Name: _____ Matrikelnummer: _____

Aufgabe 1

In dieser Aufgabe sollen Sie die Klassen *MailServer* und *MyRecipient* implementieren. Vorgegeben ist das Interface *MailRecipient*:

```
interface MailRecipient
{
     public void receive(String emailText);
}
```

Die Klasse *MailServer* soll über einen parameterlosen Konstruktor sowie die folgenden Methoden verfügen:
Mit der Methode

boolean register(String emailAddress, MailRecipient recipient)

können sich MailRecipients unter Angabe einer Email-Adresse beim MailServer anmelden. Wenn schon ein MailRecipient mit derselben Email-Adresse beim MailServer angemeldet ist, liefert die Methode *false* zurück und die Anmeldung schlägt fehl. Ansonsten liefert die Methode *true* zurück.
Mit der Methode

boolean send(String emailAddress, String emailText)

kann eine Email an den MailServer geschickt werden, der diese dann an den Adressaten weiterleitet. Dazu prüft der MailServer, ob ein MailRecipient mit der entsprechenden Adresse angemeldet ist. Ist das nicht der Fall, liefert die Methode *false* zurück und führt keine weiteren Aktionen aus. Gibt es einen MailRecipient mit der angegebenen Adresse, dann leitet der MailServer den Text der Email mittels der Methode *receive()* an den MailRecipient weiter, und die Methode *send()* liefert *true* zurück.

Die Klasse *MyRecipient* soll das Interface *MailRecipient* implementieren und alle empfangenen Emails mittels *System.out.println()* ausgeben. Weiters soll die Klasse *MyRecipient* über einen Konstruktor

MyRecipient(String emailAddress, MailServer server)

verfügen, der das erzeugte *MyRecipient*-Objekt unter der angegebenen Email-Adresse beim angegebenen MailServer anmeldet. Falls die Anmeldung erfolglos ist, soll mittels *System.out.println()* eine Fehlermeldung ausgegeben werden.

Hinweis 1: Es muss möglich sein, dass mehrere MailRecipients beim MailServer angemeldet sind.

Hinweis 2: Das Projekt Mail-System aus dem BlueJ-Buch ist zur Lösung der Aufgabe nicht hilfreicher als andere Beispiele des Buches.

Aufgabe 2

Schreiben Sie eine Klasse *Summe* mit einer Methode

boolean istSumme(int sum, int[] a),

die berechnet, ob es durch Addition von maximal 3 Elementen aus dem Array *a* möglich ist, genau den Wert *sum* zu erzielen. Dabei darf ein Element von *a* nicht mehrfach verwendet werden.

Beispiele:

`istSumme(5,{1,3,7})==false`,
`istSumme(4,{1,3,7})==true`,
`istSumme(21,{13,5,7,3})==true`,
`istSumme(22,{13,5,7,3})==false`.

Aufgabe 3

Gegeben ist eine Klasse *Ort* mit der Methode

double getDistanzZu(Ort andererOrt),

die die Distanz zu dem anderen Ort zurückliefert. Weiters ist die Unterklasse *Tankstelle* der Klasse Ort gegeben.

Schreiben Sie eine Klasse *LKW* mit dem Konstruktor

LKW(double reichweite),

dem die Reichweite des LKWs bei vollem Tank übergeben wird. Die Klasse LKW soll weiters die Methode

ArrayList<Tankstelle> berechneTankstops(Ort[] route)

enthalten, die berechnet, an welchen Orten der angegebenen Route der LKW aufgetankt werden muss, um sein Ziel zu erreichen. Sie können annehmen, dass die Route aus zumindest zwei Orten besteht, und dass die Fahrt des LKWs voll getankt im Ort *route[0]* beginnt.
Natürlich kann der LKW nur an Orten betankt werden, die Tankstellen sind. Dies kann z.B. mittels `(route[i] instanceof Tankstelle)` überprüft werden.
Liegen die Tankstellen soweit auseinander, dass der LKW die Route nicht abfahren kann, dann soll die Methode *null* zurückliefern.

[Optional und ergibt Zusatzpunkte: Die Anzahl der Tankstops soll möglichst gering sein.]