Übungsblatt 6: OO-Konzepte (Vererbung, Polymorphie, ...) Objekterzeugung

Aufgabe 1: Fehlersuche

Die folgenden Programme (die in den Dateien `OO1.java`, `OO2.java` und `OO3.java` gespeichert seien) enthalten Fehler, die zur Übersetzungszeit vom Compiler angezeigt werden. Beschreiben Sie jeweils das Problem:

(a)

```
class A {
    int a = 2;

    public A(int a){
        this.a = a;
    }
}

class B extends A {
    int b;

    public B(int b){
        this.b = b;
    }
}

public class OO1 {
    public static void main(){
        B b = new B(1);
    }
```

(b)

```
class A {
    int a = 2;
}
```

```
class B extends A {
    int b=0;

    public B(int b){
        this.b = b;
    }

    public B(){
        this(b);
    }
}

public class OO2 {
    public static void main(){
        B b = new B(1);
    }
}
```

(c)

```
class A {
    int a = 1;
}

class B extends A {
    int b = 2;

}

class C extends A,B {
    int c = 3;
}

public class OO3 {
    public static void main(){
        C c = new C();
    }
```

```
    }
```

Aufgabe 2: Objekterzeugung

Gegeben seien die nachfolgenden Klassen.

```
class Thomas {

  Thomas() {
    System.out.println("Thomas");
  }
}

class Maria extends Thomas {

  Maria() {
    System.out.println("Maria");
  }
}

class Susi extends Maria {
  Thomas t = new Thomas();

  Susi() {
    System.out.println("Susi");
  }
}

class TestSusi {
  Maria m = new Maria();

  static Thomas t = new Thomas();

  public static void main(String[] args) {
    Susi s = new Susi();
  }
}
```

Geben Sie an, was beim Start der Klasse `TestSusi` ausgegeben wird.

Aufgabe 3: OO (Vererbung)

Implementieren Sie die drei folgenden Klassen, die jeweils ein privates Attribut besitzen sollen sowie je eine zugehörige `get`- und `set`- Methode für dieses Attribut:

- Die Klasse `Punkt1D` soll ein Attribut `x` vom Typ `double` besitzen.
- Die Klasse `Punkt2D` soll von der Klasse Punkt1D erben und zusätzlich ein Attribut `y` vom Typ `double` besitzen.
- Die Klasse `Punkt3D` soll von der Klasse Punkt2D erben und zusätzlich ein Attribut `z` vom Typ `double` besitzen.

Statten sie alle Klassen mit einem passenden Konstruktor aus (mit einem, zwei bzw. drei Parametern für die Klassen `Punkt1D`, `Punkt2D` und `Punkt3D`). Die Konstruktoren sollen jeweils nur das in der Klasse definierte Attribut selbst initialisieren und darüberhinaus den Konstruktor der darüberliegenden Klasse aufrufen. Fügen Sie jeweils die passenden `get`- und `set`-Methoden hinzu.

Überschreiben Sie in jeder der Klassen die `toString`-Methode so, dass das der Klasse zugehörige Attribut formatiert ausgegeben werden kann und nutzen Sie in den Klassen `Punkt2D` und `Punkt3D` jeweils den Aufruf der `toString`-Methode aus der Oberklasse, um auch die geerbten Attribute mit zu erfassen.

Fügen Sie in der Klasse `Punkt3D` eine Methode hinzu, die den Abstand zu einem zweiten, als Parameter übergebenen Wert (in Form einer Instanz der Klasse `Punkt3D`) berechnet und zurückgibt.

Schreiben Sie eine `main`-Methode, in der die zuvor geschriebenen Methoden getestet werden.

Aufgabe 4: Verarbeitung von Zeichenketten

Schreiben Sie eine Klasse `StringWorker` mit einigen nützlichen Klassen-Methoden zur Verarbeitung von Strings. Benutzen Sie hierzu geeignete Methoden der Klassen `String` (z.B. die Methoden `charAt`, `indexOf`, `replace`) und `StringBuffer` (z.B. die Methode `append`).

- eine Methode, die eine Zeichenkette der Länge n aus zufällig gewählten Buchstaben erzeugt.

- eine Methode, die eine als Parameter übergebenen Zeichenkette in umgekehrter Reihenfolge zurückgibt.
  Beispiel: `reverse(‘‘abcdef‘‘)` → `‘‘fedcba‘‘`

- eine Methode, die überprüft, ob eine als Parameter übergebenen Zeichenkette ein Palindrom darstellt (d.h. von hinten und vorne gelesen identisch ist).
  Beispiel: `palindromP(‘‘abcdcba‘‘)` → `true`

- eine Methode, die einen Teil einer als Parameter übergebenen Zeichenkette umkehrt. Der umzukehrende Teil soll dabei durch einen Index i für die erste Position des Teilstrings in der Zeichenkette und einen Index j für die erste Position in der Zeichenkette, die nicht mehr Bestandteil des Teilsstrings ist, spezifiziert werden.
  Beispiel: `reverseSubstring(‘‘aaaxyzbbb‘‘,3,6)` → `‘‘aaazyxbbb‘‘`

- eine Methode, die in einer als Parameter übergebenen Zeichenkette ein einzelnes Zeichen an einer Position k durch ein anderes Zeichen (alle Werte als Parameter übergeben) ersetzt.
  Beispiel: `setChar(‘‘aaaXaaa‘‘,3,’Y’)` → `‘‘aaaYaaa‘‘`

