INTRODUCTION

For this assignment, you're going to create a new class named "Fraction" for managing rational numbers (fractions). (**Do not submit a main program with the Fraction class**; however, you'll definitely want to write one (or more) for your own testing purposes.) Details are as follows:

CLASS DEFINITION

CONSTRUCTORS:

Fraction(int num,int denom)	creates a new number whose value is num/denom
Fraction(int num)	creates a new number whose value is num/1

METHODS (assume x is a Fraction):

Fraction x.add(Fraction n)	Returns x+n in reduced form
Fraction x.sub(Fraction n)	Returns x-n in reduced form
Fraction x.mul(Fraction n)	Returns x*n in reduced form
Fraction x.div(Fraction n)	Returns x/n in reduced form
double x.toDouble()	Returns x as a double-precision floating point number
int getNum(),int getDenom()	Returns numerator or denominator
String toString()	Return a string representing the fraction, as described below

(OPTIONAL) You may also want to make a private method:

private x.reduce()	Adjusts num and denom so num/denom is in simplest form (as described below)

DETAILS

We'll discuss in class how to operate on rational numbers: addition, multiplication, etc. For reducing a fraction to simplest form, we'll use *Euclid's Algorithm*, which will also be reviewed in class. **You must write this yourself**: don't use a built-in GCD function.

DIVISION BY 0

Fractions with denominators of 0 are arithmetically undefined, so if such a number is created, you should somehow note internally that the number is undefined. **Note**

that any operations involving a undefined number result in an undefined number.

If toDouble() is used on an undefined number, return Double.NaN

ToString

You should create a method String toString() for treating Fractions as a String. A number of the form D/1 should convert as simply D (no “/1”). Care should be used with rationals with a negative num or a negative denom. For example, if num=1 and denom=-2 you should return -1/2 (not 1/-2); If num=-1 and denom=-2, you should return 1/2 (not -1/-2). An undefined number should return"NaN"

REDUCING NUMBERS

“Simplest form” means (a) eliminating any common integer factors from both num and denom; (b) adjusting num and denom so denom is not negative; and (c) changing 0/x to the form 0/1 (unless x=0, in which case this results in NaN).

GRADING

This assignment will be worth 100 points, and will be graded as follows:

Functional Behavior: 85 points

Perfect: 85 points
Most functions implemented, but some errors or glitches: 70 points
Some functions work perfectly, but many are unimplemented or barely present: 50 points
Most functions inoperable, but basic pieces are in place: 30 points
Very little usable code, or nothing submitted: 0 points

Documentation: 10 points

Header block present; code is cleanly written; each method is described in detail; non-obvious lines are commented: 10 points
Header block present, some useful comments but more needed: 7 points
Very little commenting, header block present: 3 points
No header block: 0 points

Submission: 5 points

I want to make sure you can follow submission instructions etc., so 5 points if your .tar file is correct. **If it deviates from the specification, you may lose up to 5 points.**

TESTING

Testing of your Fraction class is up to you, but you should do some tests in a main program to confirm the desired behaviors. Here is some code I may use to test your class:

```
public class FractionTest {
  public static void main(String[] args) {
```

```
    Fraction a=new Fraction(1,2); // a=1/2
    Fraction b=new Fraction(4); // b=4
    Fraction c,d,e;
    System.out.println("5/2: "+new Fraction(125,50));
    c=a.add(b);
    System.out.println("1: " + new Fraction(1,2).mul(new Fraction(2)));
    System.out.println("9/2: "+c);
    c=a.sub(b);
    System.out.println("-7/2: "+c);
    c=b.sub(a);
    System.out.println("7/2: "+c);
    c=a.mul(b);
    System.out.println("2: "+c);
    System.out.println("4: "+c.mul(c));
    d=new Fraction(3); // d=3
    e=new Fraction(4); // e=4
    c=d.div(e);
    System.out.println("3/4: "+c);
    a=e.div(d);
    System.out.println("4/3: "+a);
    c=a.mul(b); // c=16/3
    System.out.println("16/3: "+c);
    System.out.println("1/3=0.33333...:"+ new Fraction(1,3).toDouble());
    System.out.println("1/0 (NaN): "+new Fraction(1,0));
    a=new Fraction(1,2); // a=1/2
    b=new Fraction(1,2); // b=1/2
    c=a.sub(b); // c=0
    a=a.div(c); // a=1/0
    System.out.println("NaN: "+a.toDouble());
    a=c.div(c); // a=0/0
    System.out.println("NaN: "+a.toDouble());
    System.out.println("NaN: "+new Fraction(-1,0).toDouble());
  }
}
```

SUBMISSION

Submit your assignment as a single pa1.tar file containing your Fraction.java file and any other files you feel like you need to include. This assignment can definitely be done with a single class (Fraction.java), but if you wish to define other classes, feel free to do so. **Do not include a main method**.

Also, for those who are prone to fancier things:

- do not use a package statement
- put all your .java files in one directory
- create the .tar file from the directory containing your .java files, i.e.
  cd dir;tar cvf pa1.tar *.java
  as opposed to tar cvf pa1.tar deepdir/subdir/sourcedir/otherpaths/*.java
- do not include .class files in your .tar file

Make sure the methods described above are implemented as described, since I'll be using my own code to test their behavior. If you're in doubt, **please ask** in the discussion forum on Canvas.