I Erläuterungen

Voraussetzungen gemäß KCGO und Abiturerlass in der für den Abiturjahrgang geltenden Fassung

Standardbezug
Die nachfolgend ausgewiesenen prozessbezogenen Kompetenzbereiche sind für die Bearbeitung der jeweiligen Aufgabe besonders bedeutsam. Darüber hinaus können weitere, hier nicht ausgewiesene prozessbezogene Kompetenzbereiche für die Bearbeitung der Aufgabe nachrangig bedeutsam sein, zumal die Kompetenzbereiche in engem Bezug zueinander stehen. Die Operationalisierung des Standardbezugs erfolgt in Abschnitt II.

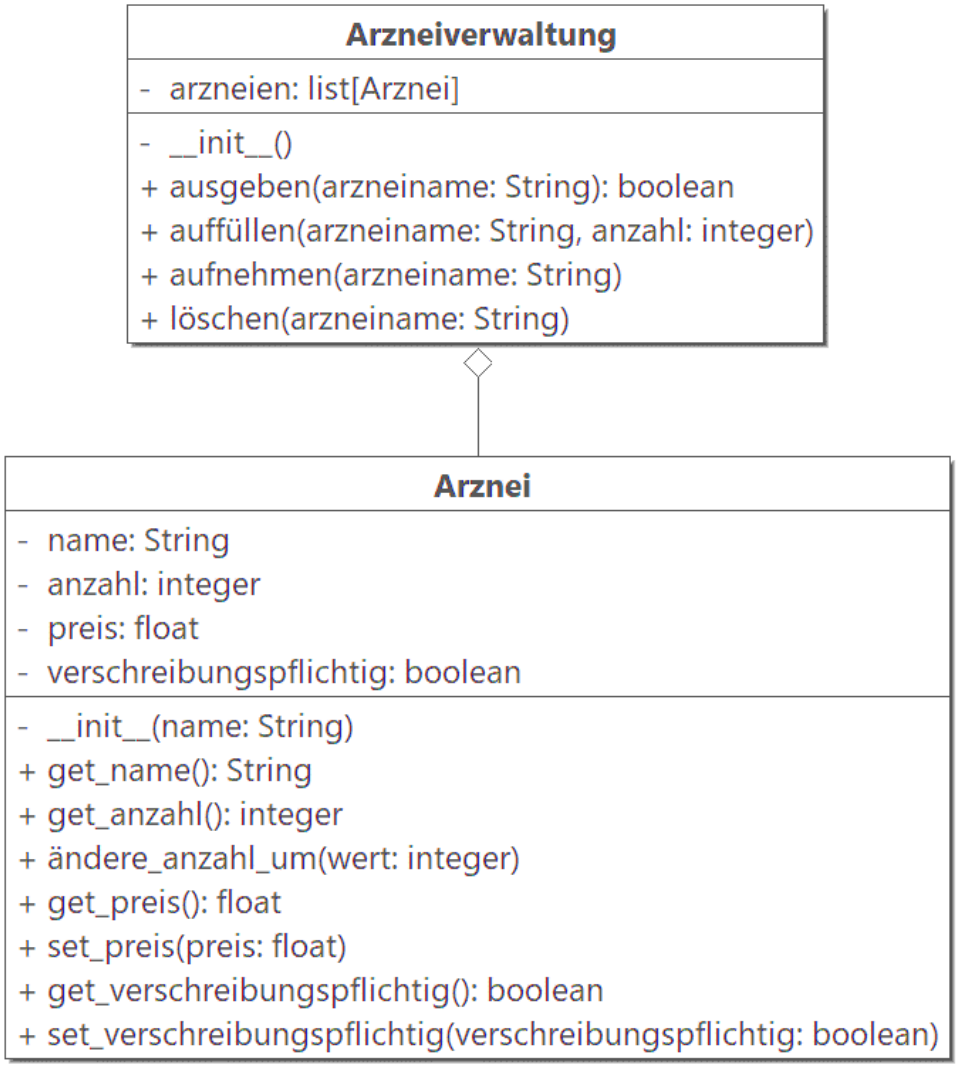
Aufgabe	Prozessbezogene Kompetenzbereiche				
	P1	P2	P3	P4	P5
1.1			X		
1.2					X
2.1		X			
2.2			X		
2.3			X		
3.1	X				
3.2			X		
3.3	X	X	X		
3.4					X
4				X	X

Inhaltlicher Bezug
Der vorliegende Vorschlag bezieht sich schwerpunktmäßig auf die inhaltsbezogenen Kompetenzbereiche Algorithmen (I1), Information und Daten (I3), Informatiksysteme (I4) nach KCGO.

