Aufgabe 1: Aufwärmen

a) Die folgenden Zuweisungen sind zum Teil falsch. Umkreisen Sie die 6 Zuweisungen, die der Compiler **nicht** erlauben würde. (6 Punkte)

1.) int x = 34;
2.) float a = 3.4;
3.) int g = 17.3;
4.) int y = x;
5.) short n = 12;
6.) int d = 2 * 2.5f;
7.) boolean empty = x;
8.) double v = x;
9.) v = 2.3f;
10.) short s = y;
11.) empty = (y == 0);
12.) float w = v + 2.5f;

b) Schreiben Sie für eine Klasse *Produkt* eine verändernde Methode `bestandAendern`, die in ihren Methodenparametern ein Zeichen (Parameter `z`) und einen float-Wert (Parameter `delta`) übergeben bekommt. Bei '+' in `z` soll sie auf das Attribut `bestand` den Betrag in `delta` aufaddieren, bei '-' davon abziehen. Der Wert von `bestand` darf dabei nicht größer als 100 und nicht kleiner als 0 werden. Bei anderen Zeichen soll eine Fehlermeldung auf der Konsole ausgegeben werden. Benutzen Sie eine *switch*-Anweisung. (9 Punkte)

```
public void bestandAendern(char z, float delta){
    switch(z){
    case '+': bestand = bestand + delta;
              if(bestand>100){ bestand = 100;} break;
    case '-': bestand = bestand - delta;
              if(bestand<0){ bestand = 0;} break;
    default: System.out.println("Ungültiger Parameter z");
```

c) Schreiben Sie eine Anweisung, die die Zeichenkette in der Variablen `wort` so zerlegt, dass das folgende Array dabei herauskommt: { "dies", "ist", "kein", "wort" }. Weisen Sie dieses Rückgabe-Array der Variablen `wortArray` vom Typ Array von *String* zu. Nutzen Sie dazu geeignete Methoden der Klasse *String*. (3 Punkte)

```
String wort  =  "DIES ## Ist ## kein ## WOrt";
String[] wortArray;
```

```
wort = wort.toLowerCase();
wort =
wortArray = wort.split(" ## ");
```

↑ Hier sind die Leerzeichen schon drin!

Aufgabe 2: Sammlungen, Schleifen, Klassenkonstanten, -methoden

a) Ergänzen Sie die Klasse *SchrittZaehler* um ein privates Attribut `kmProWoche`, das die Anzahl der zurückgelegten Kilometer für jede Woche eines Jahres in einem *Array* von Gleitkomma-Werten aufnehmen kann. Die Anzahl der Wochen im Jahr (52) und die Länge eines Schrittes in km (0,0006) sollen als private Klassenkonstanten angelegt werden. Ergänzen Sie den Konstruktor um entsprechende Zuweisungen. (7 Punkte)

Aufgabe 2: Sammlungen, Schleifen, Klassenkonstanten, -methoden

a) Ergänzen Sie die Klasse *SchrittZaehler* um ein privates Attribut `kmProWoche`, das die Anzahl der zurückgelegten Kilometer für jede Woche eines Jahres in einem *Array* von Gleitkomma-Werten aufnehmen kann. Die Anzahl der Wochen im Jahr (52) und die Länge eines Schrittes in km (0,0006) sollen als private Klassenkonstanten angelegt werden. Ergänzen Sie den Konstruktor um entsprechende Zuweisungen. (7 Punkte)

```
public class SchrittZaehler {
   private String jahr;
   private float[] kmProWoche;
   private static final float LAENGE_EINES_SCHRITTES = 0.0006f;
   private static final int WOCHEN_PRO_JAHR = 52;

   public SchrittZaehler (String jahr)
   {
      float[] kmProWoche = new float[WOCHEN_PRO_JAHR];

   }
   ...
```

b) Schreiben Sie für die Klasse *SchrittZaehler* eine öffentliche Klassenmethode `berechneAnzahlKm` die ein *Array* mit ganzzahligen Werten übergeben bekommt. Das *Array* enthält für eine beliebige Menge von Tagen die Anzahl an Schritten des jeweiligen Tages. Die Methode soll aus den Werten im *Array* die Gesamtsumme der zurückgelegten Kilometer berechnen und diesen Wert als Rückgabewert liefern. (7 Punkte)

(7)

```
public ~~float~~ static float berechneAnzahlKm (~~float~~ int[] schritteProTag) { 1,5
float strecke;  for(int zaehler = 0; zaehler < schritteProTag.length; zaehler++) { 2
        if(schritteProTag != ~~0~~ null) {
~~float strecke~~ (schritteProTag * LAENGE_EINES_SCHRITTES);
strecke = strecke +                                   1,5
    }
}
return strecke;  1
}
```

c) Schreiben Sie für die Klasse *SchrittZaehler* eine öffentliche Methode `setzeWochenKm`, die als Argumente eine Woche (als Zahl zwischen 1 und 53) und ein *Array* mit der Schrittanzahl für jeden Tag der angegebenen Woche übergeben bekommt. Die Methode soll aus den Tageswerten die Anzahl der in der Woche gelaufenen Kilometer berechnen und den ermittelten Wert in dem Array `kmProWoche` für die angegebene Woche setzen. Sorgen Sie dafür, dass die *Array*-Grenzen eingehalten werden und vermeiden Sie Codeduplikation. (5 Punkte)

