JAVA ASSIGNMENT   ASSIGNMENT-7   DATE:25-03-23

PROBLEM STATEMENT :

a a b
Assignment description: A rational number is a number in the form of b where and are integers andb # 0. Rational numbers can be added, subtracted, multiplied, and divided. Write a Java application

that will be able to add, subtract, multiply, divide, compare, convert to floating point, and find absolute
value for rational numbers.

Your program should be written in Object Oriented Programming style. The program should accept two
rational numbers from the user from keyboard (Through command line arguments) and output results
of operations to console. Your program should solve operations efficiently and be able to recover from
bad inputs. Use exception handling mechanism so as not to crash the program.

Example Inputs:

1234/5678 and 8765 / 4321
0/1 and 34/675
apple /23 and 23/0

.

CODE:

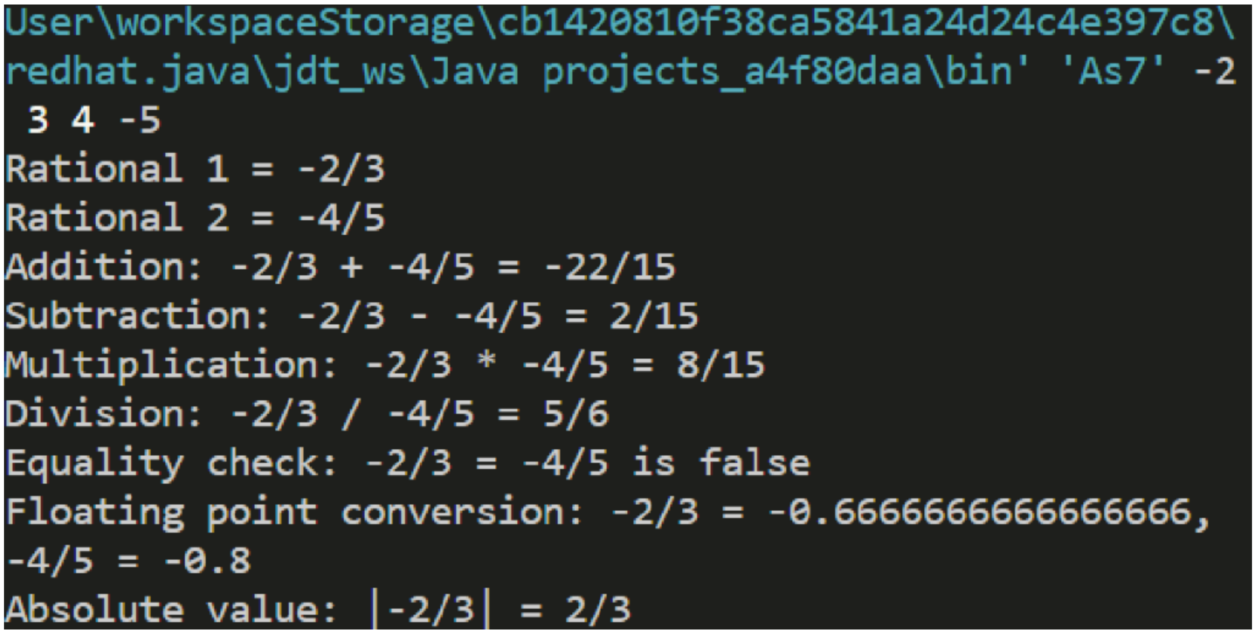
```
/*
Assignment description: A rational number is a number in the form of b where and are integers
                andb # 0. Rational numbers can be added, subtracted, multiplied, and divided.
                Write a Java application that will be able to add, subtract, multiply, divide,
compare,
                convert to floating point, and find absolute value for rational numbers.
```

```
            Your program should be written in Object Oriented Programming style. The
program should accept two
            rational numbers from the user from keyboard (Through command line
arguments) and output results
            of operations to console. Your program should solve operations efficiently and be
able to recover from
            bad inputs. Use exception handling mechanism so as not to crash the program.



Name - Nisarg Patel
PRN - 21070126060
Batch - AIML A3

LAB ASSIGNMENT - 7
*/



package as7;

import java.util.InputMismatchException;
import java.util.Scanner;

// Class representing a rational number
class RationalNumber {
    private int numerator; // Numerator of the rational number
    private int denominator; // Denominator of the rational number

    // Constructor to create a rational number
    public RationalNumber(int numerator, int denominator) {
        // Check if denominator is zero, throw an exception if it is
        if (denominator == 0) {
            throw new IllegalArgumentException("Denominator cannot be zero.");
        }
        // Initialize the numerator and denominator
        this.numerator = numerator;
        this.denominator = denominator;
        simplify(); // Simplify the rational number
    }

    // Method to add two rational numbers
    public RationalNumber add(RationalNumber other) {
```

```
    int resultNumerator = this.numerator * other.denominator + other.numerator *
this.denominator; // Calculate the numerator of the result
    int resultDenominator = this.denominator * other.denominator; // Calculate the denominator
of the result
    return new RationalNumber(resultNumerator, resultDenominator); // Create and return a
new rational number
  }

  // Method to subtract two rational numbers
  public RationalNumber subtract(RationalNumber other) {
    int resultNumerator = this.numerator * other.denominator - other.numerator *
this.denominator; // Calculate the numerator of the result
    int resultDenominator = this.denominator * other.denominator; // Calculate the denominator
of the result
    return new RationalNumber(resultNumerator, resultDenominator); // Create and return a
new rational number
  }

  // Method to multiply two rational numbers
  public RationalNumber multiply(RationalNumber other) {
    int resultNumerator = this.numerator * other.numerator; // Calculate the numerator of the
result
    int resultDenominator = this.denominator * other.denominator; // Calculate the denominator
of the result
    return new RationalNumber(resultNumerator, resultDenominator); // Create and return a
new rational number
  }

  // Method to divide two rational numbers
  public RationalNumber divide(RationalNumber other) {
    // Check if the other rational number is zero, throw an exception if it is
    if (other.numerator == 0) {
      throw new ArithmeticException("Cannot divide by zero.");
    }
    int resultNumerator = this.numerator * other.denominator; // Calculate the numerator of the
result
    int resultDenominator = this.denominator * other.numerator; // Calculate the denominator of
