/\* Henry <hc7145@bard.edu> **Fraction.java**

\* 9/14/2018

\* CMSC 201

\* lab2

\* Collaboration I worked alone on this assignment

\* With Assistance From 1) Algorithm 4th ed

\* 2) https://www.youtube.com/watch?v=eo0v-pyc08o

\* 3) https://docs.oracle.com/javase/7/docs/api/java/math/BigInteger.html

\*/

/\*This class supports fraction objects,fraction arithmetics, fraction conversion to commomon datatypes

Also, it implements the comparable interface. Dependencies: java.math.BigInteger \*/

import java.math.BigInteger;

import java.util.Arrays;

public class Fraction extends Number implements Comparable<Fraction>{

// intance variables

private BigInteger numer ;//the numerator

private BigInteger denom ;//the denominator

// constructors

public Fraction (BigInteger n, BigInteger d){

if (d.equals(BigInteger.ZERO)) throw new IllegalArgumentException("Denomintoar cannot be zero!");

numer = n; denom = d;

this.reduction();}// reduces the the fraction to its lowest terms

public Fraction (BigInteger n){numer = n; denom = BigInteger.ONE;this.reduction();}

public Fraction (int n, int d){

if (d == 0) throw new IllegalArgumentException("Denomintoar cannot be zero!");

numer = BigInteger.valueOf(n); denom = BigInteger.valueOf(d);

this.reduction();}

public Fraction (int n){numer = BigInteger.valueOf(n); denom = BigInteger.ONE;this.reduction();}

// instance methods

private void reduction(){// reduces fraction to its lowest terms

BigInteger c\_d = denom.gcd(numer);

if (c\_d.compareTo(BigInteger.ONE)==1){

numer = numer.divide(c\_d); denom = denom.divide(c\_d);

}

}

public BigInteger getNumerator(){return numer;}//returns numerator

public BigInteger getDenominator(){return denom;}//returns denominator

public Fraction add(Fraction f){// add one fraction to another

Fraction f\_new = new Fraction(f.denom.multiply(this.numer).add(this.denom.multiply(f.numer)),

this.denom.multiply(f.denom) );

f\_new.reduction();

return f\_new;}

public Fraction subtract(Fraction f){// subtract one fraction to another

Fraction f\_new = new Fraction(f.denom.multiply(this.numer).subtract(this.denom.multiply(f.numer)),

this.denom.multiply(f.denom) );

f\_new.reduction();

return f\_new;}

public Fraction multiply(Fraction f){// multiply one fraction to another

Fraction f\_new = new Fraction(f.numer.multiply(this.numer), this.denom.multiply(f.denom));

f\_new.reduction();

return f\_new;}

public Fraction divide(Fraction f){// divide one fraction to another

Fraction f\_new = new Fraction(this.numer.multiply(f.denom), f.numer.multiply(this.denom));

f\_new.reduction();

return f\_new;}

public boolean equals(Object o){

if (this == o) return true;

if (o == null) return false;

if (this.getClass() != o.getClass()) return false;

Fraction new\_f = (Fraction) o;

new\_f.reduction(); this.reduction();

if (!new\_f.denom.equals(this.denom)) return false;

if (!new\_f.numer.equals(this.numer)) return false;

return true;

}

public String toString(){//returns the String representation of a Fraction

return this.numer.toString()+"/"+this.denom.toString();}

public static Fraction valueOf(int n, int d){//returns a Fraction n/d

Fraction f = new Fraction(n,d); return f;}

public double doubleValue(){//returns the double value of a Fraction

return this.numer.doubleValue()/this.denom.doubleValue();}

public float floatValue(){//returns the float value of a Fraction

return this.numer.floatValue()/this.denom.floatValue();}

public long longValue(){//returns the long value of a Fraction

return this.numer.divide(this.denom).longValue();}

public int intValue(){//returns the int value of a Fraction

return this.numer.divide(this.denom).intValue();}

public int compareTo(Fraction o){//implements the comparable interface

BigInteger x = this.numer.multiply(o.denom);

