Arnav Shirodkar

CMSC 201: Data Structures

Lab 2 Report – Fraction ADT (13/02/2020)

Assignment Description: Create an ADT for fractions using the BigInteger class, with the functions as described in the lab handout

Collaboration Statement: I worked on this file with Elliot and Marco, also approaching Michael Ventoso for help with the boolean equals function. I frequently used the javadoc for BigInteger to explore the various functions associated with BigInteger and also looked online for the various functions used for JUnit testing.

<http://junit.sourceforge.net/javadoc/org/junit/Assert.html>

<https://docs.oracle.com/javase/7/docs/api/java/math/BigInteger.html#equal(java.lang.Object>

Fraction.java

import java.math.BigInteger;

**public class Fraction extends Number**{

private BigInteger numerator;

private BigInteger denominator;

**Fraction(BigInteger n, BigInteger d){**

// Constructor that takes 2 big int arguments

if (d.intValue() == 0){

throw new IllegalArgumentException("/ by zero");

}

BigInteger divisor = n.gcd(d); //Simplify

numerator = n.divide(divisor);

denominator = d.divide(divisor);

if(numerator.compareTo(BigInteger.ZERO) == -1 && denominator.compareTo(BigInteger.ZERO) == -1){

//Make positive if numerator and denominator are negative

numerator = numerator.multiply(BigInteger.valueOf(-1));

denominator = denominator.multiply(BigInteger.valueOf(-1));

}

}

**Fraction(int n, int d){** //Constructor for 2 int arguments

if(d == 0){

throw new IllegalArgumentException("/ by zero");

}

BigInteger x = BigInteger.valueOf(n);

BigInteger y = BigInteger.valueOf(d);

BigInteger divisor = x.gcd(y);

numerator = x.divide(divisor);

denominator = y.divide(divisor);

if(numerator.compareTo(BigInteger.ZERO) == -1 && denominator.compareTo(BigInteger.ZERO) == -1){

numerator = numerator.multiply(BigInteger.valueOf(-1));

denominator = denominator.multiply(BigInteger.valueOf(-1));

}

}

**Fraction(int n){** // constructor for 1 int argument

numerator = BigInteger.valueOf(n);

denominator = BigInteger.ONE;

}

**Fraction(BigInteger n){** // constructor for 1 big in argument

numerator = n ;

denominator = BigInteger.ONE;

}

**public BigInteger getNumerator(){**

return numerator;

}

**public BigInteger getDenominator(){**

return denominator;

}

**public Fraction add(Fraction x){**

BigInteger tempNum = numerator;

BigInteger tempDen = denominator;

numerator = (tempNum.multiply(x.getDenominator())).add (tempDen.multiply(x.getNumerator()));

denominator = tempDen.multiply(x.getDenominator());

return this.simplify(); //simplify fraction after the relevant operation

}

**public Fraction subtract(Fraction x){**

BigInteger tempNum = numerator;

BigInteger tempDen = denominator;

numerator = (tempNum.multiply(x.getDenominator())). subtract(tempDen.multiply(x.getNumerator()));

denominator = tempDen.multiply(x.getDenominator());

return this.simplify();

}

**public Fraction multiply(Fraction x){**

numerator = numerator.multiply(x.getNumerator());

denominator = denominator.multiply(x.getDenominator());

return this.simplify();

}

**public Fraction divide(Fraction x){**

numerator = numerator.multiply(x.getDenominator());

denominator = denominator.multiply(x.getNumerator());

return this.simplify();

}

**public String toString(){** // convert to String value "x/y"

if(denominator != BigInteger.ONE){

String s = String.valueOf(numerator) + "/" + String.valueOf(denominator);

return s;

} else {

String s = String.valueOf(numerator);

return s;

}

}

**public long longValue(){** //convert to relevant type

BigInteger x = numerator.divide(denominator);

return x.longValue();

}

**public int intValue(){**

BigInteger x = numerator.divide(denominator);

return x.intValue();

}

**public double doubleValue(){**

double x = numerator.doubleValue();

double y = denominator.doubleValue();

return x/y;

}

**public float floatValue(){**

float x = numerator.floatValue();

float y = denominator.floatValue();

return x/y;

}

**public boolean equals (Object o){** //check if object o equals to this fraction

if (o.getClass() != this.getClass()){

return false;

}

Fraction x = ((Fraction)o); // whatever o is, it attempts to cast o to Fraction and if o is indeed a fraction, test will be true

if(x.getNumerator().equals(numerator) == true && x.getDenominator().equals(denominator) == true){

return true;

