Name: _____ Mat.Nr.: _____

Aufgabe 1

Gegeben ist die Klasse *Wohnung* mit den folgenden Methoden:

```
class Wohnung
{
    /* Liefert die Groesse der Wohnung in qm. */
    double getGroesse() {...}

    /* Liefert den Mietpreis fuer die Wohnung. */
    double getMiete() {...}
}
```

Schreiben Sie eine Klasse *Wohnungssuche* mit folgender Methode:

```
    /*
     * Liefert die Liste aller Wohnungen in wohnungsliste
     * mit zumindest der angegebenen mindestgroesse und
     * hoechstens der angegebenen maximalmiete zurueck.
     */
    ArrayList<Wohnung> findeWohnung(
                            ArrayList<Wohnung> wohnungsliste,
                            double mindestgroesse,
                            double maximalmiete)
    {...}
```

Aufgabe 2

Gegeben ist die Klasse *LKW* mit den folgenden Methoden:

```
class LKW
{
    /*
     * Beauftragt den LKW, fuer den angegebenen Tag den
     * auftrag zu uebernehmen. Wenn der LKW fuer diesen
     * Tag schon einen Auftrag uebernommen hat, kann er
     * keinen weiteren Auftrag uebernehmen, und es wird
     * false zurueckgeliefert. Ansonsten wird true
     * zurueckgeliefert.
     */
    boolean uebernimmAuftrag(int tag, String auftrag)
    {...}
}
```

Implementieren Sie die Klasse *Fuhrpark* mit einem Konstruktor

Fuhrpark(),

der einen leeren Fuhrpark anlegt, und einer Methode

boolean addLKW(LKW einLKW),

die einen LKW zum Fuhrpark hinzufügt. Die Methode *addLKW()* liefert *false* zurück, wenn der LKW schon im Fuhrpark enthalten ist, ansonsten liefert die Methode *true* zurück.

Implementieren Sie weiters die Methode

boolean addAuftrag(LKW einLKW, int tag, String auftrag),

die den angegebenen LKW zur Übernahme des Auftrags für den angegebenen Tag auffordert. Wenn der LKW nicht im Fuhrpark enthalten ist, wird der Auftrag nicht an den LKW weitergegeben und die Methode liefert *false* zurück. Die Methode liefert auch *false* zurück, wenn der LKW den Auftrag nicht übernehmen kann, ansonsten wird *true* zurückgeliefert.

Aufgabe 3

Gegeben ist die Klasse *Fahrzeug* und das Interface *PrivateNutzung*:

```
class Fahrzeug
{
    /* Konstruktor, dem die Fahrzeugnummer uebergeben wird. */
    Fahrzeug(int nummer) {...}

    /* Erhoeht den Kilometerstand um die gefahrenen Kilometer */
    public void fahre(double anzahlKilometer) {...}

    /* Liefert den Kilometerstand zurueck. */
    public double getKilometerstand() {...}
}

interface PrivateNutzung
{
    /* Vermerkt die Laenge einer privaten Fahrt. */
    void fahrePrivat(double anzahlKilometer);

    /* Liefert die Summe der privat gefahrenen Kilometer. */
    double getSummePrivatfahrten();
}
```

Schreiben Sie eine Klasse *Dienstwagen*, die Unterklasse von *Fahrzeug* ist und auch das Interface *PrivateNutzung* implementiert. Diese Klasse soll über einen Konstruktor

Dienstwagen(int nummer, String dienstnehmer)

verfügen, dem die Nummer des Fahrzeugs und der Name des Dienstnehmers übergeben wird.
Überschreiben Sie in *Dienstwagen* auch die Methode *toString()* so, dass die Nummer des Dienstwagens, der Dienstnehmer, und der Kilometerstand zurückgeliefert werden.
Achten Sie darauf, dass die Methode *getKilometerstand()* auch für Dienstwägen korrekt funktioniert.