

# Wiederholklausur

Modul	Programmierung II
Lehreinheit	Fortgeschrittene Programmierung
Dozent	Daniel Appenmaier
Kurse	WWIBE121 und WWIBE221
Matrikelnummer	

Aufgabe	Max. Punkte	Punkte
1	12	
2	14	
3	15	
4	10	

*Viel Erfolg!*

## Aufgabe 1 (12 Punkte)

Bearbeite eine der beiden folgenden Optionen:

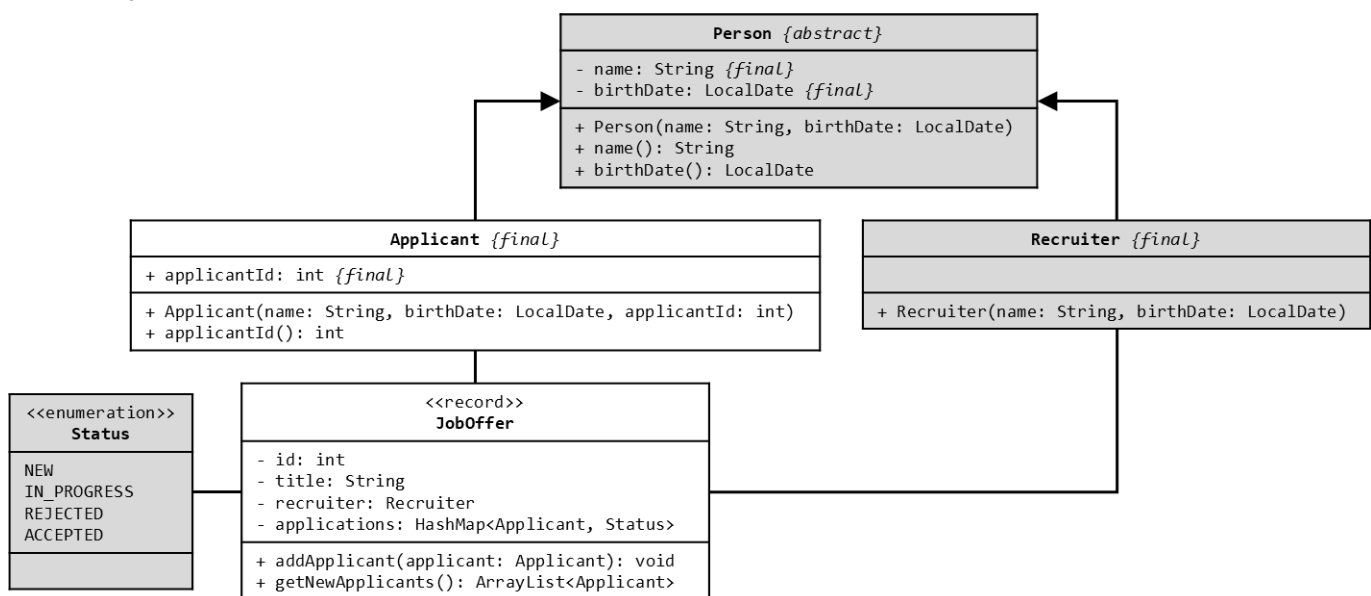
### Option 1

- Erläutere kurz, zu was sich eine Klasse verpflichtet, die eine **Schnittstelle** implementiert (2 Punkte)
- Erläutere kurz, was man unter **Kompilierungsfehler**, **Logikfehler** und **Laufzeitfehlern** versteht (3 Punkte)
- Erläutere kurz den wesentlichen Unterschied zwischen **lokalen und anonymen Klassen** (2 Punkte)
- Benenne die Methoden, die für den Zugriff auf **die Einträge, die Schlüssel sowie die Werte eines Assoziativspeichers** verwendet werden können (3 Punkte)
- Erläutere kurz die **Red-Green-Refactor-Methode** (2 Punkte)

### Option 2

Erstelle die Klassen **Applicant** (5 Punkte) und **JobOffer** (7 Punkte) anhand des abgebildeten Klassendiagramms.

