## 패키지 구성

src

- tensor
    - Scalar.java (public interface)
    - Vector.java (public interface)
    - Matrix.java (public interface)
    - ScalarImpl.java (디폴트 class ScalarImpl implements Scalar)
    - VectorImpl.java (디폴트 class VectorImpl implements Vector)
    - MatrixImpl.java (디폴트 class MatrixImpl implements Matrix)
    - Factory.java (public class)
    - Tensors.java (public class)
    - XXXException.java (public class) (여러개)
- test
    - Test.java

## Test.java 요구사항

· Factory를 이용하여 스칼라, 벡터, 행렬 객체를 얻어낸다.

· 스칼라, 벡터, 행렬 객체의 참조 타입은 인터페이스 타입만을 사용한다.

- 즉, 소스에서 구현 클래스(ScalarImpl, VectorImpl, MatrixImpl)는 등장하지 않는다.

## Factory.java 요구사항

· 행렬, 벡터, 스칼라를 생성한다.

· 모든 메소드가 public static이다.

· 생성 자체는 구현 클래스 내부에서 담당한다.

- 즉, Factory의 static 메소드들은 스칼라, 벡터, 행렬 구현 클래스에 객체 생성을 요청한 후, 획득한 객체를 반환만 해주면 된다.

## 스칼라의 자료구조

· 스칼라는 java.lang.Double 타입 객체 하나를 갖는다.

## 벡터의 자료구조

· 벡터는 논리적으로 스칼라 객체를 1차원 배열 구조로 관리한다.

- 물리적으로는 배열 혹은 Collection 중에서 선택 가능하다.

## 행렬의 자료구조

· 행렬은 논리적으로 스칼라 객체를 2차원 배열 구조로 관리한다.

- 물리적으로는 배열 혹은 Collection 중에서 선택 가능하다.

스칼라의 생성 (default 접근 지정자 사용)

01. 값(double 혹은 Double)을 지정하여 스칼라를 생성할 수 있다.
02. i 이상 j 미만의 무작위 값을 요소로 하는 스칼라를 생성할 수 있다.

벡터의 생성 (default 접근 지정자 사용)

03. 지정한 하나의 값을 모든 요소의 값으로 하는 n-차원 벡터를 생성할 수 있다.
04. i 이상 j 미만의 무작위 값을 요소로 하는 n-차원 벡터를 생성할 수 있다.
05. 1차원 배열로부터 n-차원 벡터를 생성할 수 있다.

행렬의 생성 (default 접근 지정자 사용)

06. 지정한 하나의 값을 모든 요소의 값으로 하는 m x n 행렬을 생성할 수 있다.
07. i 이상 j 미만의 무작위 값을 요소로 하는 m x n 행렬을 생성할 수 있다.
08. csv 파일로부터 m x n 행렬을 생성할 수 있다.
    · csv 파일 형식: comma(,)는 열 구분자, 라인은 행 구분자
    2 x 4 행렬 예) 1, 2, 3, 4,
                  5, 6, 7, 8
09. 2차원 배열로부터 m x n 행렬을 생성할 수 있다.
10. 단위 행렬을 생성할 수 있다.

스칼라, 벡터, 행렬의 기본 기능

11v, 11m. (only 벡터, 행렬) 특정 위치의 요소를 지정/조회할 수 있다. (스칼라로 입출력)
12. (only 스칼라) 값을 지정/조회할 수 있다. (Double로 입출력, 오토박싱 가능(double로 입력가능))

13v, 13m. (only 벡터, 행렬) 크기 정보를 조회할 수 있다. (벡터는 차원, 행렬은 행개수, 열개수)

· (@Override toString()) 객체를 콘솔에 출력할 수 있다.
    - 14s. 스칼라: double 값 하나
    - 14v. 벡터: double 값들을 1차원 배열 모양으로
    - 14m. 행렬: double 값들을 2차원 배열 모양으로

15s,v,m. (@Override equals()) 객체의 동등성 판단을 할 수 있다.
    - 크기와 값이 같으면 true

16 (implements Comparable) 스칼라의 경우 값의 대소 비교를 할 수 있다.
    - 스칼라에서만 구현 필요.

