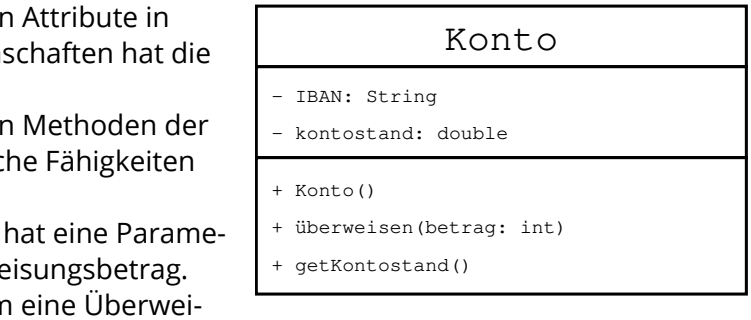
Klassen modellieren und Klassendiagramme erstellen

① Betrachte das gegebene Klassendiagramm zur Klasse **Konto**:

a) Beschreibe die angegebenen Attribute in Stichpunkten - welche Eigenschaften hat die Klasse Konto?
b) Beschreibe die angegebenen Methoden der Klasse stichpunktartig - welche Fähigkeiten hat die Klasse Konto?
c) Die Methode **überweisen()** hat eine Parameterübergabe für den Überweisungsbetrag. Welcher Parameter fehlt, um eine Überweisung korrekt ausführen zu können?
d) Die Methode **überweisen()** hat als Parameter den Integer betrag. Beurteile, ob ein Integer hier sinnvoll ist.



Quelle: pixabay.com

② Das neue Computerspiel „Masters of the Monsters“ soll entwickelt werden - ihr befindet euch in der Konzeptphase und sollt als Entwicklungsgrundlage ein Klassendiagramm mit fünf Klassen entwerfen:

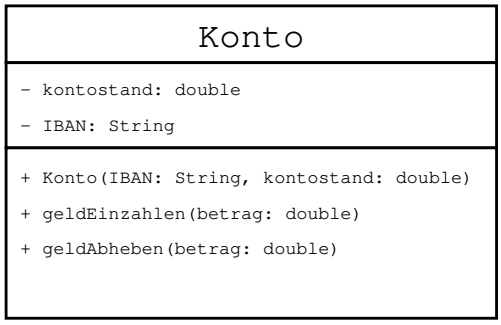
- **Spieler**: Ein Spieler hat **Benutzernamen**, **Level**, **Gesundheit** und **Punktzahl**. Ein Spieler besitzt auch eine **Waffe**.
- **Ausrüstung**: Ausrüstung hat **Name** und **Schutzwert**.
- **Rüstung**: Eine spezielle Art von Ausrüstung hat zusätzlich **Gewicht**. Sie erbt von der Klasse **Ausrüstung**.
- **Waffe**: Eine Waffe hat **Schaden**, **Typ** (z. B. Schwert, Bogen) und **Haltbarkeit**. Sie erbt von der Klasse **Ausrüstung**.
- **Helm**: Eine weitere spezielle Art von Ausrüstung, die zusätzlich **Sichtbarkeit** und **Schutzwert** hat. Sie erbt ebenfalls von der Klasse **Ausrüstung**.

③ Erweitere das Klassendiagramm um passende Methoden. Jede Klasse soll eine Methode **getImage()** besitzen, die das jeweilige Bild anzeigen kann. Spieler sollen mit einer Waffe **angreifen()** können. Ausrüstungsgegenstände sollen ihren Schutzwert und ihren Schaden ausgeben können.

④ Erweitere das Klassendiagramm von „Masters of the Monsters“ um eine Klasse **Monster**, mit der die Gegner modelliert werden sollen - überlege dir passende Attribute und Methoden.

Klassendiagramme implementieren - Klassen definieren

⑤ Implementiere die Klasse Konto im Java-Dokument **Konto.java**. Die Methode **geldEinzahlen(double betrag)** soll den übergebenen Betrag zum kontostand addieren, die Methode **geldAbheben(double betrag)** soll den übergebenen Betrag in der Kommandozeile ausgeben und vom Kontostand abziehen, wenn auf dem Konto genügend Geld vorhanden ist.



⑥ a) Erstelle in der Datei **Main.java** das Objekt *konto1* der Klasse **Konto**, IBAN: DE12345678910, Kontostand 0.0€.
b) Zahle durch einen Aufruf der Methode **konto1.geldEinzahlen(500.0)** 500€ auf das *konto1* ein.
c) Hebe durch einen geeigneten Methodenaufruf 50€ vom *konto1* ab.

Objekte instanziieren

Um ein Objekt zu erstellen schreibt man in Java:
Klassenname objektbezeichner = new Klassenname(Parameter);

⑦ ☆ Erweitere die Klasse **Konto** um eine Methode **überweisung(Konto ziel, double betrag).**

- Die Methode bekommt als Parameter das Zielkonto und den Betrag übergeben
- In der Methode soll dem Kontostand des Zielkontos der Betrag überwiesen werden und vom Kontostand des aktuellen Kontos abgezogen werden.
- Um dem Kontostand des Zielkontos Geld hinzuzufügen kann die Methode **geldEinzahlen(double betrag)** verwendet werden.
- Vor der Überweisung soll geprüft werden, ob der Kontostand für die Überweisung ausreichend ist.

⑧ Erstelle zu deinem Greenfoot-Projekt ein Klassendiagramm. Überlege dir, welche Klassen und Attribute du aktuell implementiert hast und überlege dir, welche Klassen und Attribute du benötigst, um den Rest deines Projekts implementieren zu können.

Lösungen

Java

```java
public class Konto {
    // Deklaration der Attribute
    private double kontostand;
    private String IBAN;

    // Konstruktor zur Initialisierung der Attribute
    public Konto(String IBAN, double kontostand) {
        this.kontostand = kontostand;
        this.IBAN = IBAN;
    }

    // Methode zum Einzahlen von Geld
    public void geldEinzahlen(double betrag) {
        kontostand = kontostand + betrag;
    }

    // Methode zum Abheben von Geld
    public void geldAbheben(double betrag) {
        if (kontostand >= betrag) {
            System.out.println(betrag + " wurde abgehoben");
            kontostand = kontostand - betrag;
        } else {
            System.out.println("Kontostand nicht ausreichend");
        }
    }
}

```

Lösung zu Aufgabe 5

Java

```java
public class Main {
    public static void main(String[] args) {
        // a) Erstelle das Objekt konto1 der Klasse Konto
        Konto konto1 = new Konto("DE12345678919", 0.0);

        // b) Zahle 500€ auf das konto1 ein
        konto1.geldEinzahlen(500.0);

        // c) Hebe 50€ vom konto1 ab
        konto1.geldAbheben(50.0);
    }
}

```

Lösung zu Aufgabe 6

Java

```java
// Methode für eine Überweisung
public void überweisung(Konto ziel, double betrag) {
   if (kontostand >= betrag) {
       // Betrag vom aktuellen Konto abziehen
       this.geldAbheben(betrag);
        // Betrag dem Zielkonto hinzufügen
        ziel.geldEinzahlen(betrag);
        // Erfolgsnachricht ausgeben
        System.out.println("Überweisung von " + betrag +
         " auf das Konto " + ziel + " erfolgreich.");
    }
    else {
        // Fehlermeldung bei unzureichendem Kontostand
        System.out.println("Kontostand nicht ausreichend für die Überweisung.");
    }
}

```

Lösung zu Aufgabe 7