(1)

```
public void setzeWochenKm (int woche, int[] schritteTag) {
~~void~~ ~~kmProWoche~~
for(
```

Aufgabe 3: HashMap

Es sei die folgende Klasse gegeben:

```
public class Rezept {
   private int sterne;
   private String bezeichner;
   private String beschreibung;

   ...

   public int getSterne() {
      return sterne;
   }
   public String getBezeichner() {
     return bezeichner;
   }
   public String getBeschreibung() {
     return beschreibung;
   }
}
```

a) Deklarieren und Initialisieren Sie ein privates Attribut `kochbuch` vom Typ *HashMap* in der man *Rezept*-Objekte unter dem Bezeichner des Rezepts abspeichern kann.

(2 Punkte)

private HashMap<String, Rezept> kochbuch = new HashMap<String, Rezept>();

b) Schreiben Sie eine Methode `einfuegen`, die ein *Rezept*-Objekt übergeben bekommt und in die *HashMap* in `kochbuch` einfügt. Sorgen Sie dafür, dass ein unter dem Bezeichner bereits existierendes Rezept nur durch ein Rezept mit einer höheren Anzahl an Sternen überschrieben werden darf und dass es keine *NullPointerException* gibt. (8 Punkte)

```
public void einfügen(Rezept neuesRezept){
    Iterator<String> it = kochbuch.keySet().iterator();
    while(it.hasNext()){
        if(it.getBezeichner.equals(neuesRezept.getBezeichner)){
            if(it.getSterne < neuesRezept.getSterne){
                kochbuch.put(neuesRezept.getBezeichner(), neuesRezept);
            }
        }
    }
    if(neuesRezept == null){ system.out.println("neues Rezept nicht gefunden"); }
```

c) Schreiben Sie eine Methode `gibRezeptBeschreibung`, die als Argument einen Rezeptbezeichner bekommt und als Rückgabewert die Beschreibung des zugehörigen Rezepts zurück liefert. Sollte zu dem Bezeichner kein Rezept existieren, soll eine Fehlermeldung ausgegeben und der Wert *null* zurück geliefert werden. (6 Punkte)

```
public String gibRezeptBeschreibung(String derBezeichner){
    Iterator<String> it = kochbuch.keySet().iterator();
    while(it.hasNext()){
        if(it.getBezeichner().equals(derBezeichner)){
            return it.getBeschreibung();
        }
    }
    system.out.println("kein Rezept gefunden.");
    return null;
}
```

Aufgabe 4: Vererbung / Überschreiben von Methoden

Schreiben Sie geeignete Klassen, die folgende Zusammenhänge modellieren:

Auf einem Flug können Business- und Economy-Passagiere mitfliegen. Alle Passagiere besitzen einen Namen, eine email-Adresse und einen Bonuspunktestand. Ein Passagier kann für bestimmte Sonderleistungen Bonuspunkte verbrauchen (mit der Methode `verringereBonuspunkte(int punkte)`). Bei jedem Passagier wird der Bonuspunktestand nach jedem Flug, in Abhängigkeit der zurückgelegten Meilen, um einen bestimmten Betrag erhöht (mit der Methode `erhoeheBonuspunkte(int flugmeilen)`). Die Berechnung erfolgt je nach Passagier anders. Folgende Eigenheiten sollen für die Passagiere modelliert werden:

- Für einen Business-Passagier wird abgespeichert, ob er vegetarisches Essen bevorzugt. Außerdem wird die Nummer seiner PremiumCard gespeichert, die jeder Business-Passagier besitzt. Am Ende eines Fluges wird ihm die doppelte Anzahl an Flugmeilen gutgeschrieben.
- Für einen Economy-Passagier werden die Ausgaben gespeichert, die er auf einem Flug für zusätzliche Leistungen getätigt hat (mit der Methode `ausgabenHinzufuegen(float ausgabe)`). Dieser Betrag wird am Ende des Fluges, mit dem Faktor 5 multipliziert, auf die Flugmeilen des Fluges addiert und im Bonuspunktestand gutgeschrieben.

Tipp: Um einen *float*-Wert in einen *int*-Wert zu runden können Sie die Klassenmethode `int round(float wert)` der Klasse *Math* benutzen.

