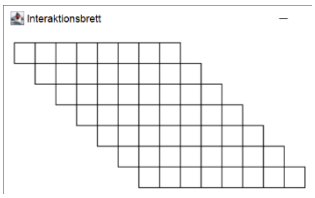
Prof. Dr. Stephan Kleuker
Hochschule Osnabrück
Fakultät Ing.-Wissenschaften und Informatik
- Software-Entwicklung -

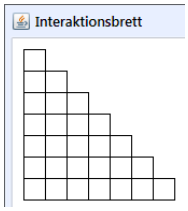
Programmierung 1

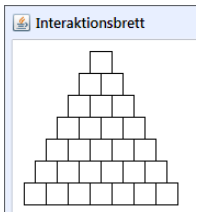
Wintersemester 2023/24
Aufgabenblatt 8

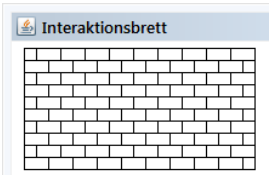
27. Aufgabe (3 Punkte, einfache geschachtelte Schleifen)

Schreiben Sie eine Klasse Dekoration, die eine Objektvariable vom Typ Interaktionsbrett nutzt und die folgenden Methoden implementiert.







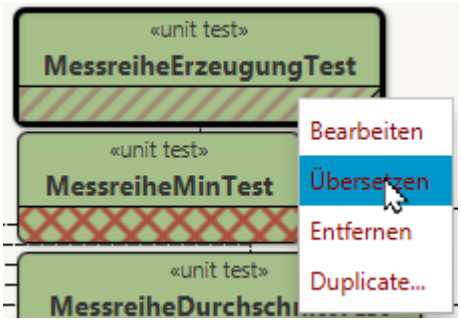


a) Mit Hilfe zweier ineinander geschachtelter Schleifen entsteht die gezeigte Raute.
b) Mit Hilfe zweier ineinander geschachtelter Schleifen entsteht die gezeigte Treppe.
c) Mit Hilfe zweier ineinander geschachtelter Schleifen entsteht die gezeigte Pyramide.
d) Mit Hilfe zweier ineinander geschachtelter Schleifen entsteht die gezeigte Mauer.

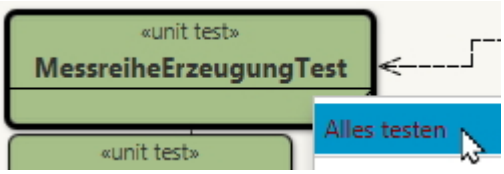
28. Aufgabe (13 Punkte, Schleifen, Achtung: Klausurähnlichkeit, muss zum Bestehen selbständig gelöst werden können)

Anmerkung: Bei Aufgaben, die mit „Klausurähnlichkeit“ gekennzeichnet sind, ist es sehr sinnvoll, diese zunächst auf Papier ohne weitere Hilfsmittel zu lösen. Danach können Sie sich mit Anderen austauschen und in BlueJ programmieren. Zum Bestehen müssen die Ideen der genutzten Algorithmen klar erkennbar sein, um zumindest die Hälfte der Punkte bekommen zu können.

Laden Sie das Projekt zur Aufgabe von der Veranstaltungsseite. In dem Projekt befinden sich zu jeder Programmieraufgabe Tests in einer eigenen Testklasse. Da die genaue Nutzung von Tests erst in der folgenden Vorlesung beschrieben wird, hier die Vorgehensweise zur Testnutzung. Bearbeiten Sie zunächst die Aufgabe, z. B. a). Wenn dann Ihre Klasse Messreihe übersetzbar/kompilierbar ist. Dann übersetzen Sie die zur Aufgabe angegebene Testklasse, bei a) MessreiheErzeugungTest. Sollten hierbei bereits Fehler auftreten, haben Sie die Aufgabenstellung nicht genau erfüllt, analysieren Sie die Fehlermeldung, die Konstruktoren oder Methoden hjaben nicht die in der Aufgabenstellung geforderte Form.