BigInteger y = o.numer.multiply(this.denom);

if (x.compareTo(y) < 0) return -1;

if (x.equals(y)) return 0;

return 1;}

}

/\* Henry <hc7145@bard.edu> **FractionTest.java**

\* 9/14/2018

\* CMSC 201

\* lab2

\* Collaboration I worked alone on this assignment

\* With assitance from 1) Jesse Hamlin-Navias

\* 2)https://medium.com/omarelgabrys-blog/comparing-objects-307400115f88\*/

/\*This program includes tests for the Fraction class. Each nameTest corresponds with the method with\*/

import org.junit.Test; import static org.junit.Assert.\*; import java.math.BigInteger;

import java.util.Arrays;

public class FractionTest{

Fraction r1 = new Fraction(BigInteger.valueOf(2), BigInteger.valueOf(200));

Fraction r2 = new Fraction(BigInteger.valueOf(1));

Fraction r3 = new Fraction(3,100);

Fraction r4 = new Fraction(-4);

@Test (expected = IllegalArgumentException.class)

public void ConstructerTestForInt(){//constructors not zero

Fraction er = new Fraction(1, 0);}

@Test (expected = IllegalArgumentException.class)

public void ConstructerTestForBig(){//constructors not zero

Fraction er = new Fraction(BigInteger.ONE, BigInteger.ZERO);}

@Test

public void getNumeratorTest(){//test reudction to lowest terms

assertEquals(r1.getNumerator(), BigInteger.ONE);}

@Test

public void getDenominatorTest(){//test reudction to lowest terms

assertEquals(r1.getDenominator(), new BigInteger("100"));}

@Test

public void addTest(){

Fraction ans = new Fraction(101,100);

assertEquals(r1.add(r2), ans);}

@Test

public void subtractTest(){

Fraction ans = new Fraction(1,50);

assertEquals(r3.subtract(r1),ans);}

@Test

public void multiplyTest(){//test negative number

Fraction ans = new Fraction(-1,25);

assertEquals(r4.multiply(r1),ans);}

@Test

public void divideTest(){

Fraction ans = new Fraction(10,11);

Fraction divider = new Fraction(11,1000);

assertEquals(r1.divide(divider), ans);}

@Test

public void equalsTest(){

Fraction ans = new Fraction(1,100);

assertEquals(r1.equals(ans),true);}

@Test

public void toStringTest(){

Fraction ans = new Fraction(101, 100);

assertEquals(ans.toString(), "101/100");}

@Test

public void valueOfTest(){

assertEquals(Fraction.valueOf(202,200), new Fraction(101, 100));}

@Test

public void doubleValueTest(){

Fraction f = new Fraction(new BigInteger("334444444444444"), new BigInteger("111111111111111"));

assertEquals(f.doubleValue(),3.009999999999999, Math.pow(10,-15));}

@Test

public void floatValueTest(){

Fraction f = new Fraction(new BigInteger("334444444444444"), new BigInteger("111111111111111"));

assertEquals(f.floatValue(),3.0099998, Math.pow(10,-7));

System.out.print(f.floatValue());}

@Test

public void longValueTest(){

Fraction f = new Fraction(new BigInteger("1111111111111111"), new BigInteger("1"));

long ans = 1111111111111111L;

assertEquals(ans, f.longValue());}

@Test

public void intValueTest(){

Fraction f = new Fraction(new BigInteger("1111111111111111"), new BigInteger("1"));

int ans = -1223331330;

assertEquals(ans, f.intValue(), 1);}

@Test

public void compareToTest(){

Fraction[] fa = {new Fraction(1,2),new Fraction(1,1),new Fraction(0,1),new Fraction(3,4)};

Fraction[] faSorted = {new Fraction(0,1),new Fraction(1,2),new Fraction(3,4),new Fraction(1,1)};

Arrays.sort(fa);

for (int i=0; i<4; i++){

assertEquals(fa[i], faSorted[i]);}

}

}