

**Name:**

**Matrikelnummer:**

### Aufgabe 1.

Gegeben sei eine Klasse *Spielkarte*, die über eine Methode

```
1 public static boolean sindEinPaar(Spielkarte s1, Spielkarte s2)
```

verfügt, die genau dann *true* zurückgibt, wenn die Karten *s1*, *s2* ein Paar bilden. Dabei können Sie davon ausgehen, dass es zu jeder Karte *s1* höchstens *eine* andere Karte *s2* gibt, sodass *s1* und *s2* zusammen ein Paar bilden (wie zum Beispiel in "Schwarzer Peter".) Weiters seien die Methoden *equals* und *hashCode* aus *Object* in *Spielkarte* überschrieben.

Schreiben Sie eine Klasse *Kartenspieler* mit Konstruktor

```
public Kartenspieler(Set<Spielkarte> dieKarten),
```

der einen Kartenspieler mit Karten *dieKarten* anlegt.

Weiters soll die Klasse *Kartenspieler* über eine Methode

```
public void entfernePaare()
```

verfügen, die alle Paare aus den Karten des Kartenspielers entfernt.

### Aufgabe 2.

Gegeben sei eine Klasse *Benutzerkonto* mit Konstruktor

```
public Benutzerkonto(String benutzername, String password),
```

der ein Benutzerkonto mit entsprechendem Benutzernamen und Passwort anlegt. Die Klasse *Benutzerkonto* verfügt außerdem über die Methode

```
public boolean anmelden(String username, String password),
```

mit der man sich unter Angabe von Benutzernamen und Passwort am Benutzerkonto anmelden kann. Bei erfolgreicher Anmeldung gibt die Methode *true* zurück, ansonsten *false*.

Weiters verfügt die Klasse *Benutzerkonto* über eine Methode

```
public void sperren(),
```

die das Benutzerkonto sperrt, sodass eine Anmeldung nicht mehr möglich ist.

Schreiben Sie eine Unterklasse *SicheresBenutzerkonto* von *Benutzerkonto* mit Konstruktor

```
public SicheresBenutzerkonto(String uname, String pwd).
```

Überschreiben Sie weiters die Methode *anmelden* so, dass der Account nach drei *aufeinanderfolgenden* erfolglosen Anmeldeversuchen gesperrt wird.

### Aufgabe 3.

Gegeben seien die Klassen *Test* und *Pruefling*. Die Klasse *Pruefling* verfügt über die Methoden

```
public boolean sitztNeben(Pruefling derAnderePruefling),  
public boolean sitztHinter(Pruefling derAnderePruefling),
```

die genau dann *true* zurückgeben, wenn der Prüfling neben bzw. hinter *derAnderePruefling* sitzt. Weiters sind in *Pruefling* die Methoden *equals* und *hashCode* aus *Object* überschrieben.

Schreiben Sie eine Klasse *Pruefungszuweisung* mit einer Methode

```
public static Map<Pruefling,Test> weiseZu(  
    ArrayList<Pruefling> diePrueflinge, Test testA, Test testB),
```

die jedem Prüfling in *diePrueflinge* einen der beiden Tests *testA*, *testB* so zuweist, dass zwei Prüflinge, die nebeneinander oder hintereinander sitzen, verschiedene Tests zugewiesen bekommen. Die Zuordnung der Tests soll in einer *Map* gespeichert und zurückgegeben werden.

*Hinweis:* Sie können der Einfachheit halber davon ausgehen, dass eine passende Zuordnung immer möglich ist, und dass die Zuweisung eines beliebigen Tests an *einen* der Prüflinge die Zuordnung der Tests an alle anderen Prüflinge eindeutig bestimmt.

*Hilfestellung:* Schreiben Sie eine Hilfsmethode

```
public static void erweitereZuweisung(  
    ArrayList<Pruefling> diePrueflinge,  
3     Map<Pruefling,Test> zuweisung, Test testA, Test testB),
```

die eine bestehende, noch nicht vollständige Zuweisung um einen oder mehrere Prüflinge aus *diePrueflinge* erweitert.