I. Erläuterungen

II. Lösungshinweise

Entsprechend den Vorgaben der VOGO/BG, Anlage 11 I. Abs. 2.3.1 werden in den nachfolgenden Lösungshinweisen 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. Selbstverständlich sind jedoch Lösungswege, die von den vorgegebenen abweichen, aber als gleichwertig betrachtet werden können, ebenso zu akzeptieren.

1	Es wird die Straßenbahn Nr. 273 betriebsbereit gemacht, im Modell also mittels Konstruktor erzeugt. Gleiches folgt für die Straßenbahn Nr. 417. Straßenbahnen können halten oder fahren. Bei haltender Straßenbahn können Passagiere ein- und aussteigen, während der Fahrt ist das nicht möglich. Je nachdem, wie viele Personen ein- oder aussteigen, ändert sich die Anzahl der Passagiere. Es können nicht mehr Passagiere aussteigen, als in einer Straßenbahn sind. Aus der Beschreibung ergibt sich das UML-Klassendiagramm.

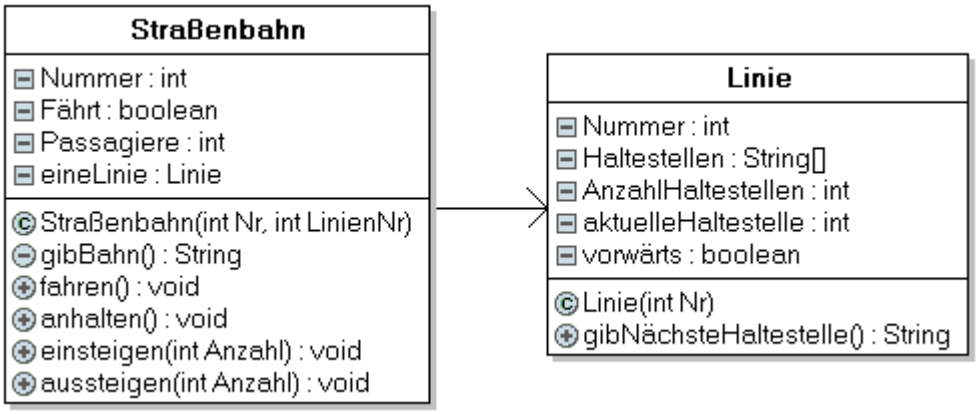
```
public class Straßenbahn {
  private int Nummer;
  private boolean Fährt;
  private int Passagiere;

  public Straßenbahn(int Nr) {
    Nummer = Nr;
    Fährt = false;
    Passagiere = 0;
    System.out.println(gibBahn() + "Betriebsbereit.");
  }

  private String gibBahn() {
    return "Straßenbahn " + Nummer + ": ";
  }

  public void fahren() {
    Fährt = true;
    System.out.println(gibBahn() + "Fährt.");
  }

  public void anhalten() {
    Fährt = false;
    System.out.println(gibBahn() + "Hält.");
  }

  public void einsteigen(int Anzahl) {
    if (Fährt) {
      System.out.println(gibBahn() +
             "Einsteigen nicht möglich. Straßenbahn fährt.");
    } else {
      Passagiere = Passagiere + Anzahl;
      System.out.println(gibBahn() +
             "Es sind " + Anzahl + " Passagiere eingestiegen.");
    }
  }

  public void aussteigen(int Anzahl) {
    if (Fährt) {
      System.out.println(gibBahn() +
             "Aussteigen nicht möglich. Straßenbahn fährt.");
    } else if (Anzahl > Passagiere) {
```

```
      System.out.println(gibBahn() +
             "So viele Passagiere sind nicht in der Bahn.");
    } else {
      Passagiere = Passagiere - Anzahl;
      System.out.println(gibBahn() +
             "Es sind " + Anzahl + " Passagiere ausgestiegen.");
    }
  }
}
```

2

```
public static void main(String[] args) {
  Straßenbahn Bahn1 = new Straßenbahn(273);
  Straßenbahn Bahn2 = new Straßenbahn(417);
  Bahn1.fahren();
  Bahn2.einsteigen(5);
  Bahn2.fahren();
  Bahn2.anhalten();
  Bahn2.aussteigen(4);
  Bahn1.einsteigen(3);
  Bahn1.anhalten();
  Bahn1.aussteigen(1);
}
```

