Im Code-Review Gespräch wird klar, dass ein Code-Abschnitt sehr umständlich geschrieben ist. Paul kann nun den Code effizienter und klarer gestalten.(HZ 6.3)

Kompetenzmatrix

Kompetenzband:	HZ	Grundlagen	Fortgeschritten	Erweitert
Probleme erfassen und Lösungsansätze entwickeln.	1	A1G: Ich kenne die Eigenschaften von gut formulierten Anforderungen	A1F: Ich kann Anforderungen erfassen und erläutern.	A1E: Ich kann einfache Anforderungen formulieren.
		A2G: Ich kann eine strukturierte Vorgehensweise für das Entwickeln einer Lösung aufzeigen	A2F: Ich kann eine strukturierte Vorgehensweise für das Entwickeln einer Lösung anwenden.	AE2: Ich kann Vorgehensweise kritisch hinterfragen und Verbesserungsvorschläge nennen.
Daten, Datentypen und Variablen ableiten und einsetzen	2,4	B1G: Ich kann die Unterschiede von Datentypen erklären. (Bsp: Ganzzahlen, Gleitkommazahlen, Zeichen usw.)	B1F: Ich kann den richtigen Datentyp für eine Variable aufgrund der Aufgabenstellung wählen.	B1E: Ich kann komplexe Datentypen (zBsp String, Integer, Double...) aufgrund der Aufgabenstellung wählen.
			B2F: Ich kann den Unterschied zwischen primitiven und komplexen Datentypen erklären.	
		B3G: Ich kann den Zweck von Variablen in einem Programm erläutern.	B3F: Ich kann Variablen deklarieren, initialisieren und Zuweisungen vornehmen.	B3E: Ich kann Variablen als Konstanten deklarieren und initialisieren.

		B4G: Ich kann die Möglichkeit der Verwendung von Variablen abhängig von Ihrem Datentyp erläutern.	B4F: Ich kann die Variablen abhängig von Ihrem Datentyp anwenden. (ZBps für arithmetische Operation)	B4E: Ich kann Variablen mit einem bestimmten Datentyp auf einen anderen umwandlen.(zBps: toString)
Zusammengesetze Datentypen einsetzen	2	C1G: Ich kann den Zweck von zusammengesetzten Datentypen erläutern. (ZBsp eindimensionaler Array)	C1F: Ich kann zusammengesetzte Datentypen deklarieren, initialisieren und Zuweisungen vornehmen. (ZBsp eindimensionaler Array)	C1E: Ich kann Methoden der zusammengesetzten Datentypen anwenden.
Anforderungen visuell darstellen	3	D1G: Ich kenne die Einsatzgebiete der grafischen Beschreibung eines Ablaufes. (zBsp mit Activity-Diagramm, Sequenz-Diagramm..)	D1F: Ich kann einen vorgegebenen Programmablauf grafisch darstellen. (zBsp Activity-Diagramm, Sequenz-Diagramm..)	D1E: Ich kann beschriebenen Ablauf in einen Programmablauf überführen und grafisch darstellen.(zBsp Activity-Diagramm, Sequenz-Diagramm..)
				D2E: Ich kann Bedingungen mit mehr als zwei Teilbedingungen formulieren.
Entwicklungsumgebung einsetzen	4,6	E1G: Ich kann den Zweck einer Entwicklungsumgebung (IDE) erklären.	E1F: Ich kann die Entwicklungsumgebung effizient einsetzen	E1E: Ich kann die Shortcuts und Funktionen der Entwicklungsumgebung effizient einsetzen
		E2G: Ich kann erläutern für was das Compilieren dient.	E2F: Ich kann vom Compiler angezeigte Fehler- und Warnmeldung interpretieren.	E2E: Ich kann die Ursachen der vom Compiler angezeigten Fehler- und Warnmeldungen beheben.
		E3G: Ich kann erläutern für was ein Debugger dient.	E3F: Ich kann einen Debugger zur Programmausführung anwenden.	E3E: Ich kann einen Debugger für die Fehleranalyse einsetzen.
Applikation implementieren	3,4	F1G: Ich kann Aufbau, Syntax und Struktur eines einfachen Programmes erklären.	F1F: Ich kann einen detailliert vorgegeben Ablauf mit einer Programmiersprache umsetzen.	F1E: Ich kann einen grob beschriebenen Ablauf detaillieren und mit einer Programmiersprache umsetzen.

		F2G: Ich kann erklären wieso die Aufteilung eines Programmes in verschiedene Methoden sinnvoll ist.	F2F: Ich kenne den Aufbau und den Aufruf einer Methode (Deklaration und Implementation) und kann diese korrekt einsetzen. (zBsp Instanzvariablen, Parametern, lokalen Variablen, Return Werte)	F2E: Ich kann Instanzvariablen, Parametern, lokalen Variablen und Return Werte gezielt einsetzen.
		F3G: Ich kann Selektionen und Iterationen (Kopf und Fussgesteuert) mit Bedingungen codieren.	F3F: Ich kann zwei Teilbedingungen mit AND oder OR verknüpfen.	F3E: Ich kann verschachtelte Kontrollstrukturen einsetzen.
Konventionen einhalten	5	G1G: Ich kann mein Programm mit ein- und mehrzeiligen Kommentaren ergänzen.	G1F: Ich kenne Möglichkeiten Kommentare zu Formatieren oder zu Annotieren (zBsp Fix Me!, TODO etc.)	G1E: Ich setze Konventionen ein. (zBsp Clean Code, Coding Guidelines..)

Kompetenzstufen

Grundlagen | Stufe 1

Diese Stufe ist als Einstieg ins Thema gedacht. Der Fokus liegt hier auf dem Verstehen von Begriffen und Zusammenhängen.

Als Richtungshinweis: Wer alle Kompetenzen in dieser Stufe erfüllt, hat die Noten 3.0

Fortgeschritten | Stufe 2

Diese Stufe definiert den Pflichtstoff, den alle Lernenden am Ende des Moduls möglichst beherrschen sollen.

Als Richtungshinweis: Wer alle Kompetenzen in dieser Stufe erfüllt, hat die Noten 4.5

Erweitert | Stufe 3

Diese Lerninhalte für Lernende gedacht, die schneller vorankommen und einen zusätzlichen Lernanreiz erhalten sollen.

Als Richtungshinweis: Wer alle Kompetenzen in dieser Stufe erfüllt, hat die Noten 6