the result
    return new RationalNumber(resultNumerator, resultDenominator); // Create and return a
new rational number
  }

  // Method to check if two rational numbers are equal
  public boolean equals(RationalNumber other) {
        return this.numerator == other.numerator && this.denominator == other.denominator; //
Return true if the numerators and denominators are equal
    }

    // Method to convert a rational number to a double
    public double toDouble() {
        return (double) this.numerator / this.denominator; // Return the result of the division as a
double
    }

    // Method to get the absolute value of a rational number
    public RationalNumber abs() {
        int absNumerator = Math.abs(this.numerator); // Calculate the absolute value of the
numerator
        int absDenominator = Math.abs(this.denominator); // Calculate the absolute value of the
denominator
        return new RationalNumber(absNumerator, absDenominator); // Create and return a new
rational number
    }
    private void simplify() { // Simplipying things
        int gcd = gcd(this.numerator, this.denominator);
        this.numerator /= gcd;
        this.denominator /= gcd;
        if (this.denominator < 0) {
            this.numerator = -this.numerator;
            this.denominator = -this.denominator;
        }
    }

    private int gcd(int a, int b) {//Making a gcd
        if (b == 0) {
            return a;
        }
        return gcd(b, a % b);
    }

    @Override
    public String toString() {
        return this.numerator + "/" + this.denominator; // Converting to string
    }
}

public class As7 {
    public static void main(String[] args) {
        try {
            int numerator1 = Integer.parseInt(args[0]);
            int denominator1 = Integer.parseInt(args[1]);
            RationalNumber rational1 = new RationalNumber(numerator1, denominator1);

            int numerator2 = Integer.parseInt(args[2]);
            int denominator2 = Integer.parseInt(args[3]);
            RationalNumber rational2 = new RationalNumber(numerator2, denominator2);

            System.out.println("Rational 1 = " + rational1); // Printing the first rational number
            System.out.println("Rational 2 = " + rational2); // Printing the second rational number


            //  For executing a single function out of many, use the following code:

            //     if(args[4].equalsIgnoreCase("add")) {
            //         RationalNumber result = rational1.add(rational2);
            //         System.out.println("Addition: " + rational1 + " + " + rational2 + " = " + result);
            //     } else if(args[4].equalsIgnoreCase("subtract")){
            //         RationalNumber result = rational1.subtract(rational2);
            //         System.out.println("Subtraction: " + rational1 + " - " + rational2 + " = " + result);
            //     } else if(args[4].equalsIgnoreCase("multiply")){
            //         RationalNumber result = rational1.multiply(rational2);
            //         System.out.println("Multiplication: " + rational1 + " * " + rational2 + " = " + result);
            //     } else if(args[4].equalsIgnoreCase("divide")) {
            //         try {
            //             RationalNumber result = rational1.divide(rational2);
            //             System.out.println("Division: " + rational1 + " / " + rational2 + " = " + result);
            //         } catch (ArithmeticException e) {
            //             System.out.println("Division error: " + e.getMessage());
            //         }
            //     } else if(args[4].equalsIgnoreCase("equals")){
            //         boolean isEqual = rational1.equals(rational2);
            //         System.out.println("Equality check: " + rational1 + " = " + rational2 + " is " +
isEqual);
            //     } else if(args[4].equalsIgnoreCase("toDouble")) {
            //         double doubleValue1 = rational1.toDouble();
            //         double doubleValue2 = rational2.toDouble();
            //         System.out.println("Floating point conversion: " + rational1 + " = " + doubleValue1
+ ", " + rational2 + " = " + doubleValue2);
            //     } else if(args[4].equalsIgnoreCase("abs")){
            //         RationalNumber result = rational1.abs();
            //         System.out.println("Absolute value: |" + rational1 + "| = " + result);
            //     } else {
```

