Name: _____ Matrikelnummer: _____

Aufgabe 1

In einem Lager ist eine gewisse Anzahl von zwei unterschiedlichen Artikeln vorhanden. Die Artikel haben auch unterschiedlichen Wert. Einer Bestellung wird vom Lager nur dann entsprochen, wenn die bestellten Mengen beider Artikel geliefert werden können.
Aus einer Liste von Bestellungen kann das Lager in der Regel nicht alle Bestellungen erfüllen. Aus der Liste der Bestellungen sollen jene ausgewählt und erfüllt werden, sodass der Gesamtwert der gelieferten Artikel maximal ist.

Gegeben ist die Klasse *Lager* mit den Methoden

public int getMenge(int artNr)

und

public int getWert(int artNr),

die die vorhandene Menge und den Wert für Artikel 0 bzw. 1 zurückliefert (*artNr* muss 0 oder 1 sein). Weiters gegeben ist die Klasse *Bestellung* mit der Methode

public int getAnzahl(int artNr),

die die Anzahl der bestellten Artikel 0 bzw. 1 zurückliefert.

Schreiben Sie eine Klasse *Planung* mit der Methode

public ArrayList<Bestellung> liefere(Lager einLager,
ArrayList<Bestellung> bestellungen),

die jene Bestellungen zurückliefert, die geliefert werden sollen, um den Gesamtwert der gelieferten Artikel zu maximieren. (Es kann nicht mehr geliefert werden als im Lager vorhanden ist.)

Aufgabe 2

Schreiben Sie eine statische Methode

static ArrayList<String> findePartitionen(int[] liste, int n),

die alle Paare von Zahlen in *liste*, die addiert *n* ergeben, in einer ArrayList zurückgibt. Jedes Paar (*x*,*y*) mit *x*+*y*=*n* soll dabei als String der Form "*x*+*y*" in der ArrayList gespeichert werden, wobei der kleinere Wert des Zahlenpaares jeweils zuerst anzugeben ist. Jedes Paar soll nur einmal gespeichert werden, auch wenn es in *liste* mehrfach vorkommt.

Beispiele:

```
findePartitionen([6,2,2,4,5,1,4],8) == ["2+6","4+4"],
findePartitionen([2,6,2,6],8) == ["2+6"],
findePartitionen([4,5,1,6],8) == [].
```

Aufgabe 3

Gegeben ist die Klasse *Spielkarte*, die das Interface *Comparable<Spielkarte>* implementiert.

Schreiben Sie eine Klasse *Kartenspiel* mit dem Konstruktor

Kartenspiel (Set<Spielkarte> hand1, Set<Spielkarte> hand2),

der zwei Spielern ihre Karten zuteilt und festlegt, dass der erste Spieler (mit *hand1*) ausspielt, d.h. beginnt. Sie können davon ausgehen, dass *hand1* und *hand2* gleich viele Karten enthalten, und dass alle vorhandenen Karten verschieden sind. Die Klasse *Kartenspiel* soll auch die Methoden

void spieleEineRunde()

und

int werSpieltAus()

enthalten.

Die Methode *spieleEineRunde()* führt eine Runde des Kartenspiels nach folgenden Regeln durch:

- Der ausspielende Spieler spielt seine höchste Karte aus.
- Falls der andere Spieler die ausgespielte Karte stechen kann (d.h. eine höhere Karte besitzt), so soll er die ausgespielte Karte mit der niedrigstmöglichen Karte stechen.
- Falls der andere Spieler die ausgespielte Karte nicht stechen kann, so soll er seine niedrigste Karte zugeben.
- Die beiden gespielten Karten werden entfernt.

In der nächsten Runde des Spiels spielt jener Spieler aus, der den Stich gemacht hat. Die Methode *werSpieltAus()* liefert – auch schon vor der ersten Spielrunde – als Nummer des ausspielenden Spielers entweder 1 oder 2 zurück.
(Sie können davon ausgehen, dass *spieleEineRunde()* nur aufgerufen wird, wenn noch Karten vorhanden sind.)