# Miniprojekt 2

### **Abgabefrist: Dienstag, 17. November 2015, 23:59 Uhr**

In diesem Miniprojekt befinden sich zwei Klassen, Measurement und Matrix, die von Ihnen bearbeitet werden sollen. Die beiden Dateien sind unabhängig voneinander und haben jede eine main-Methode, mit der sie die jeweiligen Aufgabenteile testen können.

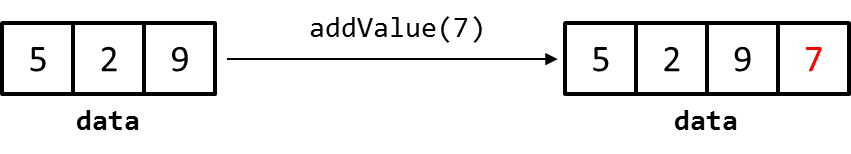
Die Klassen-, Variablen- und Methodennamen dürfen nicht verändert werden!

Bitte beachten Sie, dass für die jeweilige Testatzulassung die Abgabe einer (nicht notwendigerweise korrekten) Lösung zu jedem Miniprojekt erforderlich ist!

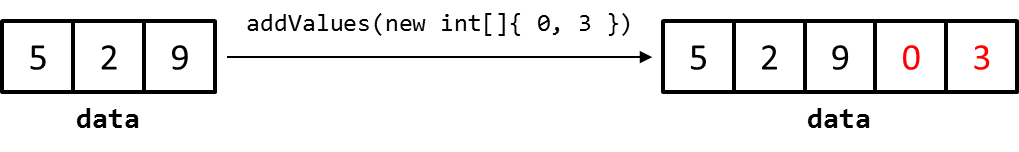
## Aufgabe 1: Klasse Measurement

Die Klasse Measurement verwaltet über das Feld data eine Messreihe in Form eines eindimensionalen Arrays von Integer-Werten. Über die Hilfsmethode Measurement.printData() können Sie die Messreihe ausgeben lassen.

1. **Konstruktor**  
   Implementieren Sie den Konstruktor so, dass data als neues int-Array mit der Länge 0 initialisiert wird.
2. **Messwert hinzufügen**  
   Die Methode addValues(int value) nimmt einen Messwert value entgegen und soll diesen als neuen Messwert an das Ende des data-Arrays setzen. Erzeugen Sie dazu ein neues, vergrößertes Array.



1. **Mehrere Messwerte hinzufügen**  
   Die Methode addValues(int[] values) nimmt ein ganzes Array von Messwerten entgegen und soll diese als neue Messwerte an das Ende des data-Arrays setzen. Erzeugen Sie auch hier ein neues, vergrößertes Array.



1. **Kleinster Wert**  
   Die Methode getMinimumValue() soll den kleinsten, in data enthaltenen Wert zurückgeben. Sie können den Fall, dass data keine Messwerte enthält, ignorieren.
2. **Werte über Schwellenwert**  
   Implementieren Sie die Methode getValuesAboveThreshold(int threshold) so, dass sie ein int-Array mit allen Werten aus data zurückgibt, die echt größer sind als der Schwellenwert treshold. Entspricht kein Wert diesem Kriterium oder enthält data keine Elemente, soll die Methode ein int-Array der Größe 0 zurückgeben.

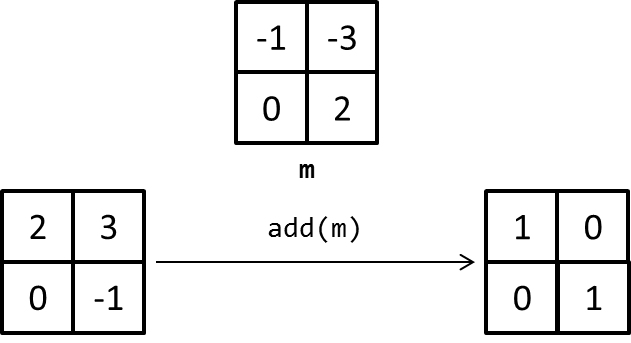
## Aufgabe 2: Klasse Matrix

Die Klasse Matrix verwaltet eine Matrix in Form eines zweidimensionalen int-Arrays. Dabei gibt die erste Dimension die Zeile, die zweite Dimension die Spalte eines Wertes in der Matrix an, d.h. values[0][1] bezeichnet den Wert in der ersten Zeile und zweiten Spalte, values[2][0] bezeichnet den Wert in der dritten Zeile, ersten Spalte, usw.:

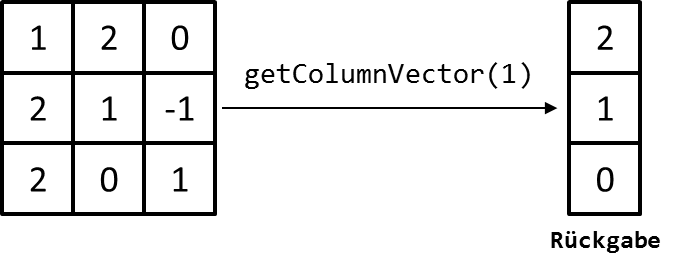
Mit der Methode print() bietet die Klasse Matrix eine Hilfsmethode zur formatierten Ausgabe ihrer Werte an.

*Hinweis*: Gehen Sie davon aus, dass die von Ihnen implementieren Methoden immer mit korrekten Parametern aufgerufen werden (bspw. der Konstruktor nicht mit null oder add() nie mit null bzw. einer Matrix inkompatibler Größe aufgerufen)

1. **Konstruktor**  
   Ergänzen Sie den Konstruktor so, dass dem Feld Matrix.values der Parameter initialValues zugewiesen wird
2. **Skalare Multiplikation**  
   Implementieren Sie die Methode scalarMultiplication(int c) so, dass sie die durch values definierte Matrix mit dem Skalarwert c multipliziert:
3. **Addition**  
   Die Methode add(Matrix m) soll die übergebene Matrix m zu der durch values definierten Matrix addieren:



1. **Spaltenvektor**Die Methode getColumnVector(int col) soll den Vektor, der Spalte col beschreibt, als int-Array zurückgeben:

****

1. **Vergleich**

**Die Methode** isEqualTo(Matrix m) **soll** true **zurückgeben, wenn die Matrix, auf der die Methode aufgerufen wurde, identisch zu** m **ist, ansonsten** false**. Zwei Matrizen sind identisch, wenn**

* **sie in der Anzahl der Zeilen übereinstimmen**
* **sie in der Anzahl der Spalten übereinstimmen**
* **jeder Wert** aij **aus der einen Matrix mit** aij **aus der anderen Matrix übereinstimmt**

1. **Transponieren  
   Die Methode** transpose() **soll die in** values **definierte Matrix transponieren, d.h. an ihrer Hauptdiagonalen spiegeln („Zeilen und Spalten werden vertauscht“):**