MA117 Project 0: Manipulating Fractions

Administrative Details

- This assignment is **unassessed**. However, you **must** submit your solution code to both check that the automated marking system is working for you, and for practice with classes and objects.
- The automated submission system requires that you closely follow instructions about the format of certain files; failure of doing in the coming assessed assignments will result in the severe loss of points.
- You may work on the assignment during the lab session, provided you have completed the other tasks that have been set. You can use the work areas when they are not booked for teaching. If you are working on the assignment on your home system you are advised to **make regular back-up** copies (for example by transferring the files to the University systems). You should note that **no** allowance will be made for domestic disasters involving your own computer system. You should make sure well ahead of the deadline that you are able to transfer all necessary files to the University system and that it works on *bell.warwick.ac.uk*.
- Remember that all work you submit should be **your own work**. Do not be tempted to copy work; this assignment is not meant to be a team exercise. There are both human and automated techniques to detect pieces of the code which have been copied from others. If you are stuck, then ask for assistance in the lab sessions. TAs will not complete the exercise for you, but they will help if you do not understand the problem, are confused by an error message, need advice on how to debug the code, require further explanation of a feature of Java or similar matters.
- If you have more general or administrative problems *e-mail me* immediately. Always include the course number (MA117) in the subject of your e-mail.

1 Introduction

This small project is designed to get you started to the world of object-orientated programming in Java. You will work on development of this class probably for one or two lab sessions, but it is another typical introductory example of using objects/classes. You should really implement it yourself, although you will find it in almost any textbook on Java. It will help you to practice the type of thinking typical for Java and other object-oriented languages.

We want to develop a "fraction calculator" which can perform exact arithmetic operations with rational numbers. This means that the numbers are represented by fractions a/b where a and b are integers and they are not actually converted to the corresponding decimal representation.

You shall design a new class called `Fraction`, instances of which will represent mathematical fractions (e.g. ½ will be represented by two integers 1 and 2; **not** by 0.5). The class should also contain several methods which implement different operations and manipulations we usually do with fractions - e.g. addition, subtraction, etc; again without converting them to the decimal representation.

2 Instructions

Firstly, go to the project moodle page and download the files `Project0.java, Fraction.java`. These files are **skeleton** files – essentially, just like the lecture notes, some parts of the class are blanked out. Each method is already "preprogrammed" so the interface (the **functions**, their **parameters** and what should be **returned**) is **fixed** and you **must not** change it. Every function has a description, which outlines:

- The purpose of the function, and intended use;
- The parameters it accepts (if any) indicated by `@param`;
- What it returns (if anything) indicated by `@return`.

BOSS – the automated marking system – will expect this interface to remain unchanged. So, if at any point you redefine the name of a `public` method, the system will be unable to see this method and you would be awarded 0 for that method.

Your task is to complete the missing functions by writing Java code which will perform the current functionality. Along this line, you should do the following:

1. **Carefully** read through the class definition and the instructions at the top of `Project0.java` and `Fraction.java`.
2. Define the methods in `Fraction.java` that are not implemented. They are indicated by a comment reading "Fill in this method", but for reference, they are:
   - The `Fraction(m,n)` constructor.
   - Convertors: `toDouble()`, `toString()`.
   - Operations with fractions: `add()`, `divide()` and `multiply()`
3. Then add two methods to the class using the names below, make them are `private` and accept **no** parameters:
   - `private int gcd()`: computes and **returns** the greatest common divisor of the numerator and denominator for this fraction.
   - `private void reduce()`: uses `gcd()` to reduce the fraction to irreducible form. e.g. the fraction $10/20$ would be simplified to $1/2$.
4. Using these methods, you should then ensure that any fraction that is created will **always** be stored in irreducible form.
5. Make some tests for your solution using the `main()` method within `Project0.java`. NOTE: BOSS will not use `main()` since it performs its own tests on the `Fraction` class.

3 Submitting to Moodle

Once you've written the `Fraction` class, ensure you have tested it on *bell.warwick.ac.uk*. You need to submit your tested `Project0.java` and `Fraction.java` files to Moodle, the online submission system. The first time you do this can be a little confusing (hence why we have project 0 in the first place!)