Hinweise für den Prüfling

Auswahlzeit: 30 Minuten

Bearbeitungszeit (insgesamt): 180 Minuten

Auswahlverfahren

Es gibt zwei Aufgabengruppen A und B, aus denen jeweils ein Vorschlag zu bearbeiten ist. Der vorliegende Vorschlag aus der Gruppe A (objektorientierte Modellierung) ist ein Pflichtvorschlag.

Wählen Sie von den zwei vorliegenden Vorschlägen der Gruppe B (Datenbanken) einen zur Bearbeitung aus. Der nicht ausgewählte Vorschlag muss am Ende der Auswahlzeit der Aufsicht führenden Lehrkraft zurückgegeben werden.

Erlaubte Hilfsmittel

1. ein Wörterbuch zur deutschen Rechtschreibung
2. eine Liste der fachspezifischen Operatoren

Sonstige Hinweise

ohne PC-Nutzung

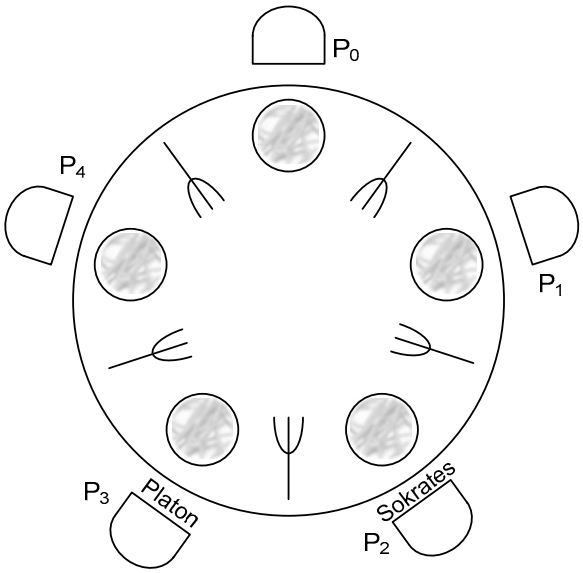
In jedem Fall vom Prüfling auszufüllen

Name: _____ Vorname: _____

Prüferin / Prüfer: _____ Datum: _____

Das Philosophenproblem

Das Philosophenproblem stellt anschaulich eine Situation dar, die eintritt, wenn mehrere Computerprozesse auf gemeinsame Ressourcen zugreifen.



Fünf Philosophen sitzen an den Plätzen P_0, …, P_4 um einen runden Tisch. Vor jedem Philosophen steht ein Teller Spaghetti, und zwischen den Tellern liegt je eine Gabel. Zum Essen benötigt jeder Philosoph beide Gabeln, die zur Rechten und die zur Linken seines Tellers. Der Ausgangszustand der Philosophen ist denkend. Nachdem ein Philosoph eine Weile nachgedacht hat, wird er hungrig und beendet das Denken. Er versucht zuerst die Gabel links zu nehmen, dann die Gabel rechts und anschließend zu essen. Hat er gegessen, legt er beide Gabeln zurück und geht wieder über zum Denken. Kann eine Gabel nicht aufgenommen werden, weil sie gerade nicht auf dem Tisch liegt, wartet der Philosoph.

Beispiel einer Aktionsfolge, wobei jeder Satz durch den Aufruf einer Methode generiert wird:

> Sokrates wird hungrig.
> Sokrates versucht seine linke Gabel zu nehmen: erfolgreich.
> Sokrates versucht seine rechte Gabel zu nehmen: erfolgreich.
> Sokrates isst.
> Platon wird hungrig.
> Platon versucht seine linke Gabel zu nehmen: erfolgreich.
> Platon versucht seine rechte Gabel zu nehmen: wartet.
> Sokrates beendet das Essen, legt die Gabeln zurück und denkt.
> Platon versucht seine rechte Gabel zu nehmen: erfolgreich.
> Platon isst.

Aufgaben

1.1 Modellieren Sie die Klasse *Gabel*, deren Beziehung zur bereits gegebenen Klasse *Philosoph* (Material 1) und die fehlenden Methoden der Klasse *Philosoph* in Form eines UML-Klassendiagramms.
Geben Sie dabei alle für die Simulation des Problems nötigen Attribute und Methoden unter Berücksichtigung der Datenkapselung, der Aufgaben 2 bis 4 und des Materials 2 an.