Q1: Algorithmik und objektorientierte Modellierung
verbindliche Themenfelder: Such- und Sortieralgorithmen (Q1.1); Rekursion (Q1.2); Klassen und Objekte (Q1.3)

II Lösungshinweise und Bewertungsraster

In den nachfolgenden Lösungshinweisen sind alle wesentlichen Gesichtspunkte, die bei der Bearbeitung der einzelnen Aufgaben zu berücksichtigen sind, konkret genannt und diejenigen Lösungswege aufgezeigt, welche die Prüflinge erfahrungsgemäß einschlagen werden. Lösungswege, die von den vorgegebenen abweichen, aber als gleichwertig betrachtet werden können, sind ebenso zu akzeptieren.

Aufg.	erwartete Leistungen	BE
1.1		11
1.2	Es handelt sich um eine Aggregation, da die Objekte der Klasse *Arznei* Teile eines Objektes der Klasse *Arzneiverwaltung* sind. Die Klasse *Arzneiverwaltung* besitzt die Liste *arzneien*, deren Elemente vom Typ *Arznei* sind.	2
2.1	Die Methode ist öffentlich und besitzt einen Rückgabewert vom Typ *boolean*. Sie bekommt einen Arzneinamen als Parameter übergeben. In einer for-Schleife wird die gesamte Arzneimittelliste durchlaufen, beginnend mit dem ersten Element. Es wird bei jedem Schleifendurchlauf überprüft, ob der Name der an der aktuellen Stelle gespeicherten Arznei dem gesuchten Arzneinamen entspricht. Falls ja, wird überprüft, ob noch mindestens eine Packung der Arznei vorrätig ist. Ist dies der Fall, wird die Anzahl um eins verringert und die Methode mit einer Rückgabe von *True* beendet. Ist die Anzahl gleich 0, wird *False* zurück geliefert. Wurde die Schleife komplett durchlaufen, ohne den Arzneinamen im Feld gefunden zu haben, gibt die Methode ebenfalls *False* zurück.	8

2.2	`def löschen(self, arzneiname: str):`
	`  for arznei in self.__arzneien:`
	`  if arznei.get_name() == arzneiname:`
	`  self.__arzneien.remove(arznei)`
		4

Aufg.	erwartete Leistungen	BE

2.3	`def suche_knappe_arzneien(self, grenzwert: int) ->list[str]:`
	`  ergebnis: list[str] = []`
	`  for arznei in self.__arzneien:`
	`  if arznei.get_anzahl() < grenzwert:`
	`  ergebnis.append(arznei.get_name())`
	`  return ergebnis`
		7

3.1	Die gesamte Liste wird von links nach rechts durchlaufen. Der Name der aktuellen Arznei wird mit dem Namen der nächsten Arznei verglichen. Liegt der Name der aktuellen Arznei alphabetisch hinter dem Namen der nächsten Arznei, werden die Plätze der Arzneien getauscht. Der Namensvergleich erfolgt nun mit der nächsten Arznei in der Liste. Wurden so alle Arzneien betrachtet, ist die Arznei mit dem alphabetisch am weitesten hinten stehenden Namen nun das letzte Element in der Liste. Nun wird erneut vorne in der Liste begonnen die beschriebenen Vergleiche durchzuführen. Im zweiten Durchlauf findet die Arznei mit dem alphabetisch vorletzten Namen ihren Platz als vorletztes Element in der Liste. So wird weiter verfahren, bis zum Schluss alle Arzneien in der Liste alphabetisch sortiert nach den Arzneinamen vorliegen.	7

3.2	`def sortiere_arzneien(self):`
	`  for i in range(len(self.__arzneien) - 1):`
	`  min: int = i`
	`  for j in range(i + 1, len(self.__arzneien)):`
	`  if (self.__arzneien[j].get_name()`
	`  < self.__arzneien[min].get_name()):`
	`  min = j`
	`  (self.__arzneien[i], self.__arzneien[min]) \`
	`  = (self.__arzneien[min], self.__arzneien[i])`
		8

