

Klausur

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	11	

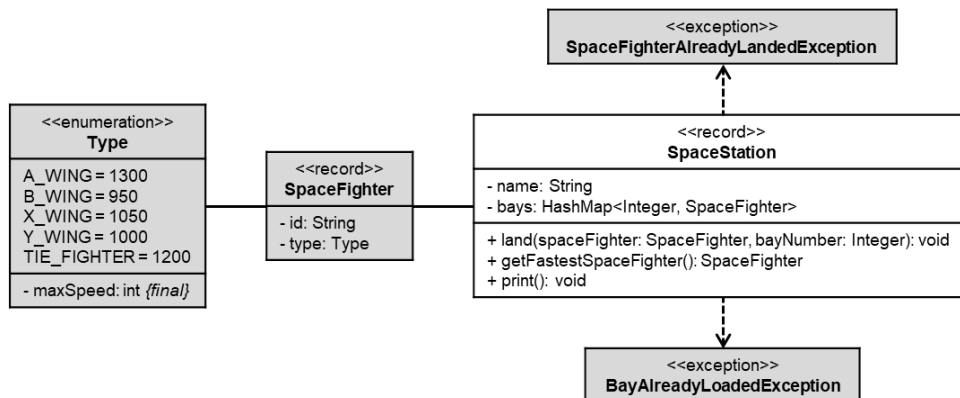
Aufgabe 1 (12 Punkte)

- a) Erläutere kurz, was man unter einer **abstrakten Klasse** versteht (2 Punkte)
- b) Erläutere kurz den wesentlichen Unterschied der Schnittstellen **Comparable<T>** und **Comparator<T>** (3 Punkte)
- c) Erläutere kurz, was man unter der **Catch-or-Throw-Regel** versteht (2 Punkte)
- d) Benenne den wesentlichen Vorteil bei der **generischen Programmierung mit Java Generics** im Vergleich zur **generischen Programmierung ohne Java Generics** (1 Punkt)
- e) Erläutere kurz, was man unter dem Begriff **Bedarfsauswertung** (Lazy Evaluation) versteht (2 Punkte)
- f) Skizziere die **Testpyramide** (2 Punkte)

Aufgabe 2 (14 Punkte)

Erstelle die Klasse **SpaceStation** anhand des abgebildeten Klassendiagramms sowie der abgebildeten Konsolenausgabe.

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 Klasse **SpaceStation**

- Die Methode **void land(spaceFighter: SpaceFighter, bayNumber: Integer)** soll den eingehenden Sternenjäger in der Bucht mit der eingehenden Buchtnummer landen lassen. Für den Fall, dass der eingehende Sternenjäger bereits gelandet ist (also bereits eine Bucht belegt), soll die Ausnahme **SpaceFighterAlreadyLandedException** ausgelöst werden und für den Fall, dass die Bucht bereits belegt ist, die Ausnahme **BayAlreadyLoadedException**
- Die Methode **SpaceFighter getFastestSpaceFighter()** soll den schnellsten Sternenjäger der Raumstation zurückgeben
- Die Methode **void print()** soll alle Attribute der Raumstation wie abgebildet ausgeben

Beispielhafte Konsolenausgabe

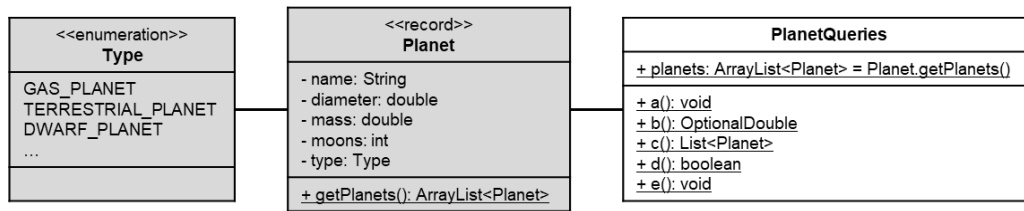
```
SpaceStation X

Bucht 1: SpaceFighter[id="XD-54", type=TIE_FIGHTER]
Bucht 2: frei
Bucht 3: SpaceFighter[id="Red Bolt", type=A_WING]
Bucht 4: SpaceFighter[id="XD-98", type=TIE_FIGHTER]
Bucht 5: frei
Bucht 6: SpaceFighter[id="Luke's X-Wing", type=X_WING]
```

Aufgabe 3 (15 Punkte)

Erstelle die Klasse **PlanetQueries** 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 Klasse **PlanetQueries**

