**Universität Leipzig**
**Institut für Informatik**

**WS 17/18**

# Modellierung und Programmierung 1 – Übungsblatt 2

Abgabetermin: 14.11.2017, 23:55 Uhr
Abgabeformat: 1 PDF Dokument & 1 ZIP-Archiv mit Java Dateien
Max. Punkte: 32

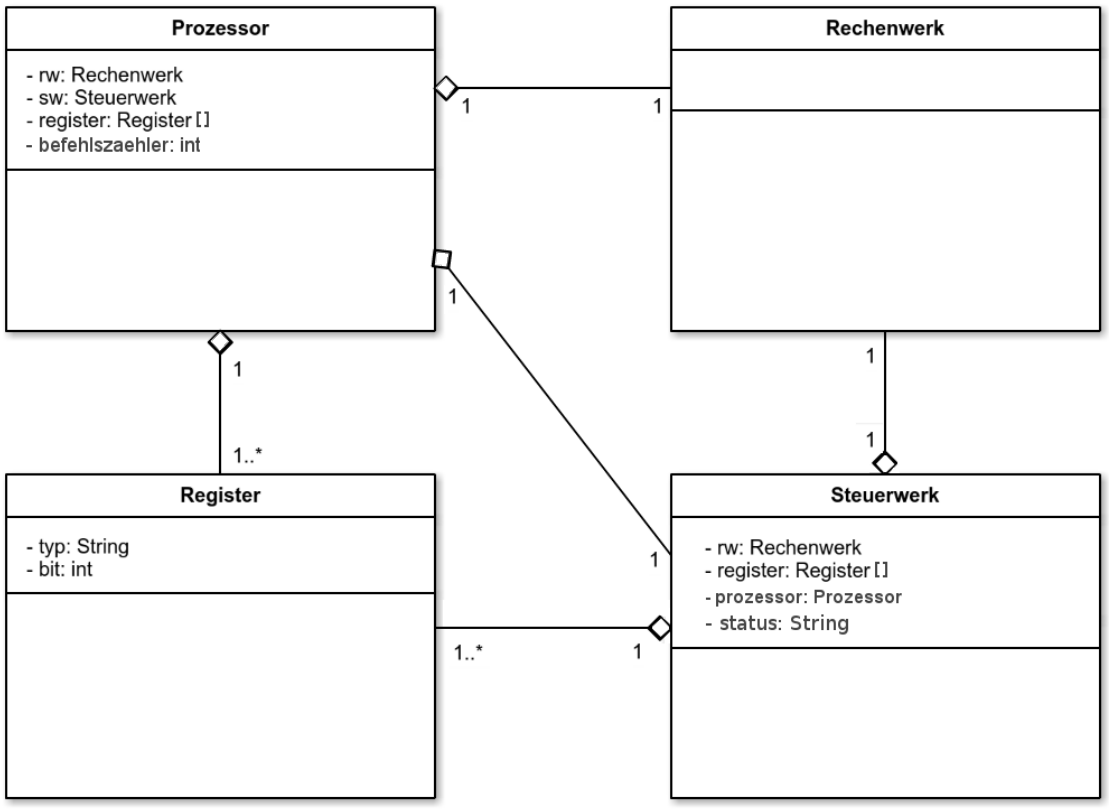
---

1. *(8 Punkte)* UML Klassendiagramm mit Funktionalitäten: Prozessor

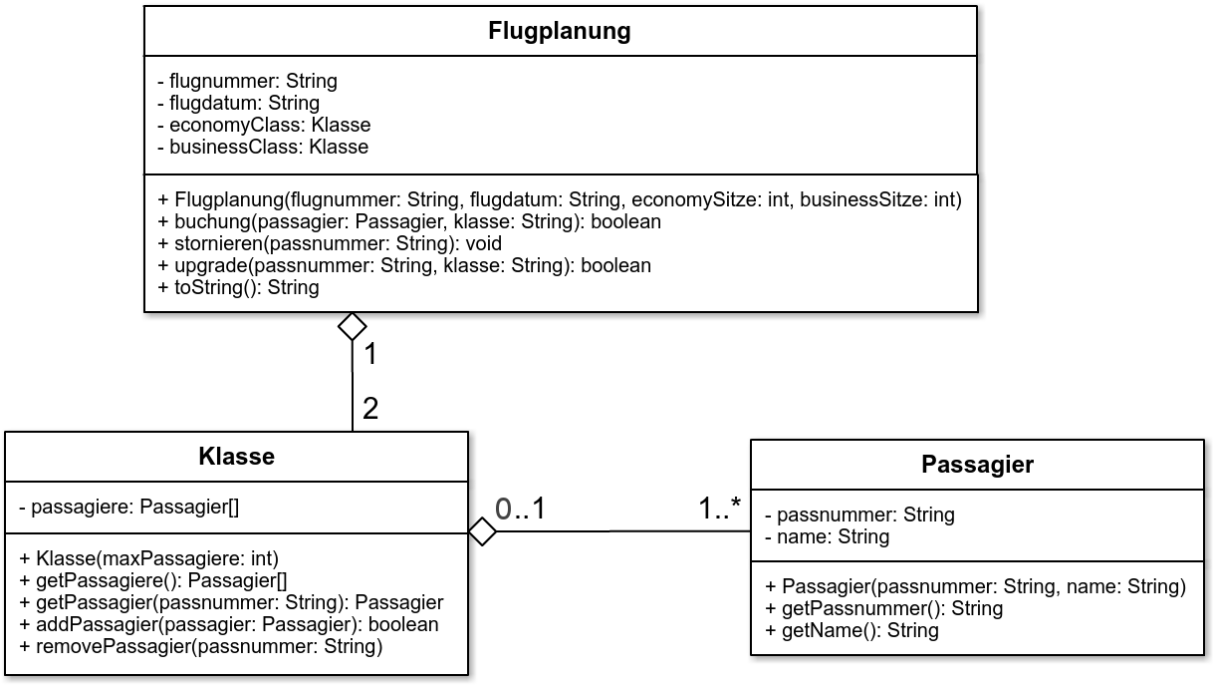
   Das Steuerwerk führt Befehle aus (wir nehmen an, es gibt eine Klasse *Befehl*, die alle notwendigen Informationen zum Ausführen eines Befehls enthält) und gibt eine Information zurück, ob ein Befehl erfolgreich abgearbeitet wurde. Dabei wird der Befehlszähler des Prozessors erhöht. Das Rechenwerk führt vier verschiedene Operationen durch, die arithmetischen Operationen Addition und Multiplikation sowie die logischen Operationen AND und OR. Dabei wird für jeweils zwei übergebene Parameter das Ergebnis entsprechend der durchgeführten Operation geliefert. Der dafür notwendige Speicher wird in den Registern verwaltet, welche Speicher einer bestimmten Größe belegen sowie nicht genutzten Speicher nach Beendigung einer Operation wieder freigeben können. Bei Speicherfreigabe wird der Wert des freien Speichers zurückgegeben, bei Speicherbelegung eine Information darüber, ob die Belegung erfolgreich war. Der aktuelle Status wird über das Steuerwerk ausgeben.