Klassendiagramm



Allgemeiner Hinweis zum Klassendiagramm

Der Stereotyp **record** impliziert, dass die Datenklasse einen entsprechenden Konstruktor, Getter zu allen Attributen sowie entsprechende Implementierungen für die Object-Methoden besitzt.

Hinweise zur Klasse **Applicant**

- Der Konstruktor soll alle Attribute mit den eingehenden Werten initialisieren
- Die Methode `int applicantId()` soll den Wert des Attributs `applicantId` zurückgeben

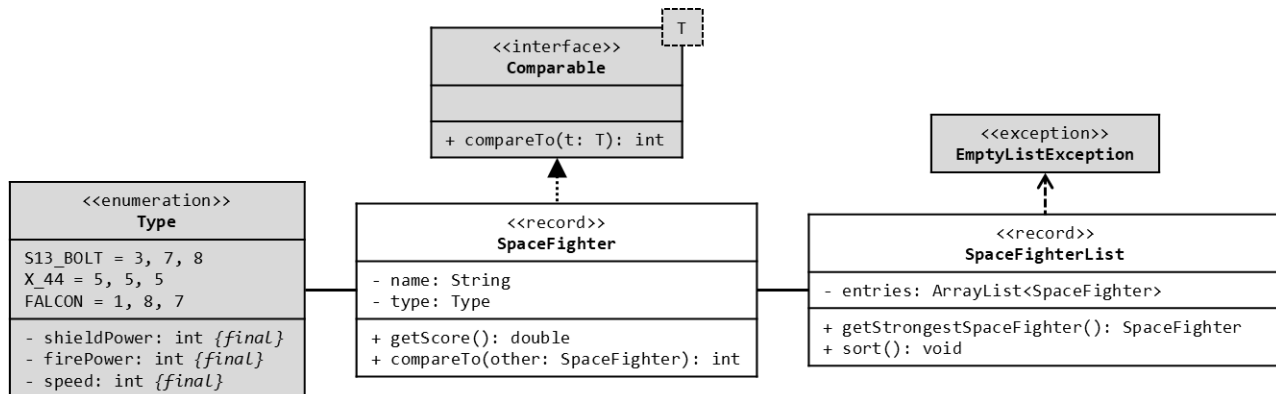
Hinweise zur Klasse **JobOffer**

- Die Methode `void addApplicant(applicant: Applicant)` soll der Bewerberliste (`applications`) den eingehenden Bewerber mit dem Status `NEW` hinzufügen
- Die Methode `ArrayList<Applicant> getNewApplicants()` soll alle Bewerber mit dem Status `NEW` zurückgeben

## Aufgabe 2 (14 Punkte)

Erstelle die Klassen **SpaceFighter** (5 Punkte) und **SpaceFighterList** (9 Punkte) anhand des abgebildeten Klassendiagramms.

## Klassendiagramm



## Allgemeine Hinweise zum Klassendiagramm

- Der Stereotyp **record** impliziert, dass die Datenklasse einen entsprechenden Konstruktor, Getter zu allen Attributen sowie entsprechende Implementierungen für die Object-Methoden besitzt
- Der Stereotyp **enumeration** impliziert, dass die Aufzählung einen passenden, privaten Konstruktor sowie ggbf. passende Setter und Getter besitzt

Hinweise zur Klasse **SpaceFighter**

- Die Methode **double** `getScore()` soll die Punktzahl des Sternenjäger als Durchschnittswert aus Schildkraft (**shieldPower**), Feuerkraft (**firePower**) und Geschwindigkeit (**speed**) zurückgeben
- Die Methode **int** `compareTo(other: SpaceFighter)` soll so implementiert werden, dass damit Sternenjäger anhand der Punktzahl der Sternenjäger absteigend sortiert werden können