17s,v,m. (@Override clone()) 객체 복제를 할 수 있다.
    - deep copy

스칼라의 연산 (non-static 메소드로 구현) ※ 연산 결과는 자신의 새로운 값이 된다.

18. 스칼라는 다른 스칼라와 덧셈이 가능하다.
19. 스칼라는 다른 스칼라와 곱셈이 가능하다.

벡터의 연산 (non-static 메소드로 구현) ※ 연산 결과는 자신의 새로운 값이 된다.

20. 벡터는 다른 벡터와 덧셈이 가능하다. (길이가 같을 때)
21. 벡터는 다른 스칼라와 곱셈이 가능하다. (벡터의 모든 요소에 스칼라를 곱한다.)

행렬의 연산 (non-static 메소드로 구현) ※ 연산 결과는 자신의 새로운 값이 된다.

22. 행렬은 다른 행렬과 덧셈이 가능하다. (크기가 같을 때)
23. 행렬은 다른 행렬과 곱셈이 가능하다. ((mxn)x(nxl)일 때)
    - 다른 행렬이 왼쪽 행렬로서 곱해지는 경우와 오른쪽 행렬로서 곱해지는 경우 모두 지원

스칼라의 연산 (디폴트 static 메소드로 구현) ※ 연산 결과 스칼라는 새로 생성되어 반환

24. 전달받은 두 스칼라의 덧셈이 가능하다.
25. 전달받은 두 스칼라의 곱셈이 가능하다.

벡터의 연산 (디폴트 static 메소드로 구현) ※ 연산 결과 벡터는 새로 생성되어 반환

26. 전달받은 두 벡터의 덧셈이 가능하다. (길이가 같을 때)
27. 전달받은 스칼라와 벡터의 곱셈이 가능하다. (벡터의 모든 요소에 스칼라를 곱한다.)

행렬의 연산 (디폴트 static 메소드로 구현) ※ 연산 결과 행렬은 새로 생성되어 반환

28. 전달받은 두 행렬의 덧셈이 가능하다. (크기가 같을 때)
29. 전달받은 두 행렬의 곱셈이 가능하다. ((mxn)x(nxl)일 때)

Tensors.java 요구사항

· 모든 메소드가 public static이다.

  - 위의 디폴트 static 메소드로 구현하라고 한 스칼라, 벡터, 행렬의 연산 기능을 호출한 후 반환된 객체를 반환하는 메소드들이다.
  - Tensors의 사용자는 Test.java로 상정하면 된다.
  - 명세한 것 외에도 이와 같은 방식으로 사용될 메소드들을 Tensors 클래스에 추가 가능하다.

## 벡터의 고급 기능

30. n-차원 벡터 객체는 자신으로부터 nx1 행렬을 생성하여 반환할 수 있다.
31. n-차원 벡터 객체는 자신으로부터 1xn 행렬을 생성하여 반환할 수 있다.

※ 위의 두 기능은 행렬x벡터 연산을 행렬x행렬 연산으로 수행하기 위함이다.

## 행렬의 고급 기능

32. 행렬은 다른 행렬과 가로로 합쳐질 수 있다. (두 행렬의 행 수가 같아야 가능)
33. 행렬은 다른 행렬과 세로로 합쳐질 수 있다. (두 행렬의 열 수가 같아야 가능)

※ 위의 두 기능은 디폴트 static 메소드로도 구현하라.

- 즉, 두 행렬을 전달받아 합쳐진 새로운 행렬을 반환하는 형식
- Tensors에서 호출할 수 있도록 하라.

