1 Erstellen der Projektstruktur

Erstellen Sie sich unter Eclipse ein neues Java-Projekt und erzeugen Sie die folgenden Packages und Klassen:

- Package `main`, Klasse `main.Main`
- Package `matrix`, Klasse `matrix.MatrixTools`

```
├── main
│   └── Main.java
└── matrix
    └── MatrixTools.java
```

Abbildung 1: Ordner- und Dateistruktur im src-Verzeichnis

2 Klasse `matrix.MatrixTools`

Erstellen und vervollständigen Sie die Klasse `matrix.MatrixTools` um die Methoden, welche in den folgenden Unterabschnitten beschrieben sind!

2.1 Methode `createMatrix()` 2P.

Implementieren Sie die Methode `public static int[][] createMatrix(int m, int n)`.

Parameter: Dimension der Matrix, mit m = Anzahl der Zeilen, n = Anzahl der Spalten

Rückgabe:

- initialisiertes *Array* (Matrix) der Dimension $m \times n$
- `null` im Fehlerfall, d.h. bei unzulässigen Werten für m oder n

Beschreibung: Die Methode erzeugt und initialisiert ein zweidimensionales *Array* (Matrix) vom Typ `int[][]`. Die Elemente der erzeugten Matrix sollen mit Zufallszahlen zwischen 1 und 10 initialisiert werden. Nutzen Sie hierfür die Java API-Funktion `Math.random()`.

2.2 Methode `printMatrix()` 1P.

Implementieren Sie die Methode `public static void printMatrix(int[][] m)`.

Parameter: zweidimensionales *Array* (Matrix)

Beschreibung: Die Methode soll die übergebene Matrix in geeigneter Form auf die Kommandozeile ausgeben. Der Parameter ist auf `null`-Referenzen zu prüfen und bei Auftreten eine Leerzeile auszugeben.

2.3 Methode `getTransposedMatrix()` 2P.

Implementieren Sie die Methode
`public static int[][] getTransposedMatrix(int[][] m)`.

Parameter: zweidimensionales *Array* (Matrix)

Rückgabe:

- die transponierte Matrix
- die Parameter-Matrix `m` ist auf `null`-Referenzen und die Rechteck-Bedingung[1] zu prüfen ⇒ bei Auftreten ist `null` zurückzugeben

Beschreibung: Die Methode soll die Transponierte der als Parameter übergebenen Matrix berechnen und zurückgeben.

Die Transponierte einer Matrix erhalten Sie, indem Sie die Zeilen und Spalten der gegebenen Matrix vertauschen. Die erste Zeile der transponierten Matrix entspricht der ersten Spalte der gegebenen Matrix, die zweite Zeile der zweiten Spalte und so weiter, für jede Spalte der gegebenen Matrix. Bildlich gesprochen, ergibt sich die Transponierte einer Matrix durch Spiegelung an der Hauptdiagonalen.

Das folgende Beispiel zeigt die Berechnung der Transponierten einer Matrix:

$$\mathbf{A} = \begin{pmatrix} 1 & 5 & 9 \\ 3 & 6 & 2 \end{pmatrix} \qquad \mathbf{A}^T = \begin{pmatrix} 1 & 3 \\ 5 & 6 \\ 9 & 2 \end{pmatrix}$$

2.4 Methode `isSymmetric()` 2P.

Implementieren Sie die Methode `public static boolean isSymmetric(int[][] m)`.

Parameter: zweidimensionales *Array* (Matrix)

Rückgabe: `true` wenn symmetrisch (Rechteck-Bedingung[1] erfüllt), sonst `false`

Beschreibung: Die Methode `isSymmetric()` soll die übergebene Matrix auf Symmetrie prüfen.

Eine symmetrische Matrix muss zum einen quadratisch sein und zum anderen sind alle Elemente spiegelsymmetrisch zur Hauptdiagonalen. Demnach ist eine symmetrische Matrix identisch mit ihrer transponierten Matrix: $\mathbf{A} = \mathbf{A}^T$.

