Aufgabe 1: Aufwärmen

Die folgenden Zuweisungen sind zum Teil falsch. Umkreisen Sie die 6 Zuweisungen, die der Compiler nicht erlauben würde.

1.) int x = 34;

2.) boolean boo = x;

3.) int g = 17.3;

4.) int y = x;

5.) y = y + 2.5;

6.) short s;

7.) s = y;

8.) float b = 3.4;

9.) double v = x;

10.) short n = 12;

11.) v = 2.3f;

12.) int d = 2 * 2.5f;

Deklarieren Sie zwei Referenzvariablen *hund1* und *hund2* vom Typ *Hund* und erzeugen Sie eine Instanz der Klasse *Hund* mit dem Konstruktor `Hund(String name)` und verknüpfen Sie beide Variablen mit dem Objekt.

Schreiben Sie Anweisung(en), die dafür sorgen, dass die Zeichenkette in der Variablen *wort* folgendermaßen aussieht: "langeslangeslangeswort". Nutzen Sie dazu geeignete Methoden der Klasse String.

```
String wort  =  "      LangesLangesLangesWort     ";
```

Aufgabe 2: Sammlungen, Schleifen

Ergänzen Sie die Klasse *Jahrestemperaturen* um ein privates Attribut *monatlicheWerte*, das die mittleren Temperaturmesswerte für jeden Monat eines Jahres in einem *Array* aufnehmen kann. Die Messwerte sollen als Gleitkommawerte gespeichert werden. Ergänzen Sie den Konstruktor um entsprechende Zuweisungen.

(3)

```
public class Jahrestemperaturen
{
    private String jahr;

    _____ 1

    public Jahrestemperaturen(String jahr)
    {
    _____ 1

    _____ 1
    }
    ...
}
```

Schreiben Sie für die Klasse *Jahrestemperaturen* eine private Methode *berechneMonatswert*, die ein Array mit Gleitkommawerten übergeben bekommt. Das Array enthält für jeden Tag eines Monats die mittlere Tagestemperatur. Die Methode soll aus den Werten im Array das arithmetische Mittel der Temperaturen des Monats berechnen und diesen Wert als Rückgabewert liefern.

(9)

Schreiben Sie für die Klasse *Jahrestemperaturen* eine öffentliche Methode *setzeMonatsMittel*, die als Argumente die Zahl des Monats (z.B. 8 für August) und ein Array mit den mittleren Tagesmesswerten für jeden Tag des angegebenen Monats übergeben bekommt. Die Methode soll das arithmetische Mittel der Temperaturwerte des entsprechenden Monats berechnen und ihn in dem Array *monatlicheWerte* für den entsprechenden Monat setzen. Sorgen Sie dafür, dass die Array-Grenzen eingehalten werden.

(5)

Aufgabe 3: HashMap

Es sei die folgende Klasse gegeben:

```
public class Person {
   private String name;
   ...

   public String gibName() {
      return name;
   }
   ...
}
```

Deklarieren und Initialisieren Sie eine Variable *personen* vom Typ *HashMap* in der man Objekte vom Typ *Person* unter dem Namen des *Person*-Objekts abspeichern kann.

Schreiben Sie eine Methode *einfuegen(..)*, die ein Person-Objekt übergeben bekommt und das Person-Objekt in die HashMap *personen* einfügt. Sorgen Sie dafür, dass es keine *NullPointerException* gibt.

Schreiben Sie eine Methode *gibPerson(..)*, die als Argument einen Namen bekommt und als Rückgabewert das Person-Objekt mit dem angegebenen Namen aus der HashMap *personen* zurück liefert.

Aufgabe 4: Klassenkonstanten/-methoden

Schreiben Sie eine Klasse *Drucker*, die zwei Dateiformate ausdrucken kann: PDF und DOC. Die Dateiformate sollen in der Klasse als öffentliche Klassenkonstanten definiert sein.

