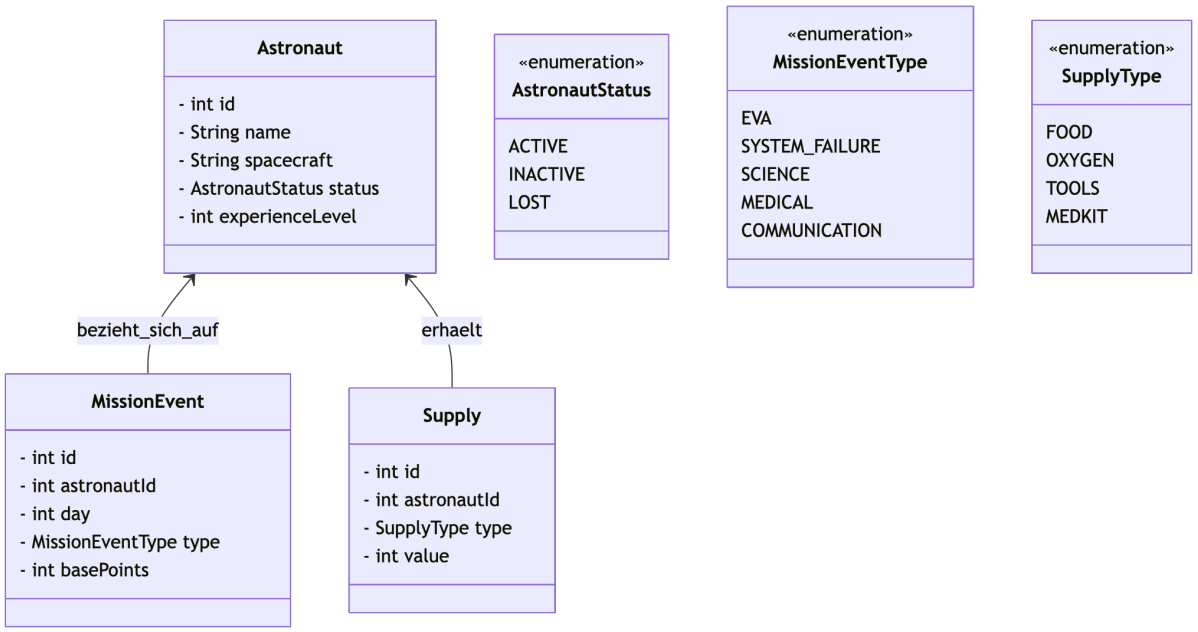
Fortgeschrittene Programmierungsmethoden
Praktische Nachprüfung
V4

Space Mission - Mission Control Log

Während einer Raumfahrtmission verwaltet das System "Mission Control" alle Astronauten, deren Ereignisse während der Mission sowie die von der Bodenstation erhaltenen Supplies. Auf Grundlage dieser Informationen werden verschiedene Listen und Berichte erstellt, welche den Verlauf der Mission beeinflussen.
Modellieren Sie in Java das folgende Klassendiagramm und verwenden Sie es in Ihrer Implementierung:



---

1. ***(1 Punkt)*** Lesen Sie die Daten aus den JSON-Dateien **astronauts.json**, **events.json** und **supplies.json** und speichern Sie diese in Java-Listen. Geben Sie anschließend auf der Konsole aus:
   - Anzahl der Astronauten
   - Anzahl der Ereignisse
   - Anzahl der Supplies
   - alle Astronauten (jeweils eine Zeile) in der Reihenfolge aus der JSON-Datei

Ausgabeformat (Astronaut):
`[#id] name | spacecraft | status | exp=<experienceLevel>`

Ausgabe:

```
Astronauts loaded: 15
Events loaded: 25
Supplies loaded: 14

[#1] Ava Ionescu | Orion | ACTIVE | exp=9
[#2] Mihai Petrescu | Dragon | ACTIVE | exp=8
[#3] Elena Pop | Orion | INACTIVE | exp=7
[#4] Radu Stan | Starliner | ACTIVE | exp=6
[#5] Sofia Marin | Dragon | ACTIVE | exp=7
[#6] Andrei Dumitru | Orion | ACTIVE | exp=5
[#7] Clara Neagu | Starliner | INACTIVE | exp=8
[#8] Victor Iliescu | Orion | ACTIVE | exp=6
[#9] Ioana Rusu | Dragon | LOST | exp=9
[#10] Daniel Voicu | Starliner | ACTIVE | exp=7
[#11] Bianca Tudor | Orion | ACTIVE | exp=8
[#12] Horia Pascu | Dragon | INACTIVE | exp=6
[#13] Nadia Ene | Starliner | ACTIVE | exp=5
[#14] Teodor Matei | Orion | ACTIVE | exp=7
[#15] Larisa Dobre | Dragon | ACTIVE | exp=6
```

---

2. ***(0.5 Punkte)*** Filtern nach spacecraft und Status
   Lesen Sie von der Tastatur einen String **spacecraft**. Geben Sie anschließend **nur** die Astronauten aus, welche folgende Bedingungen erfüllen:
   - `spacecraft == input`
   - `status == ACTIVE`

   Die Ausgabe erfolgt im selben Format wie in Aufgabe 1.