Ein Beispiel für eine symmetrische Matrix ist:

$$\mathbf{A} = \begin{pmatrix} 1 & 4 & 5 \\ 4 & 2 & 3 \\ 5 & 3 & 6 \end{pmatrix}$$

[1] Alle Zeilen der Matrix müssen die gleiche Länge besitzen

2.5 Methode `matrixSpur()` 2P.

Implementieren Sie die Methode `public static int matrixSpur(int[][] matrix)`.

Parameter: zweidimensionales *Array* (Matrix)

Rückgabe:

- berechnete Spur der Parameter-Matrix m
- wenn Matrix m **nicht quadratisch**, oder Rechteck-Bedingung[1] nicht erfüllt, Rückgabewert $= 0$

Beschreibung: Die Methode soll die Spur der als Parameter übergebenen Matrix berechnen und zurückgeben.

In der linearen Algebra bezeichnet man als Spur einer quadratischen $n \times n$-Matrix A über einem Körper K die Summe der Hauptdiagonalelemente dieser Matrix.

Für eine Matrix A ergibt sich $Spur(A)$ wie folgt:

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \qquad Spur(A) = \sum_{j=1}^{n} a_{jj} = a_{11} + a_{22} + \ldots + a_{nn} \in K$$

Gilt Spur$(A) = 0$, so bezeichnet man die Matrix A als spurfrei.

2.6 Zusatzaufgabe: Methode `matrixMul()` 3P.

Implementieren Sie die Methode
`public static int[][] matrixMul(int[][] a, int[][] b)`.

Parameter: die zu multiplizierenden Matrizen a und b

Rückgabe:

- die Ergebnismatrix der Multiplikation
- im Falle nicht multiplizierbarer Matrizen, Rückgabewert `null`

Beschreibung: Die Methode soll die beiden übergebenen Matrizen multiplizieren und die Ergebnismatrix zurückgeben.

Vor der Multiplikation soll die Kompatibilität der Parameter-Matrizen a und b geprüft werden. Die beiden Matrizen sind multiplizierbar, wenn die Spaltenanzahl der linken Matrix der Zeilenanzahl der rechten Matrix entspricht.

Bei der Multiplikation der Matrizen **A** und **B** zur Ergebnismatrix **R** berechnen sich die Elemente r_{ij} wie folgt:

$$r_{ij} = \sum_{n=0}^{Ba-1} a_{in} b_{nj}$$

Folgende Berechnung zeigt die Matrizenmultiplikation an einem Beispiel:

$$\mathbf{A} = \begin{pmatrix} 1 & 5 & 9 \\ 3 & 6 & 2 \end{pmatrix} \quad \mathbf{B} = \begin{pmatrix} 2 & 3 \\ 0 & 2 \\ 5 & -4 \end{pmatrix}$$

Die Multiplikation findet wie folgt statt:

$$\begin{pmatrix} 1 & 5 & 9 \\ 3 & 6 & 2 \end{pmatrix} \cdot \begin{pmatrix} 2 & 3 \\ 0 & 2 \\ 5 & -4 \end{pmatrix} = \begin{pmatrix} 1\cdot 2+5\cdot 0+9\cdot 5 & 1\cdot 3+5\cdot 2+9\cdot -4 \\ 3\cdot 2+6\cdot 0+2\cdot 5 & 3\cdot 3+6\cdot 2+2\cdot -4 \end{pmatrix} = \begin{pmatrix} 47 & -23 \\ 16 & 13 \end{pmatrix}$$

3 Bewertung

Aufgabe	Vorraus-setzung	Punkte
Programm compilierbar	X	-
Klassen, Packages, Methoden, Signaturen richtig benannt/umgesetzt	X	-
`createMatrix()`		2P.
`printMatrix()`		1P.
`getTransposedMatrix()`		2P.
`isSymmetric()`		2P.
`matrixSpur()`		2P.
Zusatzaufgabe `matrixMul()`		3P.
Gesamtpunkte 100%:		9P.
maximale Gesamtpunkte 133%:		12P.