Die Klasse soll eine öffentliche Klassenmethode

```
drucke(int format, String datei)
```

besitzen, der das Format (unter Angabe einer der Klassenkonstanten) und der Pfad der Datei übergeben wird. In der Methode *drucke* soll per Switch-Anweisung - je nach Dateiformat - eine spezielle private Klassenmethode aufgerufen werden (z.B. *druckeDOC(..)*), die das Ausdrucken der Datei bewirkt. Definieren Sie auch diese Methoden, deren Rumpf kann aber leer bleiben, d.h. `{...}`. Sollte die Methode *drucke(..)* ein unbekanntes Dateiformat übergeben bekommen, soll die Fehlermeldung „Unbekanntes Format" auf der Konsole ausgegeben werden. Die Fehlermeldung soll als private Konstante der Klasse angelegt werden.

(13)

Notieren Sie einen externen Methodenaufruf der Methode drucke(..) für das PDF-Dokument in *file*.

```
String file = "klausur.pdf";
```

4

Aufgabe 5: Vererbung / Überschreiben von Methoden

Gegeben seien folgende Klassen *Katze* und *Fisch*. Modellieren Sie eine Klassenhierarchie, die die Gemeinsamkeiten der beiden Tiere so abbildet, dass sich leicht weitere Tierarten einfügen lassen. Versuchen Sie dazu, möglichst viele allgemeine Eigenschaften und Verhaltensweisen bereits in der gemeinsamen Superklasse *Tier* anzubieten.

Schreiben Sie die drei Klassen auf und definieren Sie sowohl die Konstruktoren (deren Rümpfe hier ausgelassen wurden) als auch die entsprechenden Methoden. Sie müssen keine getter- und setter-Methoden für die Attributwerte hinzufügen. Überlegen Sie, ob es abstrakte Methoden gibt.

Klassen auf der nächsten Seite.

```
public class Katze
{
    private String typ;
    private boolean wild;  // Wild- oder Hauskatze
    private String fresschen;
    private boolean langhaarig;
    private int wieAlt;

    public Katze (String bezeichner, boolean wild, int wieAlt,
                  String nahrung, boolean langh)
    { ... }

    public void nahrungAufnehmen() {
        System.out.println("Ich fresse " + fresschen);
    }

    public void geraeuschMachen()
    {
        if(wild) System.out.println("Roaaar, roaaar!");
            else System.out.println("Miau, miau!");
    }

    public void fortbewegen() {
        System.out.println("Ich bin ein Landtier und laufe.");
    }
}


public class Fisch
{
    private String art;
    private String wasserArt; //Salz- oder Süßwasser
    private String futter;
    private float wasserTemperatur;
    private int alter;

    public Fisch(String futter, String art, float wTemp, int dasAlter,
                 String wArt)
    {...}

    public void fressen() {
        System.out.println("Ich fresse " + futter);
    }

    public void blubbern()
    {
        System.out.println("Blubb, blubb!");
    }

    public void bewegen() {
        System.out.println("Ich bin ein Wassertier und schwimme.");
    }
}
```

Hier kommt die Superklasse hin:

Hier kommt die Klasse *Katze* hin:

Hier kommt die Klasse *Fisch* hin:

Aufgabe 6: Substitutionsprinzip

Deklarieren und Intitalisieren Sie eine Variable *tiere*, mit einer *ArrayList, die* beliebige Tiere (aus Aufgabe 5) aufnehmen kann.

Schreiben Sie eine Methode *tiereAgierenLassen()*, die in einer For-Each-Schleife alle Tiere in der Variablen *tiere* durchläuft und bei jedem Tier alle verfügbaren Verhaltensweisen aufruft.

Aufgabe 7: Interfaces

Unten sehen Sie die Klasse *SongPlayer*. Diese besitzt eine einzige statische Methode *play(..)*. Die Methode kann alle Objekte abspielen, die das Interface *Abspielbar* implementieren.

```
public class SongPlayer
{

  public static void play(Abspielbar song) {
    if(song != null) {
      int typ = song.gibTyp();
      String pfad = song.gibDateipfad();

      if(pfad.equals("")) {
          System.out.println("ERROR ");
          return;
      }
      if(typ == Abspielbar.MP3) playMP3(pfad);
        else if(typ == Abspielbar.AAC) playAAC(pfad);
                else System.out.println("ERROR ");
    }
  ...
}
```

Schreiben Sie das Interface *Abspielbar*, das garantiert, dass die Objekte aller Klassen, die das Interface implementieren, von *SongPlayer* gespielt werden können.

Schreiben Sie eine Klasse *Song* die das Interface *Abspielbar* implementiert.