Fortgeschrittene Programmierungsmethoden
Praktische Nachholprüfung
V1

Aufgabe 1:Archiv der Ninja-Schlachten

Aufgrund zahlreicher Konflikte in der Shinobi-Welt haben die großen Ninja-Dörfer beschlossen, ein detailliertes Archiv über Schlachten und wichtige Ereignisse in ihrer Geschichte zu führen. Diese Informationen werden in Dateien gespeichert, die eines der folgenden Formate haben können: **JSON**, **TSV** oder **XML**. Obwohl jede Datei dieselben Daten enthält, unterscheidet sich ihre Struktur je nach Format.

Datenstruktur:

Jedes Ereignis wird mit den folgenden Attributen gespeichert:

- **Id** – Einzigartige Identifikationsnummer des Ereignisses ***(int)***
- **Charaktername** – Vollständiger Name des beteiligten Ninjas ***(String)***
- **Stufe** – Ninja-Rang zum Zeitpunkt des Ereignisses ***(Enum: Genin, Chunin, Jonin, Kage)***
- **Beschreibung** – Kurze Beschreibung des Ereignisses ***(String)***
- **Datum** – Datum des Ereignisses ***(LocalDate)***
- **Kraftpunkte** – Bewertetes Kraftniveau des Ereignisses ***(double)***

Beispieldateien:

JSON (evenimente.json):

```
[
  {
    "Id": 1,
    "Charaktername": "Konan",
    "Stufe": "Kage",
    "Beschreibung": "Hat einen gefährlichen Gegner besiegt",
    "Datum": "2004-09-08",
    "Kraftpunkte": 2667.78
  },
  {
    "Id": 2,
    "Charaktername": "Hiruzen Sarutobi",
    "Stufe": "Chunin",
```

```
    "Beschreibung": "Hat ein Jutsu gemeistert",
    "Datum": "2008-03-12",
    "Kraftpunkte": 2560.78
  }
]
```

TSV (evenimente.tsv):

```
Id   Charaktername  Stufe     Beschreibung   Datum
Kraftpunkte
1    Konan     Kage Hat einen gefährlichen Gegner besiegt
2008-09-08     2667.78
2    Hiruzen Sarutobi    Chunin    Hat ein Jutsu gemeistert
2008-03-12     2560.78
3    Gaara     Jonin     Hat einen Verbündeten gerettet
2013-02-12     7035.22
```

XML (evenimente.xml):

```
<logs>
  <log>
    <Id>1</Id>
    <Charaktername>Konan</Charaktername>
    <Stufe>Kage</Stufe>
              <Beschreibung>Hat   einen   gefährlichen   Gegner
besiegt</Beschreibung>
    <Datum>2004-09-08</Datum>
    <Kraftpunkte>2667.78</Kraftpunkte>
  </log>
  <log>
    <Id>2</Id>
    <Charaktername>Hiruzen Sarutobi</Charaktername>
    <Stufe>Chunin</Stufe>
    <Beschreibung>Hat ein Jutsu gemeistert</Beschreibung>
    <Datum>2008-03-12</Datum>
    <Kraftpunkte>2560.78</Kraftpunkte>
  </log>
</logs>
```

Anforderungen für die Anwendungsentwicklung

Zur Umsetzung des Projekts muss eine Anwendung entwickelt werden, die folgende Anforderungen erfüllt. Die **MVC-Architektur** ist **nicht erforderlich**, aber **jede Funktion** sollte als **separate Methode** implementiert und in der **main**-Methode aufgerufen werden.

a) Lesen von Daten aus einer Datei (1 Punkt)

- Die Anwendung muss eine Datei im Format JSON, TSV oder XML lesen können.
- Das Dateiformat bestimmt die Art der Datenverarbeitung.
- Die gelesenen Daten werden in einer geeigneten Datenstruktur gespeichert.

- **Bewertung:**
  - 0.5 Punkte – Korrekte Definition der Klasse
  - 0.5 Punkte – Korrektes Einlesen der Datei

b) Anzeige von Ninjas mit Kraftpunkten über einem bestimmten Wert (0.5 Punkte)

- Der Benutzer gibt einen Schwellenwert für die Kraftpunkte ein.
- Es werden nur die **einzigartigen Namen** der Ninjas angezeigt, in der Reihenfolge, in der sie in der Datei erscheinen.
- Nur Ninjas mit höheren Kraftpunkten als der eingegebene Wert werden berücksichtigt.