(12 BE)

1.2 Erläutern Sie die Beziehung, die zwischen den Klassen *Philosoph* und *Gabel* besteht.

(5 BE)

2. Implementieren Sie die Methode *nimmLinkeGabel()* der Klasse *Philosoph*. Dabei sollen alle betroffenen Attribute von *Philosoph* und *Gabel* verändert werden und eine Textausgabe entsprechend dem obigen Beispiel erfolgen.

(7 BE)

3. Implementieren Sie für das obige Beispiel einer Aktionsfolge eine passende Sequenz von Methodenaufrufen, durch welche zunächst die Philosophen Sokrates und Platon, die drei ihnen zugeordneten Gabeln und anschließend die Textausgabe in der Konsole erzeugt werden.

(7 BE)

4. Nehmen alle Philosophen direkt nacheinander ihre linke Gabel, tritt in der Folge ein Problem auf.

4.1 Erläutern Sie das auftretende Problem.

(5 BE)

4.2 Obiges Problem soll durch die Klasse *Verwaltung* gelöst werden (Material 2). Die Klasse sieht eine variable Anzahl von Plätzen am Tisch vor.
Analysieren Sie die Methode *geprüftLinkeGabelNehmen(String Name)* der Klasse *Verwaltung*. Die Methode wird erst aufgerufen, wenn der Tisch voll besetzt ist.

(12 BE)

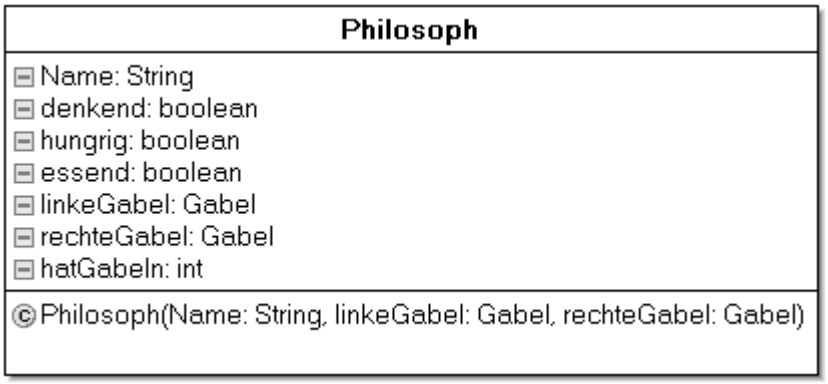
4.3 Erläutern Sie eine alternative Lösung des Problems, für die keine weitere Verwaltungsklasse benötigt wird.

(5 BE)

5. Implementieren Sie die Methode *zeigeFeld()* der Klasse *Verwaltung*, die für jede Stelle des Feldes *Philosophen* den Namen des dort gespeicherten Philosophen ausgibt. Sind Plätze am Tisch unbesetzt, soll für diese Plätze eine entsprechende Meldung erfolgen.

(7 BE)

Material 1



Material 2

```
01 public class Verwaltung {
02   private int AnzahlPlätze;
03   private int AnzahlPhilosophen;
04   private Philosoph[] Philosophen;
05
06   public Verwaltung(int Anzahl) {
07     AnzahlPhilosophen = 0;
08     AnzahlPlätze = Anzahl;
09     Philosophen = new Philosoph[Anzahl];
10   }
11
12   public void neuerPhilosoph(Philosoph einPhilosoph) {
13     if (AnzahlPhilosophen == AnzahlPlätze) {
14       System.out.println("Alle Plätze am Tisch sind bereits besetzt.");
15     } else {
16       Philosophen[AnzahlPhilosophen] = einPhilosoph;
17       AnzahlPhilosophen++;
18     }
19   }
20
21   public void zeigeFeld() {
22     // zu implementieren in Aufgabe 5
23   }
24
25   public void geprüftLinkeGabelNehmen(String Name) {
26     int i = 0;
27     while ((i < AnzahlPlätze) && !Philosophen[i].getName().equals(Name)) {
28       i++;
29     }
30     if (i == AnzahlPlätze) {
31       System.out.println("Es gibt am Tisch keinen Philosophen " +
32                          "mit dem gewünschten Namen.");
33     } else {
34       int linkerNachbar = i + 1;
35       if (linkerNachbar == AnzahlPlätze) {
36         linkerNachbar = 0;
37       }
38       if (Philosophen[linkerNachbar].getLinkeGabel().getAufDemTisch()) {
39         Philosophen[i].nimmLinkeGabel();
40       } else {
41         System.out.println("Die Gabel wird gleich von Ihrem " +
42                            "linken Nachbarn benötigt, bitte warten.");
43       }
44     }
45   }
46 }
```