} else {

return false;

}

}

**public Fraction simplify(){** // find GCD and simplify the fraction

BigInteger divisor = numerator.gcd(denominator);

numerator = numerator.divide(divisor);

denominator = denominator.divide(divisor);

if(numerator.compareTo(BigInteger.ZERO) == -1 && denominator.compareTo(BigInteger.ZERO) == -1){

numerator = numerator.multiply(BigInteger.valueOf(-1));

denominator = denominator.multiply(BigInteger.valueOf(-1));

return this;

} else if (numerator.compareTo(BigInteger.ZERO )== 0){

return new Fraction(0);

}

return this;

}

**public static Fraction valueOf(int n, int d){** //Used for quick values to avoid the client having to constantly create new Fractions.

return new Fraction(n,d);

}

}

FractionTests.Java

import org.junit.Test;

import static org.junit.Assert.\*;

import java.math.BigInteger;

public class FractionTests{

@Test

public void numeratorTest1(){

Fraction big = new Fraction(99,66);

assertEquals(BigInteger.valueOf(3) , big.getNumerator());

}

@Test

public void numeratorTest2(){

BigInteger mega1 = new BigInteger ("7263847287348274");

BigInteger mega2 = new BigInteger ("22347982948729");

Fraction mega = new Fraction(mega1, mega2);

assertEquals(mega1 , mega.getNumerator());

}

@Test

public void numeratorTest3(){

Fraction big = new Fraction(-99,66);

assertEquals(BigInteger.valueOf(-3) , big.getNumerator());

}

@Test

public void numeratorTest4(){

Fraction big = new Fraction(-99,-66);

assertEquals(BigInteger.valueOf(3) , big.getNumerator());

}

@Test

public void denominatorTest1(){

Fraction big = new Fraction(99,66);

assertEquals(BigInteger.valueOf(2) , big.getDenominator());

}

@Test

public void denominatorTest2(){

BigInteger mega1 = new BigInteger ("7263847287348274");

BigInteger mega2 = new BigInteger ("22347982948729");

Fraction mega = new Fraction(mega1, mega2);

assertEquals(mega2 , mega.getDenominator());

}

@Test

public void denominatorTest3(){

Fraction big = new Fraction(99,-66);

assertEquals(BigInteger.valueOf(-2) , big.getDenominator());

}

@Test

public void denominatorTest4(){

Fraction big = new Fraction(-99,-66);

assertEquals(BigInteger.valueOf(2) , big.getDenominator());

}

@Test

public void gcdTest1(){

Fraction big = new Fraction(99,66);

Fraction small = new Fraction(3,2);

assertEquals(big.getNumerator(), small.getNumerator());

assertEquals(small.getDenominator(), big.getDenominator());

}

@Test

public void gcdTest2(){ //

Fraction big = new Fraction(-99,66);

Fraction small = new Fraction(-3,2);

assertEquals(big.getNumerator(), small.getNumerator());

assertEquals(small.getDenominator(), big.getDenominator());

}

@Test

public void gcdTest3(){

Fraction big = new Fraction(-99,-66);

Fraction small = new Fraction(3,2);

assertEquals(big.getNumerator(), small.getNumerator());

assertEquals(small.getDenominator(), big.getDenominator());

}

@Test

public void gcdTest4(){

BigInteger n = BigInteger.valueOf(-99);

BigInteger x = BigInteger.valueOf(66);

assertEquals(BigInteger.valueOf(33),n.gcd(x)); //GCD always positive

}

@Test

public void intValueTest1(){

Fraction convert = new Fraction(10,8);

int i = 1;

assertEquals(convert.intValue(),i);

}

@Test

public void intValueTest2(){

BigInteger mega1 = new BigInteger ("7263847287348274");

BigInteger mega2 = new BigInteger ("22347982948729");

Fraction mega = new Fraction(mega1, mega2);

int i = 325;

assertEquals(mega.intValue(),i);

}

@Test

public void intValueTest3(){

Fraction convert = new Fraction(-10,8);

int i = -1;

assertEquals(convert.intValue(),i);

}

@Test

public void intValueTest4(){

Fraction convert = new Fraction(-10,-8);

int i = 1;

assertEquals(convert.intValue(),i);

}

@Test

public void LongValueTest1(){

Fraction convert = new Fraction(10,8);

long i = 1;

assertEquals(convert.longValue(),i);

}

@Test

public void LongValueTest2(){

BigInteger mega1 = new BigInteger ("7263847287348274");