Ausgabe:

```
Input Spacecraft: Orion

[#1] Ava Ionescu | Orion | ACTIVE | exp=9
[#6] Andrei Dumitru | Orion | ACTIVE | exp=5
[#8] Victor Iliescu | Orion | ACTIVE | exp=6
[#11] Bianca Tudor | Orion | ACTIVE | exp=8
[#14] Teodor Matei | Orion | ACTIVE | exp=7
```

---

3. ***(0.5 Punkte)*** Sortierung der Astronauten
   Sortieren Sie die Liste aller Astronauten nach folgenden Kriterien:
   - zuerst nach **experienceLevel absteigend**
   - bei gleichem experienceLevel nach **name aufsteigend**

   Geben Sie anschließend die sortierte Liste auf der Konsole aus.

Ausgabe:

```
[#1] Ava Ionescu | Orion | ACTIVE | exp=9
[#9] Ioana Rusu | Dragon | LOST | exp=9
[#11] Bianca Tudor | Orion | ACTIVE | exp=8
[#7] Clara Neagu | Starliner | INACTIVE | exp=8
[#2] Mihai Petrescu | Dragon | ACTIVE | exp=8
[#10] Daniel Voicu | Starliner | ACTIVE | exp=7
[#3] Elena Pop | Orion | INACTIVE | exp=7
[#5] Sofia Marin | Dragon | ACTIVE | exp=7
[#14] Teodor Matei | Orion | ACTIVE | exp=7
[#12] Horia Pascu | Dragon | INACTIVE | exp=6
[#15] Larisa Dobre | Dragon | ACTIVE | exp=6
[#4] Radu Stan | Starliner | ACTIVE | exp=6
[#8] Victor Iliescu | Orion | ACTIVE | exp=6
[#6] Andrei Dumitru | Orion | ACTIVE | exp=5
[#13] Nadia Ene | Starliner | ACTIVE | exp=5
```

---

4. ***(1 Punkt)*** Schreiben in eine Datei
   Schreiben Sie die in Aufgabe 3 sortierte Astronauten Liste in **umgekehrter Reihenfolge** in die Datei **astronauts_sorted.txt**. Jeder Astronaut soll in einer eigenen Zeile gespeichert werden, im selben Format wie bei der Konsolenausgabe.

---

5. ***(1.5 Punkte)*** Punktberechnung
   Implementieren Sie die Berechnung der **computedPoints** für jedes `MissionEvent` gemäß den folgenden Regeln:
   - **EVA** → `computedPoints = basePoints + 2 * day`
   - **SYSTEM_FAILURE** → `computedPoints = basePoints - 3 - day`
   - **SCIENCE** → `computedPoints = basePoints + (day % 4)`
   - **MEDICAL** → `computedPoints = basePoints - 2 * (day % 3)`
   - **COMMUNICATION** → `computedPoints = basePoints + 5`

   Geben Sie anschließend die **ersten 5 Events** aus `events.json` auf der Konsole aus.

Ausgabeformat:

Event <id> -> raw=<basePoints> -> computed=<computedPoints>

Ausgabe:

Event 1 -> raw=4 -> computed=9
Event 2 -> raw=7 -> computed=8
Event 3 -> raw=5 -> computed=9
Event 4 -> raw=3 -> computed=-2
Event 5 -> raw=-2 -> computed=-2

---

6. ***(1.5 Punkte)*** Ranking

   Berechnen Sie für **jeden** Astronauten den Gesamtwert totalScore nach folgender Formel:

totalScore = Sum(computedPoints aus allen MissionEvents des Astronauten) + Sum(value aus allen Supplies des Astronauten)

Anforderungen:

- Berechnen Sie totalScore für alle Astronauten.
- Geben Sie die **Top 5** auf der Konsole aus, sortiert nach:
  - totalScore **absteigend**
  - bei Gleichstand name **aufsteigend**
- Bestimmen und geben Sie zusätzlich das **Leading spacecraft** aus (= spacecraft des Astronauten auf Platz 1).

Ausgabeformat:

Top 5 Astronauts:
1. <Name> (<Spacecraft>) -> <totalScore>
...
Leading spacecraft: <Spacecraft>

Ausgabe:

Top 5 Astronauts:
1. Bianca Tudor (Orion) -> 70
2. Ava Ionescu (Orion) -> 51
3. Mihai Petrescu (Dragon) -> 47
4. Larisa Dobre (Dragon) -> 39
5. Radu Stan (Starliner) -> 25

Leading spacecraft: Orion

---

7. ***(1 Punkt)*** Abschlussbericht
   Erstellen Sie die Datei **mission_report.txt**, welche die Anzahl der **MissionEvents** pro MissionEventType (basierend auf events.json) enthält.
   Sortierung:
   - zuerst nach **Anzahl absteigend**
   - bei Gleichstand nach **Name aufsteigend**

   Ausgabeformat:
   <TYPE> -> <ANZAHL>

Ausgabe:

Inhalt der Datei *mission_report.txt*
EVA -> 6
SCIENCE -> 6
SYSTEM_FAILURE -> 6
COMMUNICATION -> 4
MEDICAL -> 3

- **2 Punkte** für **Codequalität**
  (Einhaltung der Java-Konventionen, klare Trennung in Schichten / MVC, strukturierter und gut lesbarer Code)
- **1 Punkt** wird **von Amts wegen** vergeben