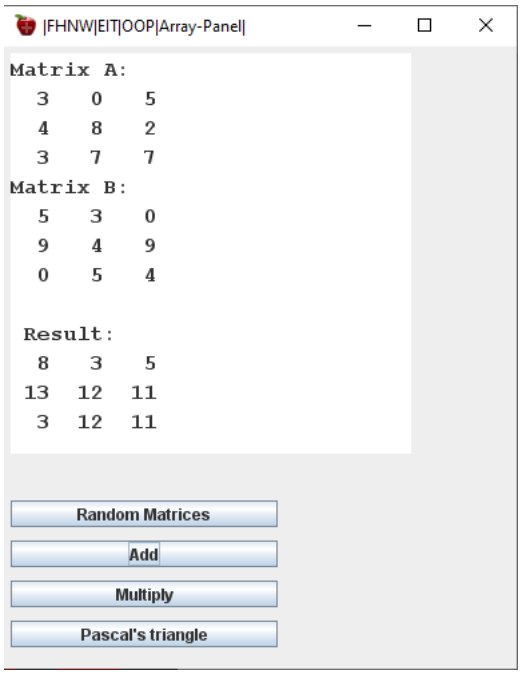
Übung : Arrays ohne Ende….

Ziel dieser Übung ist es, den Umgang mit Arrays, deren Initialisierung und Berechnungen zu vertiefen. Hierzu bauen wir ein Array-Panel, mit dem Matrizen generiert und verrechnet werden können.



Die Struktur des Codes sieht folgendermassen aus:

- Wie gewohnt verwenden wir das altbekannte JFrame mit einem JPanel, wo ein JTextArea zur Ausgabe der Matrizen eingebettet ist.
- Die beiden Matrizen A und B sollen von der Dimension 3x3 sein und wie folgt initialisiert werden: $\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$
- Mit **Add** werden beide Matrizen addiert und das Ergebnis im JTextArea dargestellt.
- Mit **Multiply** werden beide Matrizen multipliziert und das Ergebnis im JTextArea dargestellt.
- Mit **Pascals triangle** wird das Pascalsche Dreieck zur Bestimmung der Binomialkoeffizienten berechnet und im JTextArea dargestellt.

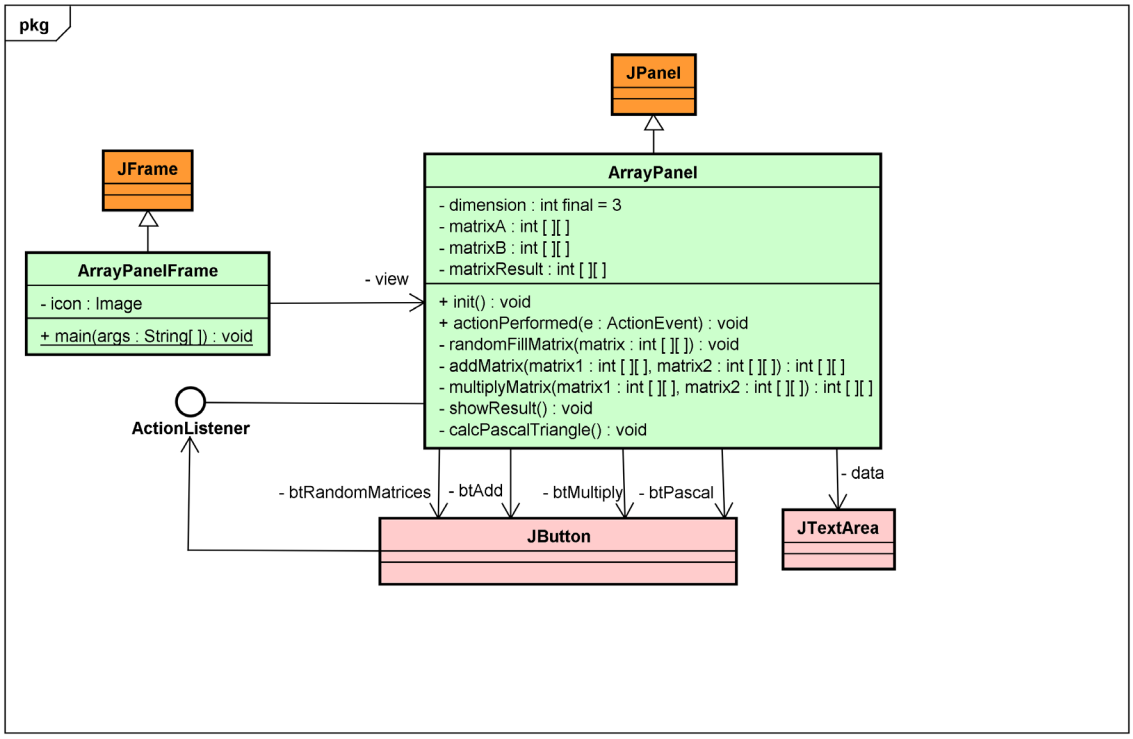
```
          1
        1   1
      1   2   1
    1   3   3   1
  1   4   6   4   1
1   5  10  10   5   1
1   6  15  20  15   6   1
```

