# Fraction Calculator report

** **, ** **

August 9, 2021

# Contents

# Part I

# Project report

# Chapter 1

# Introduction

We have written a fraction based calculator using JavaFX library and it is capable of calculating basic operations on fractions and integers.

## Used Java Libraries

- JavaFX library

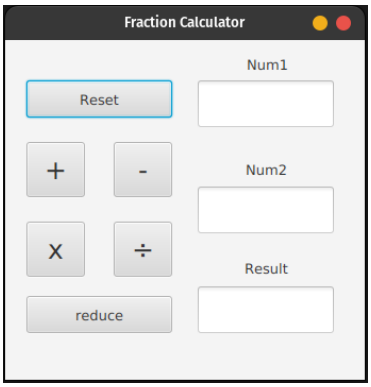
# Chapter 2

# Project

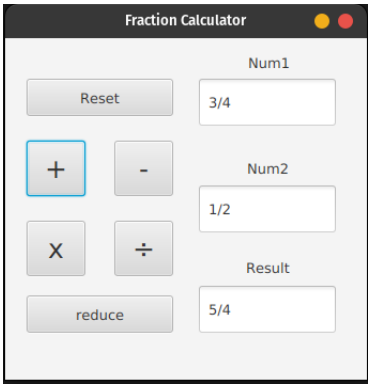
the JavaFX project which is a fraction caculator can be used to simple integer math operations or using fractions as seen below

## preview images

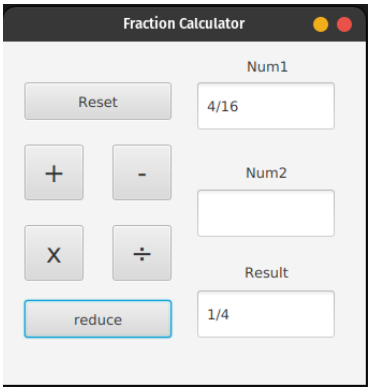
User interface



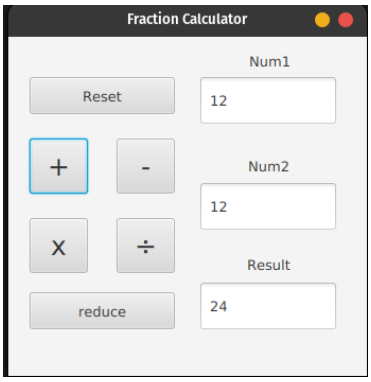

Add Fractions




Reducing Fractions




Adding Integers



## Work distrubution

- Abdussamad: App and Controller
- Hassan: Fraction and ParseInput

## Code Structure

the Code base contains 4 classes

- App : Which contains the main function
- ParseInput : which contains Input parser
- Controller : which contains code interfacing with the UI
- Fraction : which contains the calculations in the program

---

1. App class
   (a) sets the main stage and scene and loads a fxml file `App.fxml` and css file `style.css`
   (b) resizeable is set to false
   (c) only one scene and stage is used in the program
2. ParseInput class
   (a) receives String input from each TextField in the program and parses to 2 numbers
   (b) It supports special handling if the String input is an integer
   (c) Error handling is catched and thrown till reaches Controller class
3. Controller class
   (a) Method used for each button and calls other classes in each method
   (b) Error handling in the whole program is in the controller class
   (c) Objects for mutable UI elements are defined in the class
4. Fraction class
   (a) contains 2 private variables for numerator and denominator

(b) Constructor with String parameter calls ParseInput class and also it contains 2 long parameter constructor

(c) the class contains method for each operation and one for getting greatest common denominator

---

# Part II

# App.java

```java
import java.io.IOException;import java.io.IOException;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;

public class App extends Application {

    public static void main(String[] args) {
        launch(args);
    }

    @Override
    public void start(Stage primaryStage) throws
    ↪  IOException {
        primaryStage.setTitle("Fraction Calculator");
        Parent root = FXMLLoader.load(getClass()
                    .getResource("App.fxml"));
        Scene scene = new Scene(root);
        scene.getStylesheets().add(
            getClass().getResource("style.css")
            .toExternalForm());
        primaryStage.setScene(scene);
        primaryStage.setResizable(false);
        primaryStage.show();
    }
}
```

