**Fragen, Antworten und Kommentare zur aktuellen Vorlesung**

Frage: Sollten eher viele kleine Tests oder ein Test pro Methode mit mehreren Untersuchungen geschrieben werden?

Antwort: Generell sollten Tests immer so ablaufen, dass nach der Durchführung des Tests (Arrange und Act) nur noch die Überprüfung durchgeführt wird. Dies garantiert, dass nicht ein Fehler frühzeitig zum Abbruch führt und der Rest des Tests nicht stattfindet. Statt mehrerer Überprüfungen eignet sich Assertions.assertAll sehr gut dazu, dass viele Detailüberprüfungen getrennt garantiert ausgeführt werden. Solle bei der Testerstellung der Wunsch nach Copy & Paste aufkommen, muss immer über parametrisierte Tests (Folien 102ff) nachgedacht werden.

Wie immer gibt es Ausnahmen, wenn es z. B. sehr lange dauert die zu testende Situation herzustellen.

Frage: Ich bekomme bei folgenden Programm folgende Exception, verstehe aber nicht, was passiert.

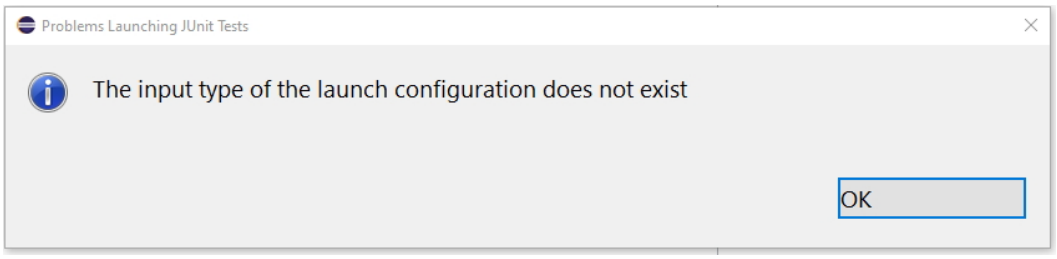
```
public static void main(String[] args) {
  List<Integer> toSort = Arrays.asList(4, 2, 5, 3, 6, 4);
  for(int i = 1; i < toSort.size(); i++) {
    int tmp = toSort.remove(i);
    boolean done = false;
    int pos = 0;
    while(!done && pos < i) {
      if (tmp < toSort.get(pos)) {
        done = true;
        toSort.add(pos, tmp);
      }
      pos ++;
    }
    if(!done) {
      toSort.add(pos, tmp);
    }
  }
  Assertions.assertEquals(List.of(2, 3, 4, 4, 5, 6), toSort);
}
```

```
Exception in thread "main" java.lang.UnsupportedOperationException
at java.base/java.util.AbstractList.remove(AbstractList.java:169)
at tmp/asList.Beispiel.main(Beispiel.java:15)
```

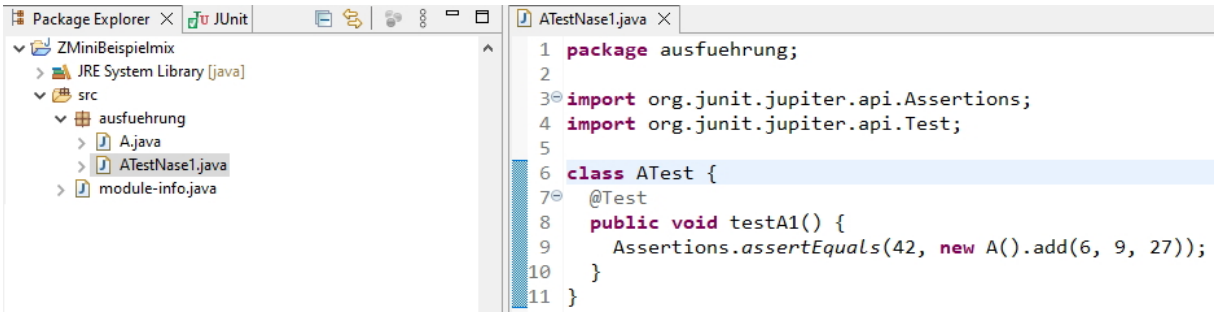
Antwort: Das Problem ist, dass Arrays.asList, genau wie List.of eine unveränderbare (immutable) Liste erzeugt. Es kann auf die Elemente nur lesend zugegriffen werden. Der Hintergrund ist, dass solche Listen sehr effizient zu nutzen sind (Basisklasse ImmutableCollections). Der folgende Trick löst das Problem, in dem alle Elemente der Collection zu einer neuen Collection hinzugefügt werden (alternativ ist auch addAll() nutzbar):

```
List<Integer> toSort = new ArrayList<>(Arrays.asList(4, 2, 5, 3, 6, 4));
```

Frage: Ich habe meine Testklasse kopiert und in ein neues Eclipse-Projekt geladen und kann die Tests nicht starten. Ich erhalte folgende Fehlermeldung:

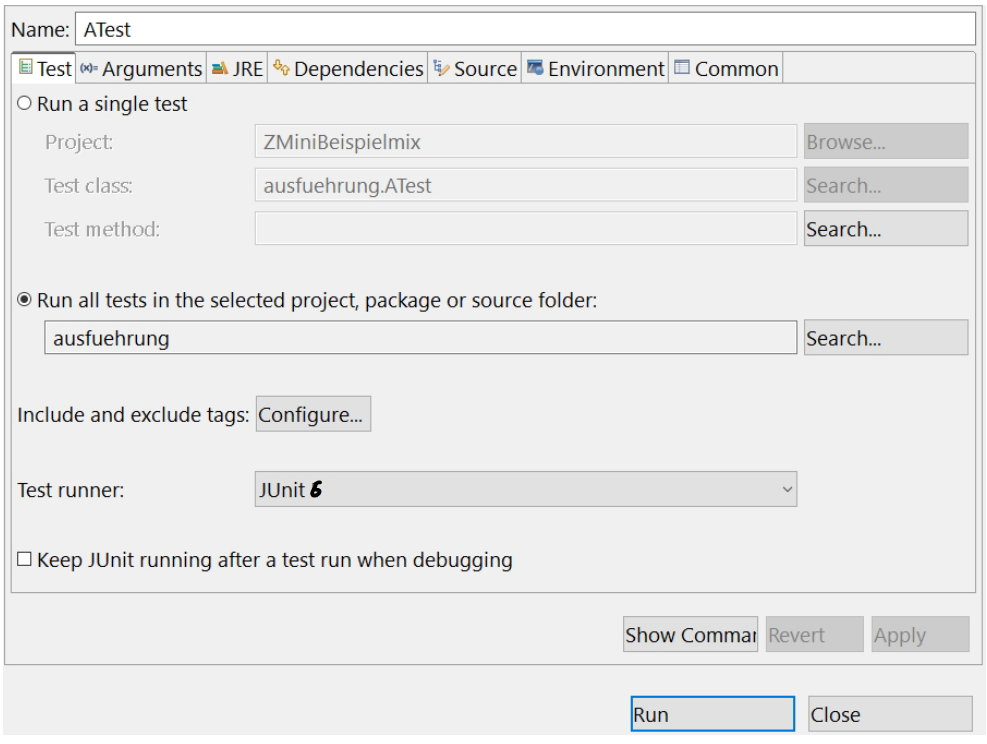



Antwort: Ihr Projekt sieht in Eclipse wie folgt aus:




Es gibt zwei verwobene Probleme. Ihre Java-Datei ATestNase1.java heißt nicht wie die programmierte Klasse ATest. Dies ist der Grund warum Eclipse das Projekt so nicht ausführen kann. Das ausgehende Problem ist aber, dass die Klasse ATest nur die Default-Sichtbarkeit in Java, also Sichtbarkeit nur im Package, erhalten hat. Dies Problem entsteht, wenn eine Entwicklungsumgebung Testklassen nicht automatisch als public deklariert. Nach Ergänzung von „public“ werden Sie aufgefordert den Namen der Klasse und der Datei zu vereinheitlichen.

Einen Workaround bietet die „Run Configuration“ in der der Haken auf „Run all tests in the selected project, package or source folder:“ gesetzt wird. Dadurch könnten natürlich weitere Testklassen im Projekt mitgestartet werden.



Frage: Mir ist im nach dem Praktikum wieder unklar, ob der gescheiterte vorletzte Test aus dem Excel-Sheet bei Aufgabe 8 ein falscher Test ist oder einen Fehler der Software zeigt.

Antwort: Der Testfall zeigt einen Fehler der Software, da die Testdaten korrekt sind. Dies geht aus folgender Formulierung in der Spezifikation hervor: „Die Methode liefert für ein Nutzerobjekt der Klasse „Gehoben“ genau dann wahr, wenn das Datenobjekt zu „Frei“ gehört und der Qualitätswert unter 60 liegt oder das Datenobjekt zu „Pauschal“ gehört und der Qualitätswert unter 40 liegt.“

Zur Erinnerung, „genau dann (wenn)“ bedeutet, dass „wenn A dann B“ und „Wenn B dann A“ gemeint bist. Daraus folgt, dass der Wert für „Gehoben“ und „Premium“ nicht erwähnt wird und damit false sein muss.

Dieser Fehler muss nebenbei bei a) nicht gefunden werden, da die einzelnen gültigen Äquivalenzklassen beliebig kombiniert werden können, so dass der Fehler nur durch Zufall gefunden werden kann.

Zur Erinnerung (oder Vorgriff): In der theoretischen Informatik wird bewiesen, dass es kein Programm gibt, dass als Eingabe ein Programm sowie eine Anforderung erhält und dass berechnet, ob das Programm diese Anforderung erfüllt oder nicht (Halteproblem bzw. allgemeiner Satz von Rice).