BigInteger mega2 = new BigInteger ("22347982948729");

Fraction mega = new Fraction(mega1, mega2);

long i = 325;

assertEquals(mega.longValue(),i);

}

@Test

public void LongValueTest3(){

Fraction convert = new Fraction(-10,8);

long i = -1;

assertEquals(convert.longValue(),i);

}

@Test

public void LongValueTest4(){

Fraction convert = new Fraction(-10,-8);

long i = 1;

assertEquals(convert.longValue(),i);

}

@Test

public void doubleValueTest1(){

Fraction convert = new Fraction(10,8);

double i = 1.25;

assertEquals(convert.doubleValue(),i,0.0001);

}

@Test

public void doubleValueTest2(){

BigInteger mega1 = new BigInteger ("7263847287348274");

BigInteger mega2 = new BigInteger ("22347982948729");

Fraction mega = new Fraction(mega1, mega2);

double i = 325.03368;

assertEquals(mega.doubleValue(),i,0.0001);

}

@Test

public void doubleValueTest3(){

Fraction convert = new Fraction(-10,8);

double i = -1.25;

assertEquals(convert.doubleValue(),i,0.0001);

}

@Test

public void doubleValueTest4(){

Fraction convert = new Fraction(-10,-8);

double i = 1.25;

assertEquals(convert.doubleValue(),i,0.0001);

}

@Test

public void floatValueTest1(){

Fraction convert = new Fraction(10,8);

float i = 1.25f;

assertEquals(convert.floatValue(),i,0.0001);

}

@Test

public void floatValueTest2(){

BigInteger mega1 = new BigInteger ("7263847287348274");

BigInteger mega2 = new BigInteger ("22347982948729");

Fraction mega = new Fraction(mega1, mega2);

float i = 325.03368f;

assertEquals(mega.floatValue(),i,0.0001);

}

@Test

public void floatValueTest3(){

Fraction convert = new Fraction(-10,8);

float i = -1.25f;

assertEquals(convert.floatValue(),i,0.0001);

}

@Test

public void floatValueTest4(){

Fraction convert = new Fraction(-10,-8);

float i = 1.25f;

assertEquals(convert.floatValue(),i,0.0001);

}

@Test

public void equalityTest1(){

Fraction big = new Fraction(99,66);

Fraction small = new Fraction(3,2);

assertTrue(big.equals(small));

}

@Test

public void equalityTest2(){

BigInteger mega1 = new BigInteger ("748746501982");

BigInteger mega2 = new BigInteger ("374373250991");

Fraction mega = new Fraction(mega1, mega2);

Fraction small = new Fraction(2);

assertTrue(mega.equals(small));

}

@Test

public void equalityTest3(){

BigInteger mega1 = new BigInteger ("-748746501982");

BigInteger mega2 = new BigInteger ("374373250991");

Fraction mega = new Fraction(mega1, mega2);

Fraction small = new Fraction(-2);

assertTrue(mega.equals(small));

}

@Test

public void equalityTest4(){

BigInteger mega1 = new BigInteger ("-748746501982");

BigInteger mega2 = new BigInteger ("-374373250991");

Fraction mega = new Fraction(mega1, mega2);

Fraction small = new Fraction(2);

assertTrue(mega.equals(small));

}

/\* @Test // To test and make sure assertTrue can fail

public void equalityTestPrime(){

Fraction big = new Fraction(99,66);

BigInteger small = BigInteger.valueOf(1); OR

Fraction small = new Fraction (4,2);

assertTrue(big.equals(small));

} \*/

@Test

public void additionTest1(){

Fraction big = new Fraction(99,66);

Fraction small = new Fraction(3,2);

Fraction sum = new Fraction(3,1);

assertTrue(sum.equals(big.add(small)));

}

@Test

public void additionTest2(){

BigInteger mega1 = new BigInteger ("748746501982");

BigInteger mega2 = new BigInteger ("374373250991");

Fraction mega = new Fraction(mega1, mega2);

BigInteger big1 = new BigInteger("99999999999999");

BigInteger big2 = new BigInteger("33333333333333");

Fraction big = new Fraction(big1,big2);

Fraction sum = new Fraction(5,1);

assertTrue(sum.equals(mega.add(big)));

}

@Test

public void additionTest3(){

BigInteger mega1 = new BigInteger ("-748746501982");

BigInteger mega2 = new BigInteger ("374373250991");

Fraction mega = new Fraction(mega1, mega2);

BigInteger big1 = new BigInteger("99999999999999");