- **Bewertung:**
  - 0.2 Punkte – Korrekte Filterung
  - 0.2 Punkte – Entfernen von Duplikaten
  - 0.1 Punkte – Korrekte Eingabe des Schwellenwerts

Beispiel input/output:
für **'5000'**

```
Gaara
Kankuro
Danzo Shimura
Neji Hyuga
Sasuke Uchiha
Deidara
Konan
Kankuro
Hinata Hyuga
```

c) Anzeige von Ereignissen der Stufe "Jonin" in absteigender Reihenfolge (0.5 Punkte)

- Es werden alle Ereignisse der Stufe Jonin in absteigender Reihenfolge nach Datum angezeigt.
- Die Stufe wird nicht über die Tastatur eingegeben.
- *Format:* `YYYY-MM-DD: Charaktername - Beschreibung`

- **Bewertung:**
  - 0.2 Punkte – Korrekte Filterung
  - 0.2 Punkte – Absteigende Sortierung
  - 0.1 Punkte – Korrekte Formatierung

Beispiel output:

```
2020-03-07: Minato Namikaze - Hat einen Spion enttarnt
2017-04-07: Naruto Uzumaki - Hat einen Verbündeten gerettet
2013-02-12: Gaara - Hat einen Verbündeten gerettet
2004-10-23: Deidara - Hat eine geheime Mission erfolgreich abgeschlossen
```

d) Schreiben der Gesamtzahl der Ereignisse pro Ninja-Stufe in eine Datei (2 Punkte)

- Die Ergebnisse werden in **gesammtzahl.txt** gespeichert.
- Jede Zeile enthält die Stufe und die Gesamtzahl der Ereignisse im Format:

  Stufe%AnzahlEreignisse#Gesamtpunkte
- Sortierung:
  - Absteigend nach Anzahl der Ereignisse.
  - Falls Gleichstand, aufsteigend nach Gesamtpunkten (Summe aller Kraftpunkte pro Stufe).

- **Bewertung:**
  - 0.5 Punkte – Korrektes Schreiben in die Datei
  - 0.5 Punkte – Korrekte Formatierung
  - 0.25 Punkte – Automatische Ermittlung der Stufen
  - 0.25 Punkte – Berechnung der Gesamtpunkte pro Stufe
  - 0.25 Punkte – Berechnung der Ereignisanzahl pro Stufe
  - 0.25 Punkte – Korrekte Sortierung

Beispiel output in gesammtzahl.txt:

```
Kage%10#34987.32
Genin%8#48379.22
Chunin%8#35715.47
Jonin%4#21116.12
```

Aufgabe 2: Shinobi-Handelsnetzwerk

Um den Handel in der Shinobi-Welt von Naruto zu verwalten, soll eine einfache, aber gut strukturierte Anwendung entwickelt werden.

a) Verwaltung von Produkten (1 Punkt)

Der Benutzer kann Produkte hinzufügen, bearbeiten, löschen und anzeigen.
Attribute: Name, Preis, Herkunftsregion.

b) Verwaltung von Charakteren (1 Punkt)

Der Benutzer kann Charaktere hinzufügen, bearbeiten, löschen und anzeigen.
Attribute: ID, Name, Herkunftsdorf, Liste der gekauften Produkte.

c) Filtern nach Herkunftsdorf (0.5 Punkte)

Der Benutzer kann sich nur die Charaktere eines bestimmten Dorfs anzeigen lassen.

Input:
```
Konoha
```

Output:
```
Naruto Uzumaki
Itachi Uchiha
```

d) Anzeigen von Charakteren, die Produkte aus einem bestimmten Dorf gekauft haben (0.5 Punkte)

Der Benutzer gibt eine Region ein. Die Anwendung zeigt alle Charaktere an, die Produkte aus dieser Region gekauft haben.
Sortierung: Alphabetisch nach Namen (aufsteigend)

Input:
```
Kirigakure
```

Output:
```
Deidara
Gaara
Itachi Uchiha
Kisame Hoshigaki
```

e) Sortieren der Produkte eines Charakters nach Preis (1 Punkt)

Der Benutzer wählt einen Charakter und gibt die Sortierart ein:

- Aufsteigend (günstigster zuerst)
- Absteigend (teuerster zuerst)

Input:
```
Charakter: Itachi Uchiha
Sortierreihenfolge:
Absteigend
```

Output:
```
Explosionskugel - 250.0
Schwert - 200.0
Schattendolch - 180.0
Giftklinge - 150.0
```

Hinweis: Die MVC-Architektur muss eingehalten werden und klar erkennbar sein.