3.3	In der Arzneiliste wird zunächst die mittlere Position geprüft. Kommt der Name der dort gespeicherten Arznei alphabetisch nach dem gesuchten Arzneinamen, wird die binäre Suche erneut aufgerufen und in der linken Teilliste auf die gleiche Weise weitergesucht. Sind die Arzneinamen jedoch identisch, wird die Position zurückgegeben und die Suche erfolgreich beendet. Kommt die dort gespeicherte Arznei alphabetisch vor dem gesuchten Arzneinamen, wird die binäre Suche erneut aufgerufen und in der rechten Teilliste auf die gleiche Weise weitergesucht. Wenn beim Aufruf der binären Suche der Anfang und das Ende des betrachteten Bereichs identisch sind, muss geprüft werden, ob die Arznei den gesuchten Namen besitzt. Ist dies der Fall, wird die Position zurückgegeben. Stimmt der Name an dieser Stelle nicht überein, wird die Suche mit dem Rückgabewert -1 beendet, da dann die gesuchte Arznei nicht im Feld vorhanden ist.	6
3.4	Die lineare Suche durch alle Elemente hat eine lineare Laufzeit, die binäre Suche dagegen ist mit einer logarithmischen Laufzeit erheblich schneller.	2
4	Der Klasse *Arznei* kann eine Liste von Packungen hinzugefügt werden. Die Klasse *Packung* hat dabei die beiden Attribute *Packungsnummer* und *Haltbarkeitsdatum* sowie get/set-Methoden für den Zugriff auf die Attribute. Zwischen den Klassen *Arznei* und *Packung* besteht dann eine Aggregationsbeziehung.	5
	Summe	**60**

III Bewertung und Beurteilung

Die Bewertung und Beurteilung erfolgt unter Beachtung der nachfolgenden Vorgaben nach § 33 der Oberstufen- und Abiturverordnung (OAVO) in der jeweils geltenden Fassung. Bei der Bewertung und Beurteilung der sprachlichen Richtigkeit in der deutschen Sprache sind die Bestimmungen des § 9 Abs. 12 Satz 3 OAVO in Verbindung mit Anlage 9b anzuwenden.

Der Fehlerindex ist nach Anlage 9b zu § 9 Abs. 12 OAVO zu berechnen. Für die Ermittlung der Punkte nach Anlage 9a zu § 9 Abs. 12 OAVO bzw. des Abzugs nach Anlage 9b zu § 9 Abs. 12 OAVO wird jeweils der ganzzahlige nicht gerundete Prozentsatz bzw. Fehlerindex zugrunde gelegt. Der prozentuale sprachliche Anteil nach Anlage 9b zu § 9 Abs. 12 OAVO wird auf 20 % festgesetzt.

Darüber hinaus sind die Vorgaben der Erlasse „Hinweise zur Vorbereitung auf die schriftlichen Abiturprüfungen (Abiturerlass)“ und „Durchführungsbestimmungen zum Landesabitur“ in der für den Abiturjahrgang geltenden Fassung zu beachten.

<Im Fach Informatik (Leistungskurs) werden Vorschläge zu den Themen der drei Kurshalbjahre Q1, Q2 und Q3 vorgelegt, wobei die Prüfungsleistung aus der Bearbeitung je eines Vorschlags zu jedem Halbjahresthema besteht, wofür insgesamt maximal 100 BE vergeben werden können. Ein Prüfungsergebnis von **5 Punkten (ausreichend)** setzt voraus, dass mindestens 45 % der zu vergebenden BE erreicht werden. Ein Prüfungsergebnis von **11 Punkten (gut)** setzt voraus, dass mindestens 75 % der zu vergebenden BE erreicht werden.>

Im Fach Informatik (Grundkurs) können Vorschläge zu den Themen der drei Kurshalbjahre Q1, Q2 und Q3 vorgelegt werden. Die Prüfungsleistung besteht aus der Bearbeitung von zwei Vorschlägen, einem zum Thema „objektorientierte Modellierung“ und einem weiteren zu einem der beiden anderen Halbjahresthemen, wofür insgesamt maximal 100 BE vergeben werden können. Ein Prüfungsergebnis von **5 Punkten (ausreichend)** setzt voraus, dass mindestens 45 % der zu vergebenden BE erreicht werden. Ein Prüfungsergebnis von **11 Punkten (gut)** setzt voraus, dass mindestens 75 % der zu vergebenden BE erreicht werden.

Gewichtung der Aufgaben und Zuordnung der Bewertungseinheiten zu den Anforderungsbereichen

Aufgabe	Bewertungseinheiten in den Anforderungsbereichen			Summe
	AFB I	AFB II	AFB III	
1	6	7		**13**
2	3	11	5	**19**
3	9	12	2	**23**
4			5	**5**
Summe	**18**	**30**	**12**	**60**

Die auf die Anforderungsbereiche verteilten Bewertungseinheiten innerhalb der Aufgaben sind als Richtwerte zu verstehen.