BigInteger big2 = new BigInteger("33333333333333");

Fraction big = new Fraction(big1,big2);

Fraction sum = new Fraction(1);

assertTrue(sum.equals(mega.add(big)));

}

@Test

public void additionTest4(){

BigInteger mega1 = new BigInteger ("-748746501982");

BigInteger mega2 = new BigInteger ("374373250991");

Fraction mega = new Fraction(mega1, mega2);

BigInteger big1 = new BigInteger("-99999999999999");

BigInteger big2 = new BigInteger("33333333333333");

Fraction big = new Fraction(big1,big2);

Fraction sum = new Fraction(BigInteger.valueOf(-5),BigInteger.ONE);

assertTrue(sum.equals(mega.add(big)));

}

@Test

public void subtractTest1(){

Fraction big = new Fraction(4,4);

Fraction small = new Fraction(1,4);

Fraction subtract = new Fraction(3,4);

assertTrue(subtract.equals(big.subtract(small)));

}

@Test

public void subtractTest2(){

BigInteger mega1 = new BigInteger ("748746501982");

BigInteger mega2 = new BigInteger ("374373250991");

Fraction mega = new Fraction(mega1, mega2);

BigInteger big1 = new BigInteger("99999999999999");

BigInteger big2 = new BigInteger("33333333333333");

Fraction big = new Fraction(big1,big2);

Fraction sub = new Fraction(BigInteger.ONE,BigInteger.ONE);

assertTrue(sub.equals(big.subtract(mega)));

}

@Test

public void subtractTest3(){

BigInteger mega1 = new BigInteger ("-748746501982");

BigInteger mega2 = new BigInteger ("374373250991");

Fraction mega = new Fraction(mega1, mega2);

BigInteger big1 = new BigInteger("99999999999999");

BigInteger big2 = new BigInteger("33333333333333");

Fraction big = new Fraction(big1,big2);

Fraction sub = new Fraction(BigInteger.valueOf(5),BigInteger.ONE);

assertTrue(sub.equals(big.subtract(mega)));

}

@Test

public void subtractTest4(){

BigInteger mega1 = new BigInteger ("-748746501982");

BigInteger mega2 = new BigInteger ("374373250991");

Fraction mega = new Fraction(mega1, mega2);

BigInteger big1 = new BigInteger("-99999999999999");

BigInteger big2 = new BigInteger("33333333333333");

Fraction big = new Fraction(big1,big2);

Fraction sub = new Fraction(BigInteger.valueOf(-1),BigInteger.ONE);

assertTrue(sub.equals(big.subtract(mega)));

}

@Test

public void subtractTest5(){

Fraction big = new Fraction(-4,-4);

Fraction small = new Fraction(-1,-4);

Fraction sub = new Fraction(3,4);

assertTrue(sub.equals(big.subtract(small)));

}

@Test

public void productTest1(){

Fraction big = new Fraction(99,66);

Fraction small = new Fraction(3,2);

Fraction product = new Fraction(9,4);

assertTrue(product.equals(big.multiply(small)));

}

@Test

public void productTest2(){

Fraction big = new Fraction(-99,66);

Fraction small = new Fraction(3,2);

Fraction product = new Fraction(-9,4);

assertTrue(product.equals(big.multiply(small)));

}

@Test

public void productTest3(){

BigInteger mega1 = new BigInteger ("-748746501982");

BigInteger mega2 = new BigInteger ("374373250991");

Fraction mega = new Fraction(mega1, mega2);

BigInteger big1 = new BigInteger("-99999999999999");

BigInteger big2 = new BigInteger("33333333333333");

Fraction big = new Fraction(big1,big2);

Fraction product = new Fraction(BigInteger.valueOf(6),BigInteger.ONE);

assertTrue(product.equals(mega.multiply(big)));

}

@Test

public void productTest4(){

BigInteger mega1 = new BigInteger ("-748746501982");

BigInteger mega2 = new BigInteger ("-374373250991");

Fraction mega = new Fraction(mega1, mega2);

BigInteger big1 = new BigInteger("-33333333333333");

BigInteger big2 = new BigInteger("99999999999999");

Fraction big = new Fraction(big1,big2);

Fraction product = new Fraction(BigInteger.valueOf(-2),BigInteger.valueOf(3));

assertTrue(product.equals(mega.multiply(big)));

}

@Test

public void divideTest1(){

Fraction big = new Fraction(198,66);

Fraction small = new Fraction(3,2);

Fraction remainder = new Fraction(2,1);

assertTrue(remainder.equals(big.divide(small)));

