課題３　ベクトル、行列の基本的処理

３－１　ベクトル、行列オブジェクトの生成と表示

授業中に紹介したベクトルや行列に関する様々な処理を実現するため、ベクトル、行列オブジェクトを生成するクラスを作成せよ。作成するクラスの仕様は以下の通り。

- 作成するクラス名はMatrixとする。作成の前に、以下の注意書きをよく読むこと
- インスタンス変数は以下の3種類
  - **m**：ベクトル or 行列の要素を格納するdouble 型の2次元配列
  - **numOfRow**：ベクトル or 行列の”行数”を格納（int型）
  - **numOfColumn**：ベクトル or 行列の”列数”を格納（int型）
- ベクトルについては、k次元の行ベクトル＆列ベクトルを、行列についてはm × n の行列を生成できるようにすること（k： ベクトルの要素数、m： 行要素数、n：列要素数　；要素数は全て1以上　＆　**k次元の行ベクトルは 1 × k の行列、k次元の列ベクトルは k × 1の行列**と同一）
- Matix型のインスタンスを生成するコンストラクタとして、以下に示す2種類は最低限、必ず作成すること
  - **二次元配列 input の値を行列の要素として m に格納（行数、列数のそれぞれも numOfRow & numOfColumn に格納）**するコンストラクタ
  - **一次元配列 input の値を行ベクトルの要素として、1行k列（k: 配列の要素数)の二次元配列mに格納（行数、列数のそれぞれも numOfRow & numOfColumn に格納）**するコンストラクタ
- 生成されたベクトル、行列オブジェクトの内容を表示するメソッドを作成すること（条件： ①メソッド名は display とする、② 引数の設定は任意、③ 必要に応じて複数作成しても（オーバーロードしても）良い（複数作成する場合、それぞれがどのような状況での使用に対応しているのか、コメントを記述すること）
- 行列の行数、列数を返すメソッド public int getNumOfRow() と public int getNumOfColumn() を作成すること
- 行列、もしくはベクトルの任意の(row行、column列目の）要素を戻り値として出力するメソッド public double getComponentOf(int rowIndex, int columnIndex) を作成すること。存在しない要素を指定された際には**エラーメッセージを出力して終了（System.exit(0)）**させる

※ サンプルソースは、Teamsのファイル、第3回課題のフォルダ内に“pr3rd_SampleProgram.docx”として用意しているので活用してください。

【注意1】　次回以降、行列（ベクトル）に関する表現・処理は、他の課題達成のために利用する（場合によっては拡張の可能性もある）。どのようにコーディングすれば、今後再利用、拡張しやすいかを考えたうえで方針を各自決定すること。

３－２　ベクトル、行列の積の計算

上で作成したクラスの中に、ベクトルや行列の積を計算するメソッドを追加せよ。追加するメソッドの仕様は以下の通り。

- 二つのベクトルの内積を戻り値とするメソッド **public double getInnerProduct(引数は各自で設定)** を作成する。以下の条件に従って作成すること
  - **Matrix インスタンス A とMatrix インスタンス B の内積（A・B）をとる場合、A.getInnerProduct（引数）という形式で実行することで計算できるよう実装する**
  - 行ベクトルと列ベクトルの内積、行ベクトルと行ベクトルの内積のみ計算可能とする
  - A が列ベクトルの場合、何らかのエラーメッセージを表示させて終了（System.exit(0)）させる（メッセージは、エラーの内容がはっきり分かるような内容とすること）
  - 二つのベクトルの次元数（要素数）が等しくない or 計算対象がベクトルではない（行列である）場合は、 上記 3. とは異なるエラーメッセージを表示させて終了（System.exit(0)）させる（メッセージは、エラーの内容がはっきり分かるような内容とすること）
- 双方ともMatrix型の行列同士の積、行列と列ベクトル（および列ベクトルと行ベクトル）の積を計算し、結果をMatrix型の戻り値として返すメソッド public **Matrix multiplyMatrix(Matrix target)** を作成すること
- メソッド multiplyMatrix を実行する前処理として、2つの行列（行列とベクトル） x と y の乗算処理が実行可能であるかどうかを判定するメソッド public **boolean multipliable(Matrix y)** を作成すること（乗算可能な場合には true、不可能な場合には false を返すこととする）。引数をMatrix x, Matrix yとする static なメソッドとして作成しても良い
- サンプルソースは、３－１で示したものを活用すること
- 作成したメソッドを利用して、以下の(1)～(5)の計算処理を実行するコードを main メソッド中に記述し、答えに間違いがないか確認すること（**教員もその結果が確認できるよう、必ずコンソールに結果を表示させること ＆ 個々の数値がどの問題の答えに対応しているのか，一目見て分かるよう，改行を入れたり問題番号も表示させるなどして工夫すること**）

※ ３－１と３－２の結果は別々のプログラムとせず、３－２まで作成した結果を課題プログラムとして提出してください（３－２ができていれば必然的に３－１はできていることになる）。

(1) $\vec{x} = (2, -3, 7),\ \vec{y} = (-1, -2, 2)$の内積

(2) $\begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & -1 \\ 4 & 2 & 6 \end{pmatrix}\begin{pmatrix} 5 & 3 & 1 \\ 3 & -3 & 2 \end{pmatrix}$

(3) $\begin{pmatrix} 3 \\ -2 \end{pmatrix}\begin{pmatrix} 3 & 7 & -5 & 2 \end{pmatrix}$

(4) $\begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & -1 \\ 4 & 2 & 6 \end{pmatrix}\begin{pmatrix} 8 & 2 \\ -3 & 2 \\ 1 & 6 \end{pmatrix}$

(5) $\begin{pmatrix} 2 & -3 \\ 4 & 2 \end{pmatrix}\begin{pmatrix} -5 & -3 & 1 \\ -3 & 3 & 2 \end{pmatrix}$

【実行例】

```
v0 =
[2.0 -3.0 7.0]

m1 =
[3.0 2.0]
[-3.0 2.0]
[1.0 6.0]

v1 =
[3.0]
[-2.0]

例題0a
v0 * m1 =
[22.0 40.0]

例題0b
v0 * v1 =
要素数が計算できる組み合わせとなっていません.
```