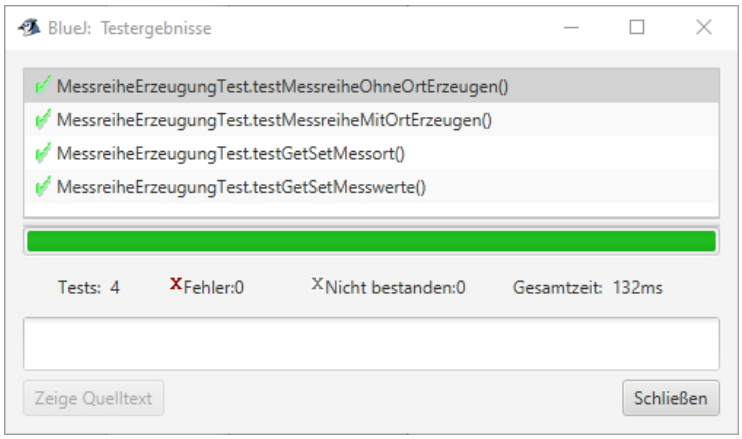


Machen Sie dann einen Rechtsklick auf der übersetzten Testklasse und wählen Sie „alle Tests“. Das Testergebnis wird Ihnen mit etwaigen Fehlermeldungen in einem Testergebnisfenster angezeigt. Bei Fehlern korrigieren Sie Ihren Code und führen Sie die Tests wieder aus, bis Sie eine Erfolgsmeldung erhalten.



Es ist natürlich nicht verboten sich den Testcode genauer anzusehen, um so zu verstehen, was getestet wird.

Wenn Sie alle Aufgaben bearbeitet haben, übersetzen Sie die Klasse AllTest und führen Sie die Methode executeAllTests() aus. Das Ergebnis im Konsolenfenster sollte sein, dass alle Tests laufen.



Prof. Dr. Stephan Kleuker
Hochschule Osnabrück
Fakultät Ing.-Wissenschaften und Informatik
- Software-Entwicklung -

Programmierung 1

Wintersemester 2023/24
Aufgabenblatt 8

Schreiben Sie die Klasse Messreihe, die als Objektvariablen den Messort in einer Objektvariablen `messort` und eine Sammlung von ganzzahligen Messwerten in einer Objektvariablen `messwerte` enthält und realisieren Sie folgende Methoden.
Hinweis: Natürlich dürfen Sie weitere Methoden zum Strukturieren Ihres Programms oder bei Code-Wiederholungen schreiben. Sie dürfen weiterhin *vereinfachend* davon ausgehen, dass nur echte Integer-Objekte in die Messwert-Sammlung und keine null-Werte eingetragen werden. Die Sammlung selbst könnte aber null sein. *Als Einschränkung dürfen von der Klasse ArrayList<Integer> nur der parameterlose Konstruktor und die Methoden add(Integer), get(int), set(int,int), isEmpty(), iterator() und size() genutzt werden.*
Überlegen Sie bei den zu erstellenden Methoden immer, wie Sie Spezialfälle (null-Wert für die gesamte Liste, leere Listen) sinnvoll bearbeiten müssen, damit es zu keinen Fehlermeldungen (genauer Exceptions) kommt.