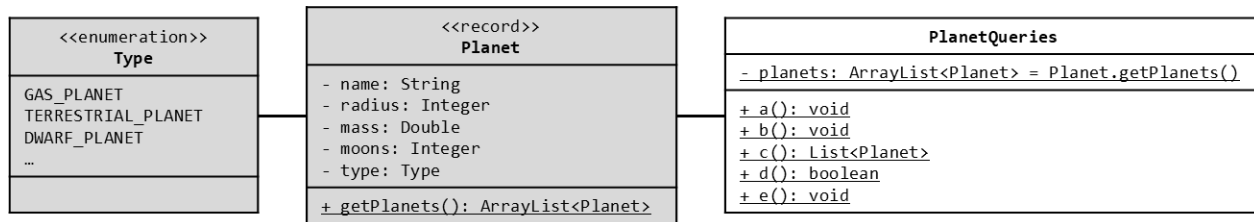
Hinweise zur Klasse **SpaceFighterList**

- Die Methode **SpaceFighter** `getStrongestSpaceFighter()` soll den Sternenjäger mit der größten Feuerkraft (**firePower**) zurückgeben
- Die Methode **void** `sort()` soll die Einträge der Sternenjägerliste (**entries**) absteigend nach der Punktzahl sortieren. Für den Fall, dass die Sternenjägerliste keine Einträge enthält, soll die Ausnahme **EmptyListException** ausgelöst werden

## Aufgabe 3 (15 Punkte)

Erstelle die Klasse **PlanetQueries** anhand des abgebildeten Klassendiagramms.

## Klassendiagramm



## Allgemeiner Hinweis zum Klassendiagramm

Der Stereotyp **record** impliziert, dass die Datenklasse einen entsprechenden Konstruktor, Getter zu allen Attributen sowie entsprechende Implementierungen für die Object-Methoden besitzt.

Hinweise zur Klasse **PlanetQueries**

- Die statische Methode **void a()** soll alle terrestrischen Planeten mit einer Masse von höchstens  $1,0 \times 10^{22} \text{ kg}$  in der Form *[Name]: [Masse]x10e22kg* ausgeben
- Die statische Methode **void b()** soll zunächst die durchschnittliche Anzahl Monde aller Planeten ermitteln und diese anschließend (falls vorhanden) in der Form *Durchschnittliche Anzahl Monde: [Durchschnittliche Anzahl Monde]* ausgeben
- Die statische Methode **List<Planet> c()** soll alle Planeten mit mindesten einem Mond absteigend nach dem Radius sortiert als Liste zurückgeben
- Die statische Methode **boolean d()** soll zurückgeben, ob es einen Planeten mit dem Namen *Erde* gibt
- Die statische Methode **void e()** soll zunächst alle Planeten gruppiert nach der Anzahl Monde ermitteln und diese anschließend in der Form *[Anzahl Monde]: [[Planet], [Planet],...], [Anzahl Monde]: [[Planet], [Planet],...],...* ausgeben