   *(a)* Erweitern Sie das unten gegebene UML Klassendiagramm um die im Text genannten Funktionalitäten. Achten Sie dabei auf korrekte Syntax für Übergabeparameter und Rückgabewerte!

   *(b)* Geben Sie im Diagramm weiterhin für jede Klasse einen Konstrukor an! Der Prozessor wird mit Rechen- und Steuerwerk sowie einem Array von Registern initialisiert. Das Steuerwerk erhält den Prozessor als Parameter. Ein Register wird mit seinem Typ sowie der entsprechenden Speichergröße initialisiert.



2. *(24 Punkte)* Gegeben sei folgendes UML Klassendiagramm für eine Flugplanung:




Erstellen sie gemäß des Diagramms ein Gerüst aus 3 Java-Klassen *(4 Punkte)*.

Achten sie dabei auf:

- korrekte Implementierung der Instanzvariablen und abgebildeten Beziehungstypen
- auftretende Multiplizitäten bei der Initialisierung von Arrays
- korrekte Implementierung der Sichtbarkeiten von Instanzvariablen und Funktionen
- Erstellung je eines Konstruktors pro Klasse

*Hinweis:* Die im `Flugplanung`-Konstrukor gegebenen Parameter `economySitze` und `businessSitze` werden zur Initialisierung der jeweiligen `Klasse` übergeben!

Zudem sind folgende Funktionen zu implementieren:

**Flugplanung.java** *(8 Punkte)*

- *buchung*: Ein Passagier soll dem Flug mit der entsprechenden Klasse ("economy" oder "business") hinzugefügt werden. Dabei soll sichergestellt werden, dass ein Passagier nicht mehrfach buchen kann. Weiterhin soll eine Information erscheinen, wenn eine Buchung nicht möglich ist. Das tritt dann ein, wenn alle Plätze der gewählten Kategorie belegt sind.
- *stornieren*: Ein gebuchter Flug wird aus der zugehörigen Klasse entfernt.
- *upgrade*: Die Klasse des Passagiers mit der gegebenen Passnummer ändert sich von "economy" zu "business". Sollte ein Upgrade nicht möglich sein, soll eine Ausgabe erfolgen.
- *toString*: Diese Funktion soll eine textuelle Repräsentation der Flugplanung mit Flugnummer und -datum sowie den Sitzplatzbelegungen der Klassen erzeugen.

**Klasse.java** *(7 Punkte)*

- *getPassagiere*: Getter für alle Passagiere der Klasse.
- *getPassagier*: Gibt den Passagier mit der zugehörigen Passnummer zurück.
- *addPassagier*: Fügt den gegebenen Passagier der Klasse hinzu, wenn freie Plätze zur Verfügung stehen.
- *removePassagier*: Entfernt den Passagier mit der zugehörigen Passnummer von der Klasse.

**Passagier.java** *(2 Punkte)*

- *getPassnummer*: Getter für die Passnummer des Passagiers.
- *getName*: Getter für den Namen des Passagiers.

Entwerfen sie zudem eine Klasse **Main.java**, die die **main**-Methode enthält, mit der sie die implementierten Klassen testen. Initialisieren sie dazu eine Flugplanung für einen Flug und weisen diesem Passagiere zu. Testen sie dabei die Methoden *buchen*, *stornieren* und *upgrade* und geben sie sich den Flugstatus mit der *toString*-Methode aus. *(3 Punkte)*