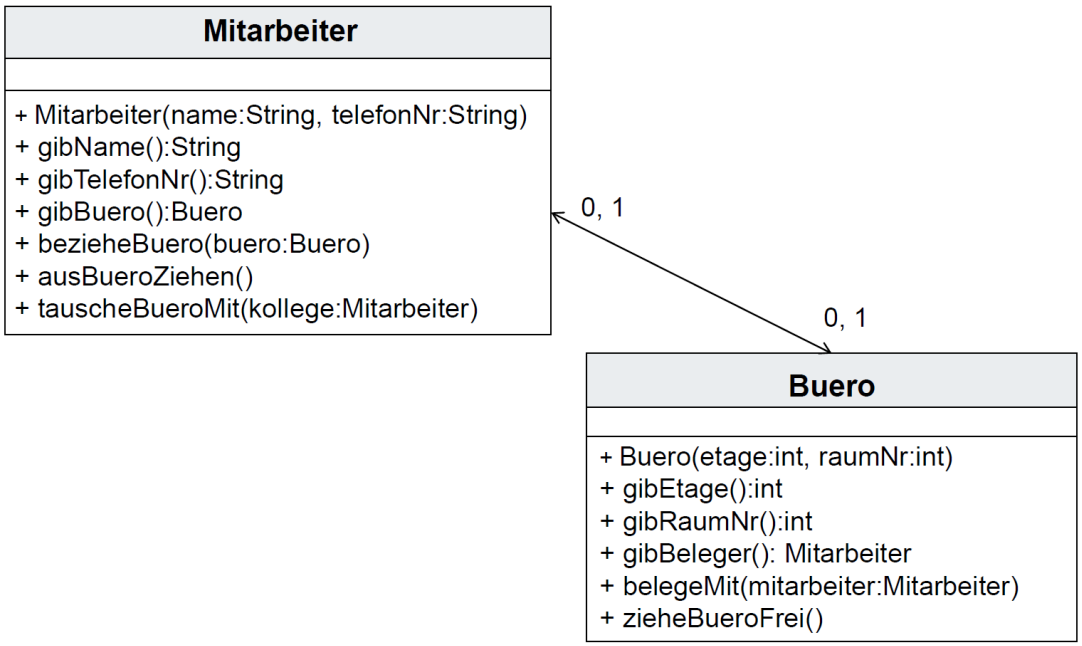
Verbundstudiengang Wirtschaftsinformatik (B.Sc.)
Algorithmen und Programmierung II
Praktikumsaufgaben

Prof. Dr. Anja Haake
Veranstaltung am 13.04.2024 (T3)

Aufgabe 1 – Funktionstests für objektorientierte Programme mittels Formulierung von Vor- und Nachbedingungen als UnitTests

Zur Raumplanung und Verwaltung der Kontaktdaten ihrer Mitarbeiter hat eine Firma ein erstes Klassenmodell entwickelt.



Dabei sollen folgende Randbedingungen gelten:
- Büros sind grundsätzliche Einzelbüros. (Die Methode `gibBeleger` der Klasse `Buero` gibt eine Referenz auf genau einen Mitarbeiter zurück.)
- Ein Mitarbeiter kann ein Büro nur beziehen, wenn das Büro noch nicht belegt ist.
- Generell soll nach der Ausführung jedes Konstruktors und jeder Methode gelten:
  Ist einem Mitarbeiter ein Büro zugewiesen, so ist das Büro von diesem Mitarbeiter belegt.
  Ist ein Büro von einem Mitarbeiter belegt, so ist dem Mitarbeiter das Büro zugewiesen.

Laden Sie die Dateien Buero.java, Mitarbeiter.java, BueroTest.java, MitarbeiterTest.java und AssoziationenTest.java in ein neues Java-Projekt.

(a) Bevor Sie die Java-Dateien studieren, überlegen Sie zunächst, welche der im UML-Klassendiagramm angegebenen Methoden der Klassen Mitarbeiter und Büro der Implementierung der Beziehung zu jeweils anderen Klasse dienen.
Welche Methoden entsprechen den in AuPI am T6 diskutierten set-, get- und remove-Methoden zur Implementierung von Beziehungen mit Obergrenze 1? (vgl. T6, Folie 25).

(b) Führen Sie die einzelnen TestCases beginnend mit `BueroTest`, dann `MitarbeiterTest` und schließlich `AssoziationenTest` aus, und "bringen Sie die fehlgeschlagenen Tests zum Laufen"! Bearbeiten Sie dabei die einzelnen Testmethoden in der Reihenfolge, in der sie im jeweiligen TestCase angegeben sind (Das ist nicht unbedingt die Reihenfolge, in der fehlschlagende Tests von JUnit angezeigt werden!). Typische Fehlerursachen sind:
   - nicht richtig/ausreichend implementierte Methoden in der/den zu testenden Klassen
   - nicht richtig/ausreichend formulierte/implementierte Testmethoden der Testklassen
   - nicht ausreichend formulierte Methoden `setUp` oder `tearDown`, die vor- bzw. nach jeder Testmethode ausgeführt werden.
   - noch überhaupt nicht implementierte Testmethoden, in deren Rümpfen jeweils nur der Methodenaufruf der JUnit-Methode `fail` steht, z.B. `fail("Hier ist noch was zu tun");`

Hinweise zum Arbeiten mit JUnit

BlueJ erkennt JUnit Testfälle (`TestCases`) und bietet für diese besondere Unterstützung. Sie können die Testfälle (hier also `MitarbeiterTest`, `BueroTest` und `AssoziationenTest`) einzeln ausführen (rechte Maus-Menu, Option "Test All").

In **Eclipse** ist regelmäßig ein JUnit Plug-in enthalten. Ggfs. ist das Plug-in noch dem *build path* Ihres Projektes hinzuzufügen (Properties des Projekts öffnen, unter Java Build Path den Reiter Libraries wählen, Schaltfläche "Add Library…").

Sie können TestCases (hier also MitarbeiterTest.java. BueroTest.java, AssoziationenTest.java) als auch direkt mit der Option "Run As\ Junit Test" (rechtes Maus-Menu) ausführen.

In beiden Werkzeugen führt ein Doppelklick auf einen fehlgeschlagenen Test direkt an die Programmzeile, die den Fehlschlag auslöste.