SE/CS 2S03: Principles of Programming

Due on November 15th

Dr. Jacques Carette

Idea

The goals of this assignment are:

1. get some understanding of records and arrays
2. write code+tests
3. learn how to work your editor to do block copy and block edits

The Task

This assignment involves writing several short routines. They will involve writing several classes. For all matrices, use `long` as the representation of the integer type (i.e. all matrices will be matrices of integers).

1. Create a `Matrix3x3flat` class which implements a 3×3 matrix using a single record with 9 fields.
2. Create a `Matrix3x3rc` class which implements a 3×3 matrix using a record of 3 rows, each containing a record of 3 columns.
3. Create a `Matrix3x3cr` class which implements a 3×3 matrix using a record of 3 columns, each containing a record of 3 rows.
4. Create a `MatrixArrayFlat` class which implements a $n \times n$ matrix using a 1D `Array`.
5. Create a `MatrixArrayRC` class which implements a $n \times n$ matrix using an `Array` of rows of `Array`s of columns.
6. Create a `MatrixArrayCR` class which implements a $n \times n$ matrix using an `Array` of columns of `Array`s of rows.

For each of these 6,

1. implement a constructor which takes as input a single `Array` with 9 elements (and fails otherwise) to fill things in.
2. implement a `determinant` method (which takes **no** arguments) which (obviously!) computes the determinant of the matrix.

You will also need to:

- in a new (testing) class, create yourself 10 3×3 matrices. Make sure some are degenerate, and some are not.
- call each of the 6 methods on all 10 matrices and verify that they give the right answer (i.e. using JUnit).

- call each pair of methods on all 10 matrices and make sure that each routine pairwise give the same answer.

Notes:

- Yes, that means 60 + 360 test cases. Learn how to use your editor properly so that you do NOT have to type this in! See bonus section below as well.
- Yes, the code in the first 3 versions will look alike a lot, and yet be subtly different. That's part of the learning objective of this assignment; even though this is clear 'in theory', seeing it (and doing it) in practice is quite enlightening.
- The same is true for the next 3 versions as well.
- You *may* look up how to compute an $n \times n$ determinant in Java in a textbook or online – just **cite** your sources!! If you do **NOT** cite your sources, you will be marked as having **plagiarized** your assignment.

Submission Requirements

- A *single* zip file containing all your java files, including your JUnit test files.

Marking Scheme

- Programs which do not compile will be given a mark of 0, no matter how *close* your code might be to the correct answer.
- The code will be worth 60%, the tests 40%.

Bonus

Each one of these will be worth extra marks:

- (easy) Implement your matrices using `BigInteger` instead of long
- (moderate) Find a way to iterate over each of the classes, so that the tests contain single loop over 6 classes, and another over 10 arrays to do the first 60 tests. Write 3 loops in the same vein for the next 360 tests.
- (hard) Implement your matrices using a *generic ring type* instead of `long`.
- (very challenging) Implement a generator (in any language, but more marks for Haskell/Scala) for this assignment which has a single method (i.e. there should not be 6 cases) that is parametrized by the choices (i.e. record/Array, flat/rc/cr, and dimension). Note it is significantly easier to have this generator take the dimension as a parameter [i.e. in theory you could generate record-based code for 10x10 matrices!]. Make sure your code works properly for degenerate (i.e. 1x1) matrices too. Using an AST (rather than strings) is definitely preferred. This bonus is not due until **Dec. 6th**. Basically, if you manage to do this, you'll pass the course even if you don't even write the final. If you've done even moderately well on things up to now, you're guaranteed an A+. Actually, if you do this, I'll probably try to hire you as a summer research assistant!

Yes, you may do many of these bonus parts together.