- Die statische Methode **void a()** soll alle Planeten mit mehr als 5 Monden in der Form *[Name]: [Anzahl Monde]* ausgeben
- Die statische Methode **OptionalDouble b()** soll den durchschnittlichen Durchmesser aller Gasplaneten zurückgeben
- Die statische Methode **List<Planet> c()** soll alle Planeten als Liste absteigend sortiert nach der Masse zurückgeben
- Die statische Methode **boolean d()** soll zurückgeben, ob alle Planeten mindestens einen Mond besitzen
- Die statische Methode **void e()** soll alle Planeten gruppiert nach ihrem Typ in der Form *[Typ]: [Planet], [Planet], ..., [Typ]: [Planet], [Planet], ..., ...* ausgeben

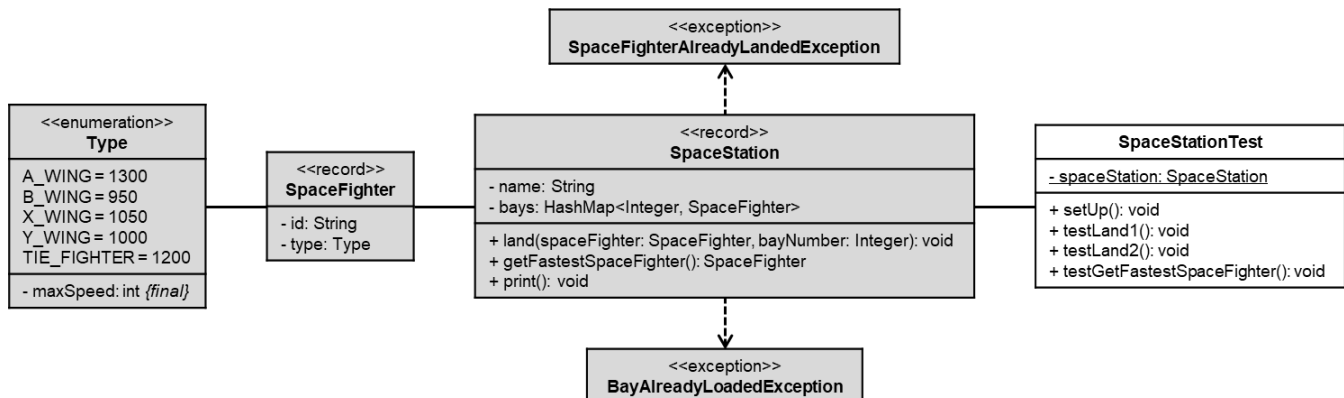
Beispielhafte Konsolenausgabe

```
a(): Erde: 1
b(): GAS_PLANET: [Planet[name=Saturn,...], Planet[name=Jupiter,...], DWARF_PLANET: [Planet[name=Pluto,...]]
```

Aufgabe 4 (11 Punkte)

Erstelle die Testklasse **SpaceStationTest** 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 **SpaceStationTest**

- Die Lebenszyklus-Methode **void setUp()** soll das nachfolgende Testszenario aufbauen:
 - Es soll eine Raumstation (**spaceStation**) mit dem Namen *SpaceStation X* sowie einem Assoziativspeicher erstellt werden
 - Dem Assoziativspeicher soll ein Eintrag mit der Buchtnummer *1* als Schlüssel und dem Sternenjäger mit der Kennung (**id**) *SF-1* und dem Typ (**type**) *A_WING* als Wert hinzugefügt werden
 - Dem Assoziativspeicher soll ein Eintrag mit der Buchtnummer *2* als Schlüssel und dem Sternenjäger mit der Kennung (**id**) *SF-2* und dem Typ (**type**) *X_WING* als Wert hinzugefügt werden
 - Dem Assoziativspeicher soll ein Eintrag mit der Buchtnummer *3* sowie keinem Wert hinzugefügt werden
- Die Testmethode **void testLand1()** soll den nachfolgenden Testfall abdecken: Beim Aufruf der Methode **void land(spaceFighter: SpaceFighter, bayNumber: Integer)** mit einer Buchtnummer, die bereits belegt ist, wird die Ausnahme **BayAlreadyLoadedException** erwartet
- Die Testmethode **void testLand2()** soll den nachfolgenden Testfall abdecken: Beim Aufruf der Methode **void land(spaceFighter: SpaceFighter, bayNumber: Integer)** mit einem Sternenjäger, der bereits gelandet ist, wird die Ausnahme **SpaceFighterAlreadyLandedException** erwartet
- Die Testmethode **testGetFastestSpaceFighter()** soll den nachfolgenden Testfall abdecken: Beim Aufruf der Methode **SpaceFighter getFastestSpaceFighter()** wird als Rückgabewert der Sternenjäger mit der Kennung (**id**) *SF-1* und dem Typ (**type**) *A_WING* erwartet