3



Es handelt sich bei der Beziehung um eine gerichtete Assoziation zwischen den beiden Klassen *Straßenbahn* und *Linie*. Ein Straßenbahn-Objekt hat demnach Zugriff auf das Linienobjekt und kann sich mit der Methode *gibNächsteHaltestelle* die nächste Haltestelle für die Ansage holen. Zur Implementierung der Assoziation erhält die Klasse *Straßenbahn* ein Attribut *eineLinie* vom Typ Linie. Dieses Attribut kann beispielsweise im Konstruktor mit der Linie belegt werden, auf der die Straßenbahn fährt.

4

```
public String gibNächsteHaltestelle() {
  if (vorwärts == true) {
    aktuelleHaltestelle = aktuelleHaltestelle + 1;
    if (aktuelleHaltestelle == AnzahlHaltestellen - 1)
      vorwärts = false;
  } else {
    aktuelleHaltestelle = aktuelleHaltestelle - 1;
    if (aktuelleHaltestelle == 0)
      vorwärts = true;
  }
  return Haltestellen[aktuelleHaltestelle];
}

System.out.println(gibBahn() +
   "Nächste Haltestelle: " + eineLinie.gibNächsteHaltestelle());
```

5

Das Problem kann beispielsweise so gelöst werden, dass sich das Straßenbahnobjekt seine nächste Haltestelle bestimmt und dann bei allen anderen Linien nachprüft, ob diese dort auch eine Haltstelle haben. Dazu muss in der Klasse *Linie* eine boolesche Funktion *hatHaltestelle*

	(String Haltestelle) realisiert werden. Hat eine andere Straßenbahn dieselbe Haltestelle, so besteht dort auch eine Umsteigemöglichkeit. Außerdem braucht die Straßenbahn Zugriff auf alle Linien, nicht nur auf die eigene. Dazu kann man eine Klasse *Liniennetz* modellieren, die alle Linien verwaltet. Der Klasse *Straßenbahn* gibt man per Assoziation Zugriff auf das Liniennetz. Somit kann ein Straßenbahnobjekt bei allen Linien prüfen, ob sie dieselbe Haltestelle haben und somit eine Umsteigemöglichkeit existiert.

III. Bewertung und Beurteilung

Die Bewertung und Beurteilung erfolgt gemäß den Bestimmungen in den Anlagen 11 sowie ggf. 9a bis 9e der VOGO/BG in der jeweils gültigen Fassung. Für die Umrechnung von Prozentanteilen der erbrachten Leistungen in Notenpunkte nach §13 Abs. 1 der VOGO/BG gelten die Werte in der Anlage 8 der VOGO/BG in der jeweils gültigen Fassung. Darüber hinaus sind die Vorgaben des Einführungserlasses für das Landesabitur 2007 in der Fassung vom 13. Oktober 2005 zu beachten.

Im Fach Informatik (Grundkurs) werden Vorschläge aus den Kategorien A (Modellierung), B (Datenbanken) und C (theoretische Informatik) vorgelegt, wobei die Prüfungsleistung aus der Bearbeitung von zwei Vorschlägen, einem aus der Kategorie A und einem weiteren aus einer der beiden anderen Kategorien besteht. Es können hierfür insgesamt maximal 100 BE vergeben werden. Ein Prüfungsergebnis von **5 Punkten** (ausreichend) setzt voraus, dass insgesamt 46 BE, ein Prüfungsergebnis von **11 Punkten** (gut), dass insgesamt 76 BE erreicht werden.

Gewichtung der Teilaufgaben und Zuordnung der Bewertungseinheiten zu den Anforderungsbereichen

Aufgabe	Bewertungseinheiten in den Anforderungsbereichen			Summe
	AFB I	AFB II	AFB III	
1	12	10		**22**
2	8			**8**
3		10		**10**
4		10	3	**13**
5			7	**7**
Summe	**20**	**30**	**10**	**60**