a) Realisieren Sie die genannte Klasse mit Konstruktoren für keinen, und einen String-Parameter, die `messwerte` mit einer leeren Liste initialisiert, get- und set-Methoden für alle Objektvariablen, einer hinzufuegen(int)-Methode für Messwerte mit der ein Messwert in die Sammlung eingefügt wird, und einer toString()-Methode für eine nette Ausgabe (ArrayList-Objekte realisieren toString() bereits sinnvoll und das soll genutzt werden). [Testklasse: MessreiheErzeugungTest]
b) Schreiben Sie eine Methode min(), die den kleinsten Messwert zurückgibt. Falls es keinen Messwert gibt, soll die Zahl 0 zurückgegeben werden. Für eine enthaltene Messwert-Sammlung {1,7,4,4,2} soll z. B. das Ergebnis 1 sein. [Testklasse: MessreiheMinTest]
c) Schreiben Sie eine Methode durchschnitt(), die den Durchschnitt der Messwerte als Double-Wert zurückliefert (ein int-Wert kann z. B. durch „`* 1.0`“ in einen double-Wert gewandelt werden). Existieren keine Messwerte ist das Ergebnis 0. Für eine enthaltene Messwert-Sammlung {1,7,4,4,2} soll z. B. das Ergebnis 3.6 sein. [Testklasse: MessreiheDurchschnittTest]
d) Schreiben Sie eine Methode anzahlZwischen(int,int), die zwei ganzzahlige (Grenz)-werte mi und ma in dieser Reihenfolge übergeben bekommt und die Anzahl der Elemente zurückgibt, deren Werte innerhalb dieser Grenzen (einschließlich) liegen. Nutzen Sie eine while-Schleife. Für eine enthaltene Messwert-Sammlung {1,7,4,4,2} die Parameter mi=2 und ma=6 soll z. B. das Ergebnis 3 sein, da es 2,4,4 gibt. [Testklasse: MessreiheAnzahlZwischenTest]
e) Methode anzahlZwischen2(int,int), gleiche Aufgabe wie in d), aber mit for-Schleife. [Testklasse: MessreiheAnzahlZwischen2Test]
f) Methode anzahlZwischen3(int,int), gleiche Aufgabe wie in d), aber mit forEach-Schleife. [Testklasse: MessreiheAnzahlZwischen3Test]
g) Methode anzahlZwischen4(int,int), gleiche Aufgabe wie in d), aber mit Iterator, die Schleifenart können Sie wählen. [Testklasse: MessreiheAnzahlZwischen4Test]
h) [ab hier evtl. geschachtelte Schleifen] Schreiben Sie eine Methode zweiGleiche(), die genau dann true zurückgibt, wenn es mindestens zwei gleiche Messwerte in den enthaltenen Messwerten gibt, z. B. soll für Messwerte {1,3,5,7,3} das Ergebnis true und für {7,3,1,4} das Ergebnis false sein. [Testklasse: MessreiheZweiGleicheTest]
i) Schreiben Sie eine Methode verschieden(Messreihe), die ein Messreihen-Objekt übergeben bekommt und die genau dann true zurückgibt, wenn kein Messwert in beiden Objekten vorkommt, z. B. soll {3,3,2,4,4} in this zusammen mit {1,5,5,5} aus der übergebenen Messreihe true ergeben, da es keine gemeinsamen Werte gibt. Nicht vorhandene Listen (null) sind in dieser Methode für das Ergebnis wie leere Listen zu behandeln. [Testklasse: MessreiheVerschiedenTest]
j) Schreiben Sie eine Methode alleWerteGleichOft(Messreihe), die ein Messreihen-Objekt übergeben bekommt und die genau dann true zurückgibt, wenn jeder Wert der einen Messreihe genauso oft in der anderen Messreihe vorkommt, z. B. soll {3,3,2,4,4} in this zusammen mit {4,3,2,4,3} aus der übergebenen Messreihe true ergeben. Sie

sehen, dass die Reihenfolge irrelevant ist. Nicht vorhandene Listen (null) sind in dieser Methode für das Ergebnis wieder wie leere Listen zu behandeln.
Hinweis: Es könnte sinnvoll sein, zunächst eine Methode zu schreiben, die zählt, wie oft ein Wert in den Messwerten vorkommt. [Testklasse: MessreiheAlleWerteGleichOftTest]

k) Schreiben Sie eine Methode gleicheWerte(Messreihe), die ein Messreihen-Objekt übergeben bekommt und die genau dann true zurückgibt, wenn jeder Wert der einen Messreihe mindestens einmal in der anderen Messreihe vorkommt, z. B. soll {3,3,2,4,4} zusammen mit {4,2,2,2,3,2} true ergeben, da in beiden nur 2, 3, 4 vorkommen. Die Häufigkeit und Reihenfolge der Werte spielt keine Rolle. Nicht vorhandene Listen (null) sind in dieser Methode für das Ergebnis wieder wie leere Listen zu behandeln. [Testklasse: MessreiheGleicheWerteTest]

Hinweis: Natürlich dürfen Sie sich Hilfsmethoden schreiben. Falls es für Sie hilfreich ist, können Sie auch zuerst ein Aktivitätsdiagramm entwerfen.