}

@Test

public void divideTest2(){

BigInteger mega1 = new BigInteger ("748746501982");

BigInteger mega2 = new BigInteger ("374373250991");

Fraction mega = new Fraction(mega1, mega2);

BigInteger big1 = new BigInteger("33333333333333");

BigInteger big2 = new BigInteger("99999999999999");

Fraction big = new Fraction(big1,big2);

Fraction remainder = new Fraction(BigInteger.valueOf(6),BigInteger.valueOf(1));

assertTrue(remainder.equals(mega.divide(big)));

}

@Test

public void divideTest3(){

BigInteger mega1 = new BigInteger ("-748746501982");

BigInteger mega2 = new BigInteger ("374373250991");

Fraction mega = new Fraction(mega1, mega2);

BigInteger big1 = new BigInteger("33333333333333");

BigInteger big2 = new BigInteger("-99999999999999");

Fraction big = new Fraction(big1,big2);

Fraction remainder = new Fraction(BigInteger.valueOf(6),BigInteger.valueOf(1));

assertTrue(remainder.equals(mega.divide(big)));

}

@Test

public void divideTest4(){

BigInteger mega1 = new BigInteger ("0");

BigInteger mega2 = new BigInteger ("374373250991");

Fraction mega = new Fraction(mega1, mega2);

BigInteger big1 = new BigInteger("33333333333333");

BigInteger big2 = new BigInteger("-99999999999999");

Fraction big = new Fraction(big1,big2);

Fraction remainder = new Fraction(BigInteger.valueOf(0),BigInteger.valueOf(1));

assertTrue(remainder.equals(mega.divide(big)));

}

@Test

public void divideTest5(){

Fraction big = new Fraction(0,1);

Fraction small = new Fraction(-1);

Fraction remainder = new Fraction(0);

assertTrue(remainder.equals(big.divide(small)));

}

// @Test // throw illegal argument

// public void divideTestPrime(){

// BigInteger mega1 = new BigInteger ("748746501982");

// BigInteger mega2 = new BigInteger ("374373250991");

// Fraction mega = new Fraction(mega1, mega2);

// BigInteger big1 = new BigInteger("33333333333333");

// BigInteger big2 = new BigInteger("0");

// Fraction big = new Fraction(big1,big2);

// Fraction remainder = new Fraction(BigInteger.valueOf(6),BigInteger.valueOf(1));

// assertTrue(remainder.equals(mega.divide(big)));

// }

@Test

public void valueOfTest1(){

Fraction big = new Fraction(99,66);

Fraction sum = new Fraction(3,1);

assertTrue(sum.equals(big.add(Fraction.valueOf(3,2))));

}

@Test

public void valueOfTest2(){

Fraction big = new Fraction(99,66);

Fraction sum = new Fraction(0,1);

assertTrue(sum.equals(big.add(Fraction.valueOf(-3,2))));

}

@Test

public void valueOfTest3(){

Fraction big = new Fraction(99,66);

Fraction sum = new Fraction(3,1);

assertTrue(sum.equals(big.add(Fraction.valueOf(3,2))));

}

@Test

public void valueOfTest4(){

BigInteger mega1 = new BigInteger ("-748746501982");

BigInteger mega2 = new BigInteger ("374373250991");

Fraction mega = new Fraction(mega1, mega2);

Fraction sum = new Fraction (-11,7);

assertTrue(sum.equals(mega.add(Fraction.valueOf(3,7))));

}

@Test

public void valueOfTest5(){

BigInteger mega1 = new BigInteger ("-748746501982");

BigInteger mega2 = new BigInteger ("374373250991");

Fraction mega = new Fraction(mega1, mega2);

Fraction product = new Fraction (-8,1);

assertTrue(product.equals(mega.multiply(Fraction.valueOf(-44,-11))));

}

@Test

public void toStringTest1(){

Fraction big = new Fraction (99,66);

String actual = big.toString();

String expect = "3/2";

assertEquals(expect,actual);

}

@Test

public void toStringTest2(){

BigInteger mega1 = new BigInteger ("-748746501982");

BigInteger mega2 = new BigInteger ("374373250991");

Fraction mega = new Fraction(mega1, mega2);

String actual = mega.toString();

String expect = "-2/1";

assertEquals(expect,actual);

}

@Test

public void toStringTest3(){

Fraction big = new Fraction (7376,16);

String actual = (big.divide(Fraction.valueOf(11,1))).toString();

String expect = "461/11";

assertEquals(expect,actual);

}

}