Schreiben Sie die nötigten Klassen mit den zugehörigen Methoden. Überlegen Sie, welche Attributwerte im Konstruktor übergeben werden sollten. Implementieren Sie an geeigneter Stelle die oben genannten Methoden. Setter- und getter-Methoden können Sie weglassen. Überlegen Sie, ob man von allen Klassen Instanzen bilden sollte.

(20 Punkte)

7,5

```
public abstract class Passagier {
  protected String name;
  private String email;
  protected int punkte;

  public Passagier (String name, String email, int punkte){
    this.name = name;
    this.email = email;
    this.punkte = punkte;
  }

  public void verringere Bonuspunkte (int punkte){
    this.punkte = this.punkte - punkte;
  }

  protected abstract void erhoehe Bonuspunkte (int flugmeilen);
}
```

```
public class Business extends Passagier {
    private boolean veg;
    private String premiumCard;

    public Business(String name, String email, int punkte, boolean veg,
                    String premiumCard) {
        super(name, email, punkte);
        this.veg = veg;
        this.premiumCard = premiumCard;
    }

    private void erhoeheBonuspunkte(int flugmeilen) {
        punkte = punkte + (2 * flugmeilen);
    }
}
```

(5)

(7)

```
public class Economy extends Passagier {
private float ausgaben;

    public Economy (String name, String email, int punkte){
        super(name, email, punkte);
        ausgaben = 0.00f;
    }

    private void erhoeheBonuspunkte ( int flugmeilen){
punkte = punkte + ((ausgaben * 5) + flugmeilen);
    }

private void ausgabeHinzufügen (float ausgabe){
    ausgaben = ausgaben + Math.round(ausgabe);
}
```

Aufgabe 5: ArrayList, Substitution

Es sei *DVD* eine Superklasse der Klassen *Film* und *Spiel*. Jedes Objekt vom Typ *DVD* besitzt eine Bestellnummer, die mit der Methode `String getBestellnummer()` und einen Titel, der mit der Methode `String getTitel()` abgefragt werden kann.

a) Deklarieren und Initialisieren Sie eine Variable `dvds`, mit einer *ArrayList*, die sowohl Objekte der Klasse *Film* als auch *Spiel* aufnehmen kann. (1 Punkt)

```
protected ArrayList<DVD> dvds = new ArrayList<DVD>();
```

1

b) Schreiben Sie eine Methode `entferneSpiele()`, in der mit einem Iterator über alle Objekte in `dvds` iteriert wird. Wenn es sich bei dem Objekt um ein Spiel handelt, so soll das Objekt aus der Liste entfernt werden. (7 Punkte)

```
public void entferneSpiele(){
[illegible]
[illegible]
[illegible]
[illegible]
Iterator<DVD> it = dvds.iterator();                    2
While(it.hasNext()){                                   1
    if(it.Next() instance of Spiel){                   2,5
        it.remove();                                   1,5
    }
}
}
```

7

c) Schreiben Sie eine Methode `gibTitelZuBestellnummer`. Diese bekommt die Bestellnummer einer DVD übergeben und gibt den Titel der entsprechenden DVD zurück. Sollte es keine DVD mit der Bestellnummer geben, so soll der Wert *null* zurück geliefert und eine Fehlermeldung auf der Konsole ausgegeben werden. (7 Punkte)

```
public String gibTitelZuBestellnummer(String bestellnummer){
for(DVD diesesDVD : dvds){
    if(diesesDVD.getBestellnummer().equals(bestellnummer){
      return diesesDVD.getTitel();
    }
  else{
      System.out.println("keine DVD unter dieser Bestellnummer
                          gefunden");
      return null;
    }
}
}
```

Aufgabe 6: Methoden *toString* und *equals* überschreiben

Unten sehen Sie die Klasse *Mitarbeiter*, die von der Klasse *Person* abgeleitet ist. Gehen Sie davon aus, dass sowohl die Klasse *Person* als auch die Klasse *Bankverbindung* die Methoden `toString` und `equals` mit ihrer eigenen Implementierung überschrieben haben.

```
public class Mitarbeiter extends Person {

  private String id;

  private Bankverbindung konto;

  ...
}
```

a) Überschreiben Sie die Methode `equals` in der Klasse *Mitarbeiter*. Gehen Sie davon aus, dass zwei Mitarbeiter identisch sind, wenn sie in allen relevanten von *Person* geerbten Attributwerten übereinstimmen und dieselbe Mitarbeiter-Id besitzen.

(7 Punkte)

```
public boolean equals(Object o){
if([illegible]
    o.getID().equals(id)        && [illegible]
                                   o.konto == konto
    return true;

else {
    return false;
  }

}
```

b) Überschreiben Sie die Methode `toString` in der Klasse *Mitarbeiter*.

(5 Punkte) (4)

```
public String toString(){   String ausgabe = "";
ausgabe = ausgabe + super.toString();
ausgabe = ausgabe + "ID: " + id + "\n" + "Konto: " +
          konto.getKontonummer() + "\n";
return ausgabe;
}
```