Für die Punkte c, d und e werden die folgenden Daten verwendet, die auch im Dokument **"Aufgabe2_V1_Initialisierung.txt"** zu finden sind. Leichte Anpassungen des Codes zur eigenen Lösung sind erlaubt, jedoch nicht der Inhalt.

```
public static void main(String[] args) {
        // Initialisierung der Produkte
        List<Produkt> produkte = new ArrayList<>();
        produkte.add(new Produkt("Kunai", 50.0, "Konoha"));
        produkte.add(new Produkt("Shuriken", 30.0, "Konoha"));
        produkte.add(new Produkt("Schwert", 200.0, "Kirigakure"));
        produkte.add(new Produkt("Heiltrank", 100.0, "Sunagakure"));
        produkte.add(new Produkt("Sprengsiegel", 75.0, "Iwagakure"));
        produkte.add(new Produkt("Riesenfächer", 300.0, "Sunagakure"));
        produkte.add(new Produkt("Giftklinge", 150.0, "Kirigakure"));
        produkte.add(new Produkt("Explosionskugel", 250.0, "Iwagakure"));
        produkte.add(new Produkt("Schattendolch", 180.0, "Konoha"));
        produkte.add(new Produkt("Wasserperle", 90.0, "Kirigakure"));

        // Initialisierung der Charaktere
        List<Charakter> charaktere = new ArrayList<>();

        Charakter c1 = new Charakter(1, "Naruto Uzumaki", "Konoha");
        c1.kaufeProdukt(produkte.get(0)); // Kunai
        c1.kaufeProdukt(produkte.get(3)); // Heiltrank
        c1.kaufeProdukt(produkte.get(8)); // Schattendolch
        c1.kaufeProdukt(produkte.get(5)); // Riesenfächer

        Charakter c2 = new Charakter(2, "Gaara", "Sunagakure");
        c2.kaufeProdukt(produkte.get(2)); // Schwert
        c2.kaufeProdukt(produkte.get(4)); // Sprengsiegel
        c2.kaufeProdukt(produkte.get(6)); // Giftklinge
        c2.kaufeProdukt(produkte.get(1)); // Shuriken

        Charakter c3 = new Charakter(3, "Kisame Hoshigaki", "Kirigakure");
        c3.kaufeProdukt(produkte.get(1)); // Shuriken
        c3.kaufeProdukt(produkte.get(2)); // Schwert
        c3.kaufeProdukt(produkte.get(3)); // Heiltrank
        c3.kaufeProdukt(produkte.get(7)); // Explosionskugel
        c3.kaufeProdukt(produkte.get(9)); // Wasserperle

        Charakter c4 = new Charakter(4, "Deidara", "Iwagakure");
        c4.kaufeProdukt(produkte.get(0)); // Kunai
        c4.kaufeProdukt(produkte.get(4)); // Sprengsiegel
        c4.kaufeProdukt(produkte.get(7)); // Explosionskugel
        c4.kaufeProdukt(produkte.get(9)); // Wasserperle

        Charakter c5 = new Charakter(5, "Itachi Uchiha", "Konoha");
        c5.kaufeProdukt(produkte.get(8)); // Schattendolch
```

```
        c5.kaufeProdukt(produkte.get(6)); // Giftklinge
        c5.kaufeProdukt(produkte.get(2)); // Schwert
        c5.kaufeProdukt(produkte.get(7)); // Explosionskugel

        charaktere.add(c1);
        charaktere.add(c2);
        charaktere.add(c3);
        charaktere.add(c4);
        charaktere.add(c5);
    }
```

Hinweis: 1 Punkt wird für Code Qualität ausgegeben (Java Conventions, MVC- oder Schichtenarchitektur, Javadocs, lesbarer Code, etc.)

Benotung:
A1: 4p
A2: 4p
Code Qualiät: 1p
1 Punkt von Amtswegen