Aufgabe 1 (15 Punkte)

Für eine kleine Schreinerei soll ein Verzeichnis aller noch ausstehenden externen Montageaufträge angelegt werden. Das Verzeichnis soll nach dem Datum der Montage aufsteigend sortiert werden können. Alle Montagen beginnen um 8 Uhr und es findet höchstens eine Montage pro Tag statt. Die Schreinerei hat nur 1 Lieferwagen mit Platz für maximal 4 Monteure.

`Getter`- und `setter`-Methoden müssen Sie nicht angeben und können diese als vorhanden voraussetzen.

a) Geben Sie eine Klasse `Montage` an, die die privaten Attribute `auftraggeber`, `datum` und `anzFreiePlaetze`, enthält. Ein Zähler `anzMontagen` soll geeignet über die Anzahl aller instanzierten Montagen Buch führen.

Hinweis: Achten Sie darauf geeignete Datentypen für die Attribute zu wählen!

b) Die Klasse soll über zwei Konstruktoren zur problemspezifischen Initialisierung verfügen: den Standard-Konstruktor und einen Konstruktor, der alle drei oben genannten Attribute als Parameter besitzt. Geben Sie die beiden Konstruktoren an.

c) Die öffentliche Methode `monteureBuchen` der Klasse `Montage` hat einen Parameter, der die Anzahl der benötigten Monteure angibt. Der Rückgabewert der Methode soll anzeigen, ob noch genügend Plätze frei waren oder nicht. Geben Sie die komplette Metode an.

Aufgabe 2 (15 Punkte)

Gegeben sei folgendes UML-Klassen-Diagramm:

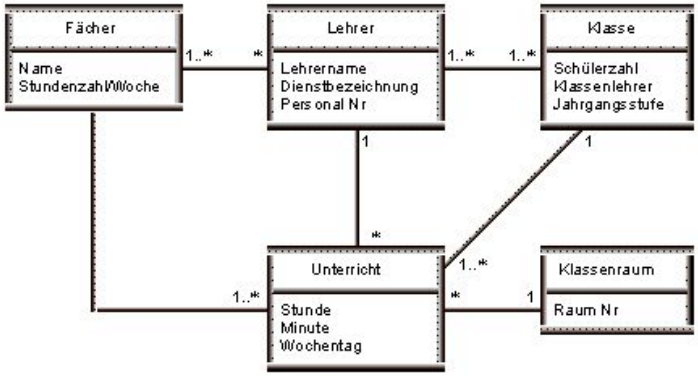

Abbildung 1:

Beachten Sie auf die Darstellung der Kardinalitäten und Assoziationen!

Hinweis: Konstruktoren, `getter`- und `setter`-Methoden müssen Sie nicht angeben und können diese als vorhanden voraussetzen.

a) Geben Sie die Klasse `Lehrer` als exakt entsprechende Java-Klasse an.

b) Geben Sie die Klasse `Unterricht` als exakt entsprechende Java-Klasse an.

Aufgabe 3 (5 Punkte)

Gegeben sei folgende Klasse `Test`.

```
1  public class Test {
2          private static int anzahl;
3          private static Auto a1, a2;
4          public static double quotient;
5          public static void main(String[] args) {
6                  quotient = 7.5;
7                  a2 = new Auto();
8                  a1 = a2;
9                  a2 = null;
10                 anzahl++;
11                 System.out.println("Anzahl: "+anzahl);
12         }
13 }
```

Es existiere außerdem eine Klasse `Auto` mit Standardkonstruktor. Es wird die `main`-Methode der Klasse `Test` aufgerufen.

Kreuzen Sie die Aussagen an, die *korrekt* sind:

☐ Das Programm `Test` besitzt keine einzige Instanzvariable.

☐ Die variablen `a1` und `a2` sind Klassenvariablen vom Referenztyp.

☐ Dadurch, dass `a2=null` gesetzt wird, ist das Auto-Objekt fortan unreferenziert und wird vom Garbage-Collector aus dem Speicher entfernt.

☐ Der Wert der Variablen `anzahl` in der Ausgabe in Zeile 11 ist nicht genau bestimmt, da die Variable `anzahl` nicht initialisiert wurde.

☐ Die Variable **quotient** ist eine Klassenvariable eines primitiven Datentyps.

Aufgabe 4 (10 Punkte)

Gegeben sei der folgende Sachverhalt:

Jede `Person` hat einen `namen`, eine `telefonnummer` und `eMail`.

Jede `Wohnadresse` wird von nur einer `Person` bewohnt. Es kann aber sein, dass einige Wohnadressen nicht bewohnt sind. Eine `Wohnadresse` enthält eine `straße`, eine `stadt`, eine `plz` und ein `land`.

Alle Wohnadressen können über eine Operation bestätigt werden und als Beschriftung (für Postversand) gedruckt werden.

Es gibt zwei Sorten von Personen: `Student`, welcher sich für ein Modul einschreiben kann und `Professor`, welcher einen `lohn` hat. Der `Student` besitzt eine `matrikelnummer` und eine `durchschnittsnote`.

Modellieren Sie diesen Sachverhalt mit einem UML-Klassendiagramm.

Aufgabe 5 (5 Punkte)

Betrachten Sie die folgenden Aussagen im Kontext von Klassen und Objekten.

Kreuzen Sie die Aussagen an, die *korrekt* sind:

☐ Konstruktoren besitzen niemals einen Rückgabetyp.

☐ Methoden und Datenfelder mit dem Zugriffsmodifikator `private` können unter keinen Umständen aus fremden Paketen angesprochen werden.

☐ `this` ist eine Referenz auf die eigene Klasse.

☐ Variablen werden nur dann automatisch mit Default-Werten initialisiert, wenn es sich um lokale Variablen (z.B. in Methodenblöcken) handelt.

☐ Einer als `final` spezifizierten Variaben kann man genau einmal einen Wert zuweisen.

Zusatzblatt