WHZ Fak. PTI **Fachgruppe Informatik**	**Grundlagen der Programmierung 2 (SS)**	Hausaufgabe 13
WS11/12		
F. Grimm / Prof. W. Golubski / O. Arnold / I. Klar / Dr. H. Steinchen		

Hausaufgabe 13

Die Hausaufgaben sollen zu Hause im Laufe der folgenden Woche selbstständig gelöst und über Ihr persönliches SVN-Repository auf dem Projektserver abgegeben werden. Legen Sie dazu ein neues Java-Projekt mit der Bezeichnung **„HA13_Vorname_Nachnachme“** an und laden Sie es in den trunk-Ordner Ihres persönlichen Repository hoch. Wenn Sie die endgültige Version hochladen, die für die Kontrolle bestimmt ist, **machen Sie das durch den Commit-Kommentar „Endgültige Version. Bitte kontrollieren.“ deutlich und schicken Sie bei vorzeitiger Abgabe eine kurze Mail an Herrn Arnold.** Sollte es zu dem Zeitpunkt der Abgabe Probleme mit dem Projektserver geben, schicken Sie das fertige Projekt notfalls gezippt per Mail an Herrn Arnold.

Hausaufgabe 13.1

Als Ausgangspunkt erhalten Sie die JAR-Datei *HA_13_RoomManager.jar* sowie die zugehörige Javadoc.

Fügen Sie also Ihrem neu angelegten Projekt zunächst die gegebene JAR-Datei hinzu (z.B. über das Kontextmenü des Projekts unter *Build Path → Add External Archives*…). Zum einfacheren Programmieren kann es auch sinnvoll sein, die zugehörige Javadoc in Eclipse verfügbar zu machen. Dies ist über das Kontextmenü der JAR-Datei in Ihrem Projekt möglich (*Properties → Javadoc Location*).

Die JAR-Datei enthält ein Interface sowie drei Klassen. Die Klasse `Room` stellt einen Unterrichtsraum der Hochschule mit seinen typischen Eigenschaften dar. Das Interface `RoomManager` beschreibt einen Verwalter für Räume mit typischen Methoden (Hinzufügen und Entfernen von Räumen sowie diverse Suchmethoden nach Räumen mit bestimmten Eigenschaften).

Die JAR-Datei enthält zwei Klassen, die dieses Interface implementieren: `GoodRoomManager` und `BadRoomManager`. Ihre Aufgabe besteht darin, mithilfe von Unit-Tests möglichst viele Fehler in der Implementierung von `BadRoomManager` zu entdecken. Die Klasse `GoodRoomManager` dient dabei als Leitfaden dafür, wie sich ein `RoomManager` eigentlich verhalten sollte.

Gehen Sie dazu wie nachfolgend beschrieben vor. Legen Sie zunächst eine Testklasse für `GoodRoomManager` an (bei Fragen dazu siehe Punkt 12.6 aus der letzten Hausaufgabe). In der Klasse können Sie eine oder mehrere Variablen vom Typ `RoomManager` anlegen (sowie einige Räume). Diese Variable hat also den Typ des Interfaces. Als Implementierung wählen Sie zunächst `GoodRoomManager`:

```
RoomManager rm1 = new GoodRoomManager();
```

Schreiben Sie nun sinnvolle Testmethoden gegen die `GoodRoomManager`-Implementierung. Hier müssten letzten Endes also alle Tests erfolgreich durchlaufen.

Tauschen Sie dann die genutzte Implementierung aus:

```
RoomManager rm1 = new BadRoomManager();
```

Ohne Veränderung der Testmethoden müssten jetzt einige der Tests aufgrund der in `BadRoomManager` enthaltenen Fehler fehlschlagen. Falls das nicht der Fall sein sollte, sind Ihre Tests noch zu oberflächlich und müssen verfeinert werden.

Verfeinern Sie Ihre Tests solange, bis Sie mindestens 4 Fehler in `BadRoomManager` gefunden haben (wissentlich eingebaut sind 8). Beachten Sie aber bei Ihren Überarbeitungen der Tests, dass diese bei der Nutzung von `GoodRoomManager` immer noch problemlos durchlaufen müssten.

Hinweis: Beachten Sie bitte ferner, dass eine JUnit-Testmethode immer nur bis zum ersten fehlgeschlagenen `Assert` ausgeführt wird. Kommentieren Sie somit ggf. `Asserts` zu bereits entdeckten Problemen bei Ihrer weiteren Suche aus.

Hausaufgabe 13.2

Beschreiben Sie direkt an passender Stelle in Ihren Tests in Quellcodekommentaren, welche Fehler in `BadRoomManager` Sie identifizieren konnten. Passende Stellen wären in der Regel direkt bei den `assert`-Methodenaufrufen, die bei der Verwendung von `BadRoomManager` fehlschlagen.