# Part III

# ParseInput.java

```java
import java.util.InputMismatchException;
import java.util.regex.Pattern;

public class ParseInput {
    private long top;
    private long down;

    public ParseInput(String input) throws
    ↪ InputMismatchException, NumberFormatException,
    ↪ ArithmeticException {
        String[] arr = input.split("/");
        // used regex to get if the string is only
        ↪ numbers
        if (Pattern.matches("[0-9]+[\\.]?[0-9]*", input))
        ↪ {
            top = Long.parseLong(input);
            down = 1;
        } else if (arr.length != 2) {
            throw new InputMismatchException("Wrong
            ↪ Input. Input must be a fraction");
        } else {
            top = Long.parseLong(arr[0]);
            down = Long.parseLong(arr[1]);
        }
        if (down == 0) {
            throw new ArithmeticException("Divide by Zero
            ↪ error");
        }
    }

    public long getTop() {
        return top;
    }

    public long getDown() {
        return down;
    }
}
```

# Part IV

# Fraction.java

```java
import java.util.InputMismatchException;

public class Fraction {
    private long numerator;
    private long denominator;

    public Fraction(String input) throws
    ↪  InputMismatchException, NumberFormatException,
    ↪  ArithmeticException {
        ParseInput parsedInput = new ParseInput(input);
        long parsedDenominator = parsedInput.getDown();
        long parsedNumerator = parsedInput.getTop();
        long gcd =
        ↪  greatCommonDenominator(parsedNumerator,
        ↪  parsedDenominator);
        this.numerator = (parsedDenominator > 0 ? 1 : -1)
        ↪  * parsedNumerator / gcd;
        this.denominator = Math.abs(parsedDenominator) /
        ↪  gcd;
    }

    public Fraction(long numerator, long denominator) {
        long gcd = greatCommonDenominator(numerator,
        ↪  denominator);
        this.numerator = (denominator > 0 ? 1 : -1) *
        ↪  numerator / gcd;
        this.denominator = Math.abs(denominator) / gcd;
    }

    private static long greatCommonDenominator(long n,
    ↪  long d) {
        long num1 = Math.abs(n);
        long num2 = Math.abs(d);
        int gcd = 1;

        for (int i = 1; i <= num1 && i <= num2; i++) {
            if (num1 % i == 0 && num2 % i == 0)
                gcd = i;
        }
```

```
31          return gcd;
32      }
33
34      public long getNumerator() {
35          return numerator;
36      }
37
38      public long getDenominator() {
39          return denominator;
40      }
41
42      public Fraction addFraction(Fraction other) {
43          long n = numerator * other.getDenominator() +
            ↪  denominator * other.getNumerator();
44          long d = denominator * other.getDenominator();
45          return new Fraction(n, d);
46      }
47
48      public Fraction subtractFraction(Fraction other) {
49          long n = numerator * other.getDenominator() -
            ↪  denominator * other.getNumerator();
50          long d = denominator * other.getDenominator();
51          return new Fraction(n, d);
52      }
53
54      public Fraction multiplyFraction(Fraction other) {
55          long n = numerator * other.getNumerator();
56          long d = denominator * other.getDenominator();
57          return new Fraction(n, d);
58      }
59
60      public Fraction divideFraction(Fraction other) throws
        ↪  ArithmeticException {
61          long n = numerator * other.getDenominator();
62          long d = denominator * other.numerator;
63          if (d == 0) {
64              throw new ArithmeticException("Divide by Zero
                ↪  error");
65          }
66          return new Fraction(n, d);
```

```
67      }
68
69      public void reduceFraction() {
70          long gcd = greatCommonDenominator(numerator,
            ↪  denominator);
71          this.numerator = (denominator > 0 ? 1 : -1) *
            ↪  numerator / gcd;
72          this.denominator = Math.abs(denominator) / gcd;
73      }
74
75      @Override // Override toString()
76      public String toString() {
77          if (denominator == 1)
78              return numerator + "";
79          else if (numerator == 0)
80              return "0";
81          else
82              return numerator + "/" + denominator;
83      }
84  }
```