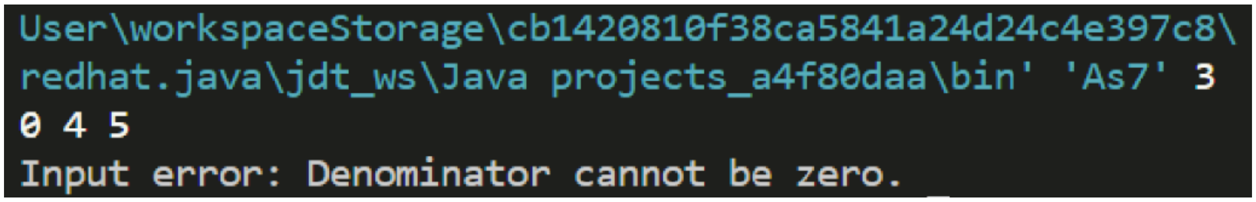
```
        //         System.out.println("Invalid operation");
        //     }
        // } catch (IllegalArgumentException e) {
        //     System.out.println("Invalid input: " + e.getMessage());
        // }

        //For executing all the functions, use the following code:

        RationalNumber result = rational1.add(rational2);
        System.out.println("Addition: " + rational1 + " + " + rational2 + " = " + result); //Add
        result = rational1.subtract(rational2);
        System.out.println("Subtraction: " + rational1 + " - " + rational2 + " = " + result); //Subtract
        result = rational1.multiply(rational2);
        System.out.println("Multiplication: " + rational1 + " * " + rational2 + " = " + result); //
Multiply

        try {
            result = rational1.divide(rational2);
            System.out.println("Division: " + rational1 + " / " + rational2 + " = " + result);  // Division
        } catch (ArithmeticException e) {
            System.out.println("Division error: " + e.getMessage());
        }

        boolean isEqual = rational1.equals(rational2);
        System.out.println("Equality check: " + rational1 + " = " + rational2 + " is " + isEqual);

        double doubleValue1 = rational1.toDouble();
        double doubleValue2 = rational2.toDouble();
        System.out.println("Floating point conversion: " + rational1 + " = " + doubleValue1 + ", " +
rational2 + " = " + doubleValue2);

        result = rational1.abs();
        System.out.println("Absolute value: |" + rational1 + "| = " + result); // abs
      } catch (NumberFormatException e) {
        System.out.println("Input error: " + e.getMessage() + ". Please enter integers as input.");
      } catch (IllegalArgumentException e) {
        System.out.println("Input error: " + e.getMessage()); // Throwing Error
      } catch (ArrayIndexOutOfBoundsException e) {
        System.out.println("Usage: java As7 <numerator1> <denominator1> <numerator2>
<denominator2>");
      }
   }
}
```

OUTPUT :



```
User\workspaceStorage\cb1420810f38ca5841a24d24c4e397c8\
redhat.java\jdt_ws\Java projects_a4f80daa\bin' 'As7' -2
 3 4 -5
Rational 1 = -2/3
Rational 2 = -4/5
Addition: -2/3 + -4/5 = -22/15
Subtraction: -2/3 - -4/5 = 2/15
Multiplication: -2/3 * -4/5 = 8/15
Division: -2/3 / -4/5 = 5/6
Equality check: -2/3 = -4/5 is false
Floating point conversion: -2/3 = -0.6666666666666666,
-4/5 = -0.8
Absolute value: |-2/3| = 2/3
```

Exception thrown:



```
User\workspaceStorage\cb1420810f38ca5841a24d24c4e397c8\
redhat.java\jdt_ws\Java projects_a4f80daa\bin' 'As7' 3
0 4 5
Input error: Denominator cannot be zero.
```

GitHub Repository: https://github.com/SnakeEyes1308/Java-Assignment-