$$(a+b)^2 = a^2 + 2ab + b^2$$
$$(a+b)^3 = a^3 + 3a^2b + 3ab^2 + b^3$$

→ Das Pascalsche Dreieck ist ein Beispiel für ein "nichtrechteckiges" zweidimensionales Array. Jede Zahl ergibt sich aus der Summe der zwei links und rechts über ihr stehenden Zahlen.

1) Grundgerüst

Vervollständigen Sie das Gerüst der Vorlage gemäss folgende Klassendiagramm



2) Initialisierung und GUI

Initialisieren Sie die Matrizen ***matrixA*** und ***matrixB*** gemäss Vorgaben (matrixResult ist bei der Deklaration default mit 0 initialisiert).

Bauen Sie in der ***init()*** Methode von **ArrayPanel** das GUI gemäss obiger Skizze auf und bringen Sie die Matrizen mit ***showResult()*** (ist in der Vorlage bereits vorgegeben) auf dem **JTextArea** zur Darstellung.

3) Implementierung *actionPerformed()*

Durch Drücken der jeweiligen Buttons sollen die jeweiligen Aktionen per Methodenaufruf ausgeführt werden.

4) Implementierung *addMatrix()*

Zwei n x n Matrizen werden addiert, indem die einzelnen Elemente addiert werden. Die Dimension der resultierenden Matrix bleibt erhalten.

5) Implementierung *multiplyMatrix()*

Zwei Matrizen gleicher Dimension lasse sich immer multiplizieren. Die einzelnen Elemente der Ergebnismatrix ergeben sich nach der Rechenvorschrift:

$$c_{ik} = \sum_{j=0}^{n-1} a_{ij} \cdot b_{jk}$$

c_{ik} ist dann dass Element in der i-ten Zeile, k-te Spalte

6) Testen der Matrixoperation

Mit den Initialwerten der Matrizen ergibt sich als Resultat für die Addition $\begin{pmatrix} 2 & 4 & 6 \\ 8 & 10 & 12 \\ 14 & 16 & 18 \end{pmatrix}$

und für die Multiplikation $\begin{pmatrix} 30 & 36 & 42 \\ 66 & 81 & 96 \\ 102 & 126 & 150 \end{pmatrix}$.

7) Zufällige Matrizen erstellen

Implementieren Sie nun die Funktion ***randomFillMatrix()*** die für jedes Element der Matrix Zahlen zwischen 0 und 9 zufällig erstellt.

8) Pascalsches Dreieck

In der Funktion ***CalcPascalTriangle()*** wird sowohl die nichtrechteckige Matrix berechnet als auch zur Darstellung gebracht. Gemäss Beschreibung soll ein Dreieck mit 7 Zeilen realisiert werden.

Orientieren Sie sich bei der Implementierung der Darstellung auf dem **JTexArea** an der Methode ***showResult()***.

Bringen Sie das Ergebnis zunächst auf die Form

```
1
1   1
1   2   1
1   3   3   1
1   4   6   4   1
1   5  10  10   5   1
1   6  15  20  15   6   1
```

Das Dreieck erfolgt dann mit einem kleinen Formatierungstrick, in dem Sie folgende Zeilen zu Beginn der äusseren for- Schleife ergänzen ("black magic").

```
String spc = new String(new char[(14 - 2 * row)]).replace('\0', ' ');
data.append(spc);
```

Dadurch werden dann die passenden Einrückungen erzeugt.