Hinweise für den Prüfling

Auswahlzeit: 30 Minuten

Bearbeitungszeit (insgesamt): 180 Minuten

Auswahlverfahren

Es gibt zwei Aufgabengruppen A und C, 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 C (Konzepte und Anwendungen der theoretischen Informatik) 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
Java-Variante

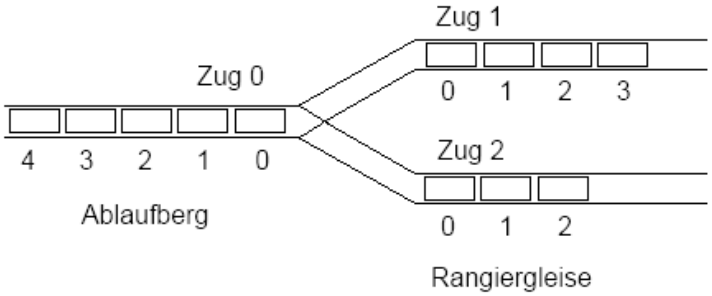
In jedem Fall vom Prüfling auszufüllen

Name: _____ Vorname: _____

Prüferin / Prüfer: _____ Datum: _____

Rangierbahnhof

Auf einem Rangierbahnhof werden Güterzüge in Waggons zerlegt und diese zu neuen Zügen zusammengestellt. Zum Rangieren verwendet man einen Ablaufberg. Die Rangierlokomotive kann einen auf dem Rangiergleis stehenden Zug auf den Ablaufberg ziehen. Auf dem Ablaufberg werden die Waggons der Reihe nach entkuppelt. Sie rollen dann eigenständig das Gefälle hinab und gelangen gemäß der jeweiligen Weichenstellung auf die vor Beginn des Rangiervorgangs leeren Rangiergleise.



Aufgaben

1. Für die Modellierung des Rangiervorgangs besteht ein Güterzug aus keinem, einem oder mehreren Waggons und hat keine Lokomotive. Wir beschränken uns auf Züge mit maximal zwölf Waggons. Von jedem Waggon sind seine Start- und Zielbahnhofsnummer zu speichern.

1.1 Modellieren Sie einen Güterzug durch ein UML-Klassendiagramm mit den beiden Klassen *Zug* und *Waggon*. Sehen Sie in der Klasse *Waggon* Attribute und Methoden so vor, dass die im Text genannten Informationen modelliert und später abgefragt werden können. Bei der Modellierung der Klasse *Zug* sind auch Material 1 und aus Material 2 die in der Methode *WaggonsAbstossen()* verwendeten Methoden sowie die sich daraus ergebenden Attribute zu berücksichtigen. Begründen Sie die von Ihnen ausgewählte Klassenbeziehung.

(12 BE)

1.2 Beschreiben Sie die beiden Konstruktoren der Klasse *Zug* in Material 1.

(6 BE)

2.1 Das An- und Abkuppeln von Waggons kann beim Rangieren nur am Zugende stattfinden. Es wird davon ausgegangen, dass der letzte Waggon an der Position 0 im Feld gespeichert ist. Implementieren Sie die Methode *ankuppeln(Waggon einWaggon)* der Klasse *Zug*, um einen Waggon an einen Zug anzukuppeln.

(8 BE)

2.2 Die Methode *sucheMaxZiel()* bestimmt die größte Zielbahnhofsnummer der Waggons eines Zuges. Implementieren Sie *sucheMaxZiel()*.

(8 BE)

3. Die Klasse *Rangierbahnhof* in Material 2 besitzt als Attribute die drei Züge, die in obiger Grafik dargestellt sind. *Zug0* ist dabei der zu rangierende Güterzug, dessen Waggons mit Hilfe der zwei anfangs leeren Züge *Zug1* und *Zug2* umrangiert werden sollen.

3.1 Analysieren Sie die beiden in Material 2 dargestellten Methoden *WaggonsAbstossen()* und *ZugAufDenAblaufberg(Zug einZug)*.

(12 BE)

3.2 Analysieren und erklären Sie unter Einbezug von Material 2 den Algorithmus *Rangieren*, der in Material 3 als Struktogramm dargestellt ist.

(10 BE)

4. Erläutern Sie unter Berücksichtigung aller Aufgabenteile, warum die Verwendung zweier Konstruktoren innerhalb der Klasse *Zug* sinnvoll ist.

(4 BE)

Material 1

```
1   public class Zug {
2     private int Anzahl;
3     private int MaxZahl = 12;
4     private Waggon[] Waggons = new Waggon[MaxZahl];
5
6     public Zug() {
7       for (int i = 0; i < MaxZahl; i++)
8         Waggons[i] = null;
9       Anzahl = 0;
10    }
11
12    public Zug(int Start)  { // ein Testzug
13      Waggons[ 0] = new Waggon(Start, 4);
14      Waggons[ 1] = new Waggon(Start, 6);
15      Waggons[ 2] = new Waggon(Start, 7);
16      Waggons[ 3] = new Waggon(Start, 5);
17      Waggons[ 4] = new Waggon(Start, 1);
18      Waggons[ 5] = new Waggon(Start, 8);
19      Waggons[ 6] = new Waggon(Start, 6);
20      Waggons[ 7] = new Waggon(Start, 2);
21      Waggons[ 8] = new Waggon(Start, 9);
22      Waggons[ 9] = new Waggon(Start, 1);
23      Waggons[10] = new Waggon(Start, 7);
24      Waggons[11] = new Waggon(Start, 6);
25      Anzahl = MaxZahl;
26    }
27
28    public int sucheMaxZiel() {
29      // zu implementieren in Aufgabe 2.2
30    }
31
32    ...
33  }
```

Material 2

```
1   public class Rangierbahnhof {
2     private Zug Zug0;
3     private Zug Zug1;
4     private Zug Zug2;
5
6     public void WaggonsAbstossen() {
7       Waggon einWaggon;
8       int MaxZiel = Zug0.sucheMaxZiel();
9       int AnzahlWaggons = Zug0.getAnzahl();
10      for (int i = 0; i < AnzahlWaggons; i++) {
11        einWaggon = Zug0.abkuppeln();
12        if (einWaggon.getZiel() < MaxZiel)
13          Zug2.ankuppeln(einWaggon);
14        else
15          Zug1.ankuppeln(einWaggon);
16      }
17    }
18
19    public void ZugAufDenAblaufberg(Zug einZug) {
20      int AnzahlWaggons = einZug.getAnzahl();
21      for (int i = 0; i < AnzahlWaggons; i++)
22        Zug0.ankuppeln(einZug.abkuppeln());
23    }
24
25    ...
26  }
```

Material 3

Algorithmus Rangieren