## Beispielhafte Konsolenausgabe

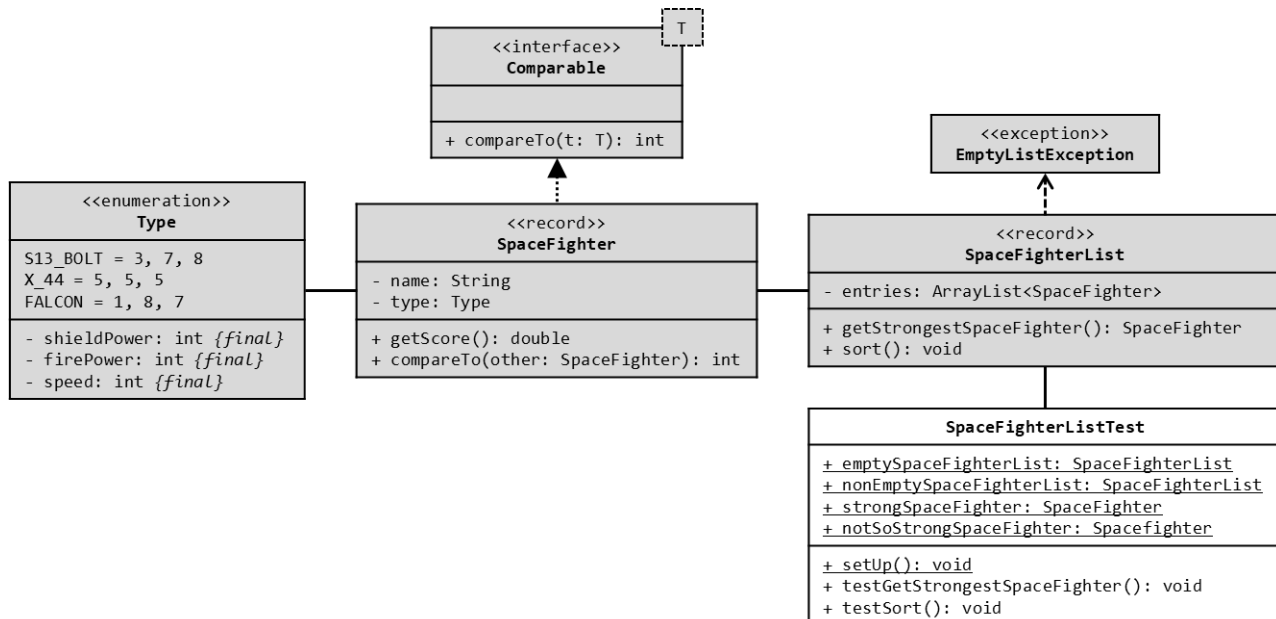
```

a(): Merkur: 0.33x10e22kg, Mars: 0.64x10e22kg
b(): Durchschnittliche Anzahl Monde: 25.625
e(): 0: [Planet[name=Merkur, radius=2439,...],...], 1: [Planet[name=Erde, radius=6378,...]],...
  
```

## Aufgabe 4 (10 Punkte)

Erstelle die JUnit-5-Testklasse **SpaceFighterListTest** anhand des abgebildeten Klassendiagramms.

## Klassendiagramm



## Allgemeine Hinweise zum Klassendiagramm

- Der Stereotyp **record** impliziert, dass die Datenklasse einen entsprechenden Konstruktor, Getter zu allen Attributen sowie entsprechende Implementierungen für die Object-Methoden besitzt
- Der Stereotyp **enumeration** impliziert, dass die Aufzählung einen privaten Konstruktor sowie ggbs. passende Setter und Getter besitzt

Hinweise zur Testklasse **SpaceFighterListTest**

- Die statische Lebenszyklus-Methode **void setUp()** soll das nachfolgende Testszenario aufbauen:
  - Es soll eine Sternenjägerliste erstellt und dem Attribut **emptySpaceFighterList** zugewiesen werden
  - Es soll ein Sternenjäger mit der Kennung **SF-1 (id)** und dem Typ **S13\_BOLT (type)** als Wert erstellt und dem Attribut **strongSpaceFighter** zugewiesen werden
  - Es soll ein Sternenjäger mit der Kennung **SF-2 (id)** und dem Typ **X\_44 (type)** als Wert erstellt und dem Attribut **notSoStrongSpaceFighter** zugewiesen werden
  - Es soll eine Sternenjägerliste mit den beiden erstellten Sternenjägern als Einträge erstellt und dem Attribut **nonEmptySpaceFighterList** zugewiesen werden
- Die Testmethode **testGetStrongestSpaceFighter()** soll den nachfolgenden Testfall abdecken: Beim Aufruf der Methode **SpaceFighter getStrongestSpaceFighter()** auf das Attribut **nonEmptySpaceFighterList** wird als Rückgabewert der Sternenjäger mit der Kennung (**id**) **SF-1** und dem Typ (**type**) **S13\_BOLT** erwartet
- Die Testmethode **void testSort()** soll den nachfolgenden Testfall abdecken: Beim Aufruf der Methode **void sort()** auf das Attribut **emptySpaceFighterList** wird die Ausnahme **EmptyListException** erwartet