- eine Methode, die in einer als Parameter übergebenen Zeichenkette einen Teil der Zeichenkette ab einer bestimmten Position durch eine weitere Zeichenkette (beide Werte ebenfalls als Parameter übergeben) ersetzt. Dies soll nur dann gemacht werden, wenn die ursprüngliche Zeichenkette hierfür lang genug ist.
  Beispiel: `setSubstring(‘‘aaaxyzbbb‘‘,3,‘‘ccc‘‘)` → `‘‘aaacccbbb‘‘`

- Eine Methode, die die Anzahl der Vorkommen eines Strings in einem anderen zählt. Hierbei sollen überlappende Vorkommen entsprechend mehrfach gezählt werden.
  Beispiel: `countPositions1(‘‘aaabbbaaa‘‘,‘‘aa‘‘)` → `4`

- Eine Methode, die die Anzahl der Vorkommen eines Strings in einem anderen zählt. Hierbei sollen überlappende Vorkommen nur einfach gezählt werden.
  Beispiel: `countPositions1(‘‘aaabbbaaa‘‘,‘‘aa‘‘)` → `2`

- eine Methode, die die zwei Zeichenketten zeichenweise mischt. Ist eine der beiden Zeichenketten länger als die andere, soll der Überhang hin-

tendran gehängt werden.
Beispiel: `mixStrings(‘‘abc‘‘, ‘‘xyz00‘‘)` → `‘‘axbycz00‘‘`

Ergänzen Sie die Klasse durch eine `main`-Methode mit Aufrufen der von Ihnen geschriebenen Methoden um diese zu testen.

Aufgabe 5: Stringverarbeitung, Hangman

Ziel dieser Aufgabe ist es, eine Klasse zu entwickeln, mit der man das Hangman-Spiel spielen kann. In diesem Spiel gibt es ein geheimes Wort zu erraten, wobei zunächst nur die Information preisgegeben wird, wieviele Buchstaben das Wort hat. Das Spiel besteht darin, alle Buchstaben zu erraten, die in diesem Wort vorkommen (und damit das Wort), wobei es eine festgelegte maximale Anzahl von Fehlversuchen gibt.

Als Hilfestellung sind in den Dateien `Hangman.java` und `WGetter.java` bereits einige Methoden vorgegeben. Die Klasse `Hangman` enthält die Funktionalität, die benötigt wird, eine Liste von möglichen Rate-Wörtern anzulegen und eines davon zufällig als das geheime Wort auszuwählen. Die vorgegebene Datei (`words.txt`) stellt herzu eine Liste möglicher Wörter bereit. Die Klasse `WGetter` besitzt eine Klassenmethode `getAllWords`, mit der diese Datei eingelesen und die darin enthaltenen Wörter extrahiert werden. Die Datei muss dabei im aktuellen Arbeitsverzeichnis liegen.

Implementieren Sie jetzt die noch fehlende Funktionalität, die darin besteht, eine Spiel-Methode `play` zu entwickeln und die dort auszuführenden Teilaufgaben wiederum sinnvoll in weiteren Methoden umzusetzen.

Der Spielablauf besteht dabei aus folgenden Schritten.

1. Zunächst wir immer das geheime Wort in der Form angezeigt, dass für jeden noch nicht erratenen Buchstaben ein '_' Zeichen angezeigt wird erratenen Buchstaben werden dagegen an allen Stellen, an denen Sie in dem Wort vorkommen, direkt angezeigt.

2. Dann wird der Benutzer darum gebeten, einen Buchstaben einzugeben. Die Eingabe soll solange wiederholt werden, bis der Benutzer einen Buchstaben eingegeben hat, den er noch nicht in einem früheren Versuch geraten hat (dazu muß man sich diese natürlich merken, z.B. in einer Zeichenkette).

3. Anschließend wird der eingegebene Buchstabe daraufhin geprüft, ob er in dem geheimen Wort vorkommt. Wenn nicht, wird die Anzahl

der Fehlversuche entsprechend erhöht und eine entsprechende Meldung ausgegeben.

Diese Schritte werden solange wiederholt, bis die Anzahl der verbliebenen Fehlversuche den Wert 0 erreicht hat oder alle Buchstaben, die in dem Wort vorkommen, erraten wurden.

Ein Spielverlauf könnte z.B. wie folgt aussehen:

```
java Hangman

Noch 10 Fehlversuche

-----

Buchstabe: a

Noch 10 Fehlversuche

-a---

Buchstabe: r
Der Buchstabe r kommt nicht vor.

Noch 9 Fehlversuche

-a---

Buchstabe: n

Noch 9 Fehlversuche

-a--n

Buchstabe: u
Der Buchstabe u kommt nicht vor.

Noch 8 Fehlversuche

-a--n
```

```
Buchstabe: t
Der Buchstabe t kommt nicht vor.

Noch 7 Fehlversuche

-a--n

Buchstabe: e

Noch 7 Fehlversuche

-a-en

Buchstabe: m
Der Buchstabe m kommt nicht vor.

Noch 6 Fehlversuche

-a-en

Buchstabe: p
Der Buchstabe p kommt nicht vor.

Noch 5 Fehlversuche

-a-en

Buchstabe: w
Der Buchstabe w kommt nicht vor.


Noch 4 Fehlversuche

-a-en

Buchstabe: h
```

```
Noch 4 Fehlversuche

ha-en

Buchstabe: b
haben
Herzlichen Glueckwunsch!
```