34. 행렬은 특정 행을 벡터 형태로 추출해 줄 수 있다. (행 벡터 추출)
35. 행렬은 특정 열을 벡터 형태로 추출해 줄 수 있다. (열 벡터 추출)

36. 행렬은 특정 범위의 부분 행렬을 추출해 줄 수 있다.
    - 시작과 끝 행 인덱스, 시작과 끝 열 인덱스를 사용하여 범위 지정

37. 행렬은 특정 범위의 부분 행렬을 추출해 줄 수 있다.
    - 특정 하나의 행과 하나의 열을 제외한 부분 행렬 (minor)
    - 예) 1 2 3 4 에서 2행 2열을 제외한 minor는 1 3 4 이다. (코드 기준으로는 1행 1열을 제외)

      5 6 7 8 → 5 7 8

      9 3 4 1 → 9 4 1

38. 행렬은 전치행렬을 (새로 생성하여) 구해줄 수 있다. (transpose)

39. 행렬은 대각 요소의 합을 구해줄 수 있다. (nxn 행렬) (trace)

40. 행렬은 자신이 정사각 행렬인지 여부를 판별해 줄 수 있다.
41. 행렬은 자신이 상삼각 행렬인지 여부를 판별해 줄 수 있다. (nxn 행렬) (upper triangular matrix)
42. 행렬은 자신이 하삼각 행렬인지 여부를 판별해 줄 수 있다. (nxn 행렬) (lower triangular matrix)
43. 행렬은 자신이 단위 행렬인지 여부를 판별해 줄 수 있다. (nxn 행렬) (identity matrix)
44. 행렬은 자신이 영 행렬인지 여부를 판별해 줄 수 있다. (zero matrix)

45. 행렬은 특정 두 행의 위치를 맞교환할 수 있다.
46. 행렬은 특정 두 열의 위치를 맞교환할 수 있다.

47. 행렬은 특정 행에 상수배(스칼라)를 할 수 있다.
48. 행렬은 특정 열에 상수배(스칼라)를 할 수 있다.

49. 행렬은 특정 행에 다른 행의 상수배를 더할 수 있다.
50. 행렬은 특정 열에 다른 열의 상수배를 더할 수 있다.

※ 위의 여섯 개의 기능은 elementary row operations에 해당한다.

51. 행렬은 자신으로부터 RREF 행렬을 구해서 반환해 줄 수 있다. (row reduced echelon form)
52. 행렬은 자신이 RREF 행렬인지 여부를 판별해 줄 수 있다.

53. 행렬은 자신의 행렬식을 구해줄 수 있다. (nxn 행렬) (determinant)

54. 행렬은 자신의 역행렬을 구해줄 수 있다. (nxn 행렬)

### 예외 클래스 정의

· 위의 기능을 구현하다보면 다양한 예외상황이 발생할 것이다. 이 경우 예외 클래스를 정의하여 예외 발생 및 처리하는 형태로 구현하라.

· 예외 클래스들은 tensor 패키지에 두어라.

### 과제 점검을 위한 요구 사항

· Test.java는 위의 요구사항들을 모두 만족시켰음을 보이도록 작성하라.

  - Test.java의 출력 화면만을 보고 채점할 수 있도록 작성하라.

· Test.java는 다른 팀원의 Test.java로도 테스트하여 인터페이스 설계가 잘 공유되었는지도 점검할 것이다.

  - 팀마다 tensor 패키지의 인터페이스 파일, Tensors와 Factory 클래스의 정적 메소드 선언부(메소드 바디{...} 제외), 예외 클래스명은 공유해야 한다.

    - 결국 tensor 패키지의 메소드 구현은 개별적으로 수행할 수 있다.

· 벡터와 행렬의 내부 자료구조를 배열기반으로 구현했을 경우 B+가 만점, Collection 기반으로 구현했을 경우 A+가 만점

### 과제 수행을 위한 추천 사항

· 과제 명세서 순서로 구현하는 것을 추천

  - 이 순서를 따르면 앞서 구현한 기능을 나중에 부품으로 사용할 수 있다.

· 과제 명세서는 최소한의 명세로 간주하고 작업 도중 도출된 필요한 기능은 자유롭게 추가하라.

· 학생의 작업 편의를 고려하여 명세서가 다소 촘촘하게 구성하였다. 그러나, 채점은 다소 관대하게 할 예정이다. 그러니, 처음에는 막막하겠으나 머리를 모아 수행해야 할 내용을 정리한 후 자유롭게, 주도적으로 과제를 수행하라.