

 **Artemis** 6.4.0


☰

Course Overview



 ge64baw ▾

Courses > Praktikum: Grundlagen der Programmierung WS22/23 > Exercises > W03H02 - Testing fuer die Formel 1

 **W03H02 - Testing fuer die Formel 1**


Hausaufgabe

Medium

Submission due:


9 months ago


Points: 6 of 6

Assessment: automatic 

Complaint due:


9 months ago

 Resume practice in exercise

 100% (9 months ago)


GRADED

Recent results:



100%
(9 months ago)

GRADED



100%
(9 months ago)

GRADED

Show all results ▾

Tasks:

W03H02 - Testing für die Formel 1

In der kommenden Formel 1 Saison wollen die Raceuine mit einem eigenen Team antreten und haben dafür das Team Alfa Südpoleo gegründet. Allerdings müssen sie dafür noch einiges vorbereiten und entwickeln, u.a. auch Software. Unsere Raceuine sind nämlich sehr ambitioniert und wollen es direkt mit dem stärksten Gegner, Red Polar Bear, aufnehmen können. Um sicher zu gehen, dass Programme korrekt funktionieren, müssen diese getestet werden. Sie entscheiden sich dafür JUnit 5 zu benutzen. Dabei sind die Raceuine jedoch auf deine Hilfe angewiesen, da sie sich nicht mit JUnit-Tests auskennen. Aus der Präsenzaufgabe kennst du bereits einige Array-Methoden, die für die Verarbeitung großer Datenmengen benötigt werden und nun getestet werden müssen. Überlege dabei gut, welche Fehler sich bei den Implentierungen eingeschlichen haben könnten und durch welche Tests diese entdeckt werden können.

Hinweis: Das Thema Unit-Testing wird, wegen des Ausfalls um Allerheiligen, nicht einer Zentralübung besprochen. Stattdessen wird aber im Laufe des Wochenendes 05.11.-06.11. ein kleines Tutorial (Nachtrag, 07.11.: i.e. ein Foliensatz; der ist jetzt online auf [Moodle](#)) zum Thema Unit-Testing mit JUnit 5 veröffentlicht. Das mit dieser Aufgabe mitgelieferte Code-Template enthält allerdings bereits einen Beispiel-Test, sodass alles, was du zur Bearbeitung dieser Aufgabe tun musst, ist, JUnit 5 in deiner IDE hinzuzufügen und dann beliebig viele Methoden mit der [Annotation @Test](#) darüber zu schreiben, in denen du die unten aufgeführten Methoden testest.

Hinweis: (Nachtrag, 07.11. wegen Anfragen) Zum Testen von Konsolen-Output bitte `PinguLib.getConsoleOutput()` verwenden und nicht das, was wir in den Public und Hidden Tests für die Hausaufgaben wie W01H01 machen (siehe weiter unten).

Aufgabe:


Im Package `pgdp.tests` findest du die Klasse `ArrayTest`, in der von dir nun Tests für folgende Methoden implementiert werden sollen:

- `print()`
- `minAndMax()`
- `invert()`
- `intersect()`
- `linearize()`
- `bubbleSort()`

`bubbleSort` ist dabei eine Methode, die ein gegebenes Array in aufsteigender Reihenfolge sortiert. Also wird bei Eingabe `[3, 1, 4, 5, 2]` das Array `[1, 2, 3, 4, 5]` zurückgegeben und bei Eingabe `[4, 3, 4, 1, 1]` wird das Array `[1, 1, 3, 4, 4]` zurückgegeben. Der genaue Algorithmus wird in Woche 04 behandelt. Ihn musst du aber zum Testen nicht verstehen. Es reicht, wenn du das Eingabe-Ausgabe-Verhalten von `bubbleSort()` verstehst.

Wir haben in unseren Tests eurer Tests einige fehlerhafte Implementierungen der sechs Methoden vorbereitet, die es zu finden gilt. Schreibe für jede Methode ausreichend Tests, um alle diese falschen Implementierungen zu finden. Wir zeigen dir unsere fehlerhaften Implementierungen der sechs Methoden natürlich nicht. Beim Testen geht es schließlich darum, Fehler in einem Programm zu finden, ohne den konkreten Programmcode anzusehen. Allerdings teilen wir dir bei jedem Push bereits für jede der sechs Methoden mit, ob du alle falschen Implementierungen gefunden hast. So weißt du nach jedem Push direkt, wie viele der sechs maximal erreichbaren Punkte du mit diesem nach der Deadline erhalten würdest.

Nach der Deadline wird dir dann mitgeteilt, welche fehlerhaften Implementierungen deine Tests genau übersehen, falls sie noch welche übersehen.

 **Korrekte Implementierung**

No results

Dieser Test prüft, ob deine Tests die korrekte Implementierung akzeptieren. Dieser Test muss durchlaufen, sonst gibt es keine Punkte und die Raceuine wären ganz traurig :(

Struktur des Repos:

- Die `PinguLib` kann dir dabei helfen, Output von der Konsole zu lesen.
 - Vor jedem Test muss die Methode `setup()` ausgeführt werden,

https://artemis.ase.in.tum.de/courses/201/exercises/8312

1/2

- dann kann mit `getConsoleOutput()` der Output auf der Konsole eingelesen werden,
 - und nach jedem Test muss die Methode `reset()` ausgeführt werden.
- Im Package `pgdp.array` findest du die korrekte Implementierung der oben genannten Methoden. Aus praktischen Gründen sind dabei die Implementierungen im `ArrayInterface`, welche von der `Array`-Klasse aufgerufen werden (Was ein Interface ist, lernst du später noch). Für dich ist nur wichtig, dass du über die oben in `ArrayTest` stehenden Imports auf die Methoden zugreifen kannst. Falls du fehlerhafte Implementierungen ausprobieren möchtest, kannst du die Methoden-Bodies im `ArrayInterface` beliebig verändern.


Tipps:

- Pro Methode werden nicht mehr als 4 Tests benötigt
- Pro Methode wird meistens mehr als ein Test benötigt
- [Hier findest du unter "Annotation Interfaces" die Documentation einiger Annotations von JUnit-5, die nützlich sein können](#)

[Lösungsvorschlag](#)

[Tests](#)

Exercise details	
Release date:	Nov 3, 2022 18:30
Submission due:	Nov 13, 2022 18:00
Complaint due:	Nov 20, 2022 18:00

Every student is allowed to complain once per exercise. In total 1000 complaints are possible in this course. You still have **998** complaints left. 

[About](#)

[Request change](#) [Release notes](#) [Privacy Statement](#) [Imprint](#)