Allgemeines

Wir freuen uns sehr, dass du dich dazu entschieden hast, dich bei uns zu bewerben!

Die Aufgaben sollst du alleine lösen, selbstverständlich darfst du Internet, Literatur usw. als Hilfe nutzen - uns geht es darum, dass die Lösung von DIR kommt.
Bitte löse die Aufgaben mit Java.

Bewertungskriterien

Wie bereits erwähnt, ist es uns wichtig dass die Lösung von dir kommt.
Wir bewerten nicht ausschließlich, dass dein Code am Ende funktioniert, sondern interessieren uns mehr für dein Endergebnis und deinen Weg dort hin.

Das heißt ein professionelles Gesamtpaket ist uns wichtig, damit meinen wir folgende Faktoren:

- Struktur des Projekts
- Code Qualität
  - Lesbarkeit des Codes
  - Kommentare
  - Struktur des Codes
  - Nutzung von Patterns (wenn sinnvoll)
  - Einhaltung von Standards
- Nutzung aktueller Features des JDKs
- Sinnvoller Einsatz von Frameworks/ Dependencies
- Unit Tests
- Nutzung eines Versionskontrollsystems (vorzugsweise Git)
- Nutzung von Maven

Hinweis: Solltest du eine Library finden, die Teile einer Aufgabe oder sogar die ganze Aufgabe lösen, wirkt sich das keinesfalls negativ auf die Bewertung aus.

Aufgabe 1

Grundlage sind zwei Textdateien (`List1.txt` , `List2.txt`) mit jeweils einem Namen (Vor- und Nachname) pro Zeile

Schreibe eine Anwendung welche die folgenden Daten ermittelt:

- Welche Namen kommen nur in `List1.txt` vor?
- Welche nur in `List2.txt`
- Welche in beiden?

Output als JSON auf der Console (gerne `System.out`)

Beispiel:
Für die Listen
Liste1: `[word1, word2, word5, word6]`
Liste2: `[word5, word6,word3,word4]`
erwarten wir folgende Ausgabe

```
{
  "onlyInList1": [
    "word1",
    "word2"
  ],
  "onlyInList2": [
    "word3",
    "word4"
  ],
  "inBothLists": [
    "word5",
    "word6"
  ]
}
```

Aufgabe 2

Es soll eine Web Anwendung entwickelt werden, die den Endpoint `/text/analyze` enthält. Dieser Endpoint nimmt einen JSON Body entgegen, der verschiedene Metadaten über Produkte enthält. Ein Beispiel Body hierfür befindet sich in der Datei `sampleProductsData.json`.
In der Response sollten sich folgende Informationen befinden:

- Teuerstes Produkt
- Billigstes Produkt
- Am meisten verkauftes Produkt
- Alle Namen der Deutschen Produkte
- Alle Namen der Chinesischen Produkte

- Ob die Datei ein oder mehrere fragile Produkte enthält

Bro tip: Das Attribut "countryOfOrigin" nutzt den `ISO-3166-2` Standard. Die Komplette Liste findest du hier

Beispiel
Für die Datei

```
[
  {
    "name": "product1",
    "countryOfOrigin": "DE",
    "price": 123.50,
    "isFragile": true,
    "timesPurchased": 150
  },
  {
    "name": "product2",
    "countryOfOrigin": "DE",
    "price": 45,
    "isFragile": false,
    "timesPurchased": 1241
  },
  {
    "name": "product3",
    "countryOfOrigin": "ES",
    "price": 13,
    "isFragile": false,
    "timesPurchased": 772
  }
]
```

```
{
  "mostExpensiveProduct": "product1",
  "cheapestProduct": "product3",
  "mostPopularProduct": "product2",
  "germanProducts": [
    "product1",
    "product2"
  ],
  "chineseProducts": [],
  "containsFragileProducts": true
}
```

Aufgabe 3

Auf einem Flohmarkt verkauft der findige Floh Händler Florian Flöhe.
Alle angebotenen Flöhe wurden von einem Expertengremium in eine Skala von 1 bis 10 eingeordnet (wobei 10 am besten ist).

Ergänze die folgende Funktion so, dass sie mit einem zur Verfügung gestellten Betrag (`money`) die Flöhe mit der aufsummiert höchsten Wertung errechnet.

Hinweis: Die Aufgabe kann nur mit Brute Force Ansatz erfolgreich (also zu 100%) gelöst werden ;)

Vorgaben

```
public static int getOptimalValue(float money, List<Flear> flears){
    int res = 0;

    // ...
    // TODO implement me
    // ...

    return res;
}
```

Item sieht aus wie folgt:

```java
package de.arvato.application;

public class Flear {

 String name;

 float price;

 int rating;

 public Flear() {}

public Flear(final String name, final float price, final int rating){
    this.name=name;
    this.price=price;
    this.rating=rating;
}

 public String getName() {
  return this.name;
 }

 public void setName(String value) {
  this.name = value;
 }

 public float getPrice() {
  return this.price;
 }

 public void setPrice(float value) {
  this.price = value;
 }

 public int getRating() {
  return this.rating;
 }

 public void setRating(int value) {
  this.rating = value;
 }

 public boolean equals(Object object) {
  if (this == object) return true;
  if (object == null || getClass() != object.getClass()) return false;
  if (!super.equals(object)) return false;
  Item item = (Item) object;
  return java.util.Objects.equals(name, item.name);
 }
```

```
  public int hashCode() {
   return Objects.hash(super.hashCode(), name);
  }
 }
```

Abgabe

Cool!
Wir freuen uns auf dein Werk ;D

Übermittlung der Lösung

Du kannst deine Lösung in einem Repository deiner Wahl ablegen und uns am Ende darauf zugreifen lassen, sowohl beispielsweise Bitbucket (kostenlose private Git Repositories) als auch GitHub sind kein Problem.

Du kannst uns deine Lösung auch als .zip file zukommen lassen