# Part V

# Controller.java

```java
import java.util.InputMismatchException;

import javafx.fxml.FXML;
import javafx.scene.control.Alert;
import javafx.scene.control.Button;
import javafx.scene.control.ButtonType;
import javafx.scene.control.TextField;
import javafx.scene.control.Alert.AlertType;

public class Controller {

    @FXML
    private Button resetBtn;
    @FXML
    private Button addBtn;
    @FXML
    private Button subBtn;
    @FXML
    private Button multBtn;
    @FXML
    private Button divBtn;
    @FXML
    private Button reduceBtn;
    @FXML
    private TextField num1TextField;
    @FXML
    private TextField num2TextField;
    @FXML
    private TextField resultTextField;
    @FXML
    private Alert alert;

    public void showAlert(String log) {
        alert = new Alert(AlertType.ERROR);
        alert.setTitle("Error");
        alert.setHeaderText(log);
        alert.setContentText("Click on ok to reset
        ↪  fields");
        if (alert.showAndWait().get() == ButtonType.OK) {
```

```
39              resetFields();
40          }
41      }
42
43      public void resetFields() {
44          num1TextField.clear();
45          num2TextField.clear();
46          resultTextField.clear();
47      }
48
49      public void add() {
50          try {
51              Fraction num1 = new
                ↪  Fraction(num1TextField.getText());
52              Fraction num2 = new
                ↪  Fraction(num2TextField.getText());
53              resultTextField.setText(num1
54                      .addFraction(num2).toString());
55          } catch (InputMismatchException |
                ↪  NumberFormatException | ArithmeticException
                ↪  e) {
56              showAlert(e.getMessage());
57          }
58      }
59
60      public void subtract() {
61          try {
62              Fraction num1 = new
                ↪  Fraction(num1TextField.getText());
63              Fraction num2 = new
                ↪  Fraction(num2TextField.getText());
64              resultTextField.setText(num1
65                      .subtractFraction(num2).toString());
66          } catch (InputMismatchException |
                ↪  NumberFormatException | ArithmeticException
                ↪  e) {
67              showAlert(e.getMessage());
68          }
69      }
70
```

```
71      public void multiply() {
72          try {
73              Fraction num1 = new
                ↪  Fraction(num1TextField.getText());
74              Fraction num2 = new
                ↪  Fraction(num2TextField.getText());
75              resultTextField.setText(num1
76                      .multiplyFraction(num2).toString());
77          } catch (InputMismatchException |
            ↪  NumberFormatException | ArithmeticException
            ↪  e) {
78              showAlert(e.getMessage());
79          }
80      }
81
82      public void divide() {
83          try {
84              Fraction num1 = new
                ↪  Fraction(num1TextField.getText());
85              Fraction num2 = new
                ↪  Fraction(num2TextField.getText());
86              resultTextField.setText(num1
87                      .divideFraction(num2).toString());
88          } catch (InputMismatchException |
            ↪  NumberFormatException | ArithmeticException
            ↪  e) {
89              showAlert(e.getMessage());
90          }
91      }
92
93      public void reduce() {
94          try {
95              if (num2TextField.getText().isBlank() &&
                ↪  !num1TextField.getText().isBlank()) {
96                  Fraction num1 = new
                    ↪  Fraction(num1TextField.getText());
97                  resultTextField.setText(num1.toString());
98              } else if (num1TextField.getText().isEmpty()
                ↪  && !num2TextField.getText().isEmpty()) {
```

```
                Fraction num2 = new
                ↪  Fraction(num2TextField.getText());
                num2.reduceFraction();
                resultTextField.setText(num2.toString());
            } else {
                throw new InputMismatchException("Please
                ↪  Enter one fraction");
            }
        } catch (InputMismatchException |
        ↪  NumberFormatException | ArithmeticException
        ↪  e) {
            showAlert(e.getMessage());
        }
    }
}
```