Name:

Matrikelnummer:

Aufgabe 1. Gegeben sei eine Klasse *Bahnhof* mit Unterklasse *Hauptbahnhof*. Weiters gebe es eine Klasse *Zug* mit Konstruktor

```
1   public Zug(int nummer),
```

der einen Zug mit der angegebenen Nummer erzeugt, sowie Methoden

```
public void setStrecke(ArrayList<Bahnhof> strecke),
public ArrayList<Bahnhof> getStrecke(),
```

zum Setzen bzw. Auslesen jener Bahnhöfe, in denen der Zug hält.

Weiters verfüge die Klasse *Zug* über eine Methode

```
public Map<Bahnhof,Zug[]> getAnschluesse(),
```

die die Anschlusszüge in den Aufenthaltsbahnhöfen zurückgibt.

Schreiben Sie eine Klasse *Eilzug*, die von *Zug* erbt. Die Klasse *Eilzug* soll über einen Konstruktor

```
public Eilzug(int nr, ArrayList<Bahnhof> strecke)
```

verfügen, der einen Eilzug mit Nummer *nr* anlegt, der auf den in *strecke* enthaltenen Hauptbahnhöfen hält. Überschreiben Sie dementsprechend die Methode *setStrecke*, sodass nur die Hauptbahnhöfe in der übergebenen ArrayList berücksichtigt werden. Überschreiben Sie auch die Methode *getAnschluesse*, sodass jeweils nur Anschlüsse in Hauptbahnhöfen zurückgegeben werden. Die Methode *getStrecke* soll nicht überschrieben werden.

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 angegeben wird. Jedes Paar soll nur einmal gespeichert werden, auch wenn es in *liste* mehrfach vorkommt.

Beispiele:
findePartitionen([2,6,2,4,5,1,4],8) = ("2+6", "4+4")
findePartitionen([2,6,2,6],8) = ("2+6")
findePartitionen([4,5,1,6],8) = ()

(Bitte wenden!)

Aufgabe 3.

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

Schreiben Sie eine Klasse *Kartenspiel* mit Konstruktor

```
Kartenspiel (TreeSet<Spielkarte> hand1,
                    TreeSet<Spielkarte> hand2),
```

der den beiden Spielern ihre Karten zuteilt und festlegt, dass der erste Spieler ausspielt (d.i. beginnt). Sie können davon ausgehen, dass beide Spieler gleich viele Karten zugeteilt bekommen.

Schreiben Sie eine Methode

```
boolean stich(),
```

die einen Stich gemäß folgenden Regeln simuliert: Der ausspielende Spieler spielt zunächst seine höchste Karte aus. Falls der andere Spieler die ausgespielte Karte stechen kann (d.h. eine höhere Karte hat), so soll er die ausgespielte Karte mit der niedrigstmöglichen Karte stechen. Falls der andere Spieler keine höhere Karte als die ausgespielte hat, soll er seine niedrigste Karte zugeben. Die Methode soll *true* zurückgeben, wenn der ausspielende Spieler den Stich macht, ansonsten *false*. Außerdem soll die Methode festlegen, dass der Spieler, der den Stich macht, als nächstes ausspielt.