```
"""
Author: Coral S. Schmidt Montilla
Student Number: 148830

This module provides a class called Raster for performing arithmetic with
fractions.
The class Raster handles addition, subtraction, multiplication, and division
of rational numbers.
It also reduces fractions to their simplest form and supports conversion to
floating-point format.
"""


class Rational:
    def __init__(self, numerator=0, denominator=1):
        self.__denominator = None
        self.__numerator = None
        self.set_fraction(numerator, denominator)

    def set_fraction(self, numerator, denominator):
        if denominator == 0:
            raise ValueError("Denominator cannot be zero.")
        self.__numerator = numerator
        self.__denominator = denominator
        self.simplest_form()

    def simplest_form(self):
        gcd = self.greatest_common_denominator(self.__numerator,
self.__denominator)
        self.__numerator //= gcd
        self.__denominator //= gcd

    def to_mixed_fraction(self):
        whole = self.__numerator // self.__denominator
        remainder = self.__numerator % self.__denominator
        if remainder == 0:
            return str(whole)
        elif whole == 0:
            return f"{remainder}/{self.__denominator}"
        else:
            return f"{whole} {abs(remainder)}/{self.__denominator}"

    @staticmethod
    def greatest_common_denominator(a, b):
        while b != 0:
            a, b = b, a % b
        return a

    def __str__(self):
        return f"{self.__numerator}/{self.__denominator}"

    def to_float(self, precision=None):
        if self.__denominator == 0:
            return "Undefined"
        result = self.__numerator / self.__denominator
        if precision is not None:
            return f"{result:.{precision}f}"
```

```
        return str(result)

    def add(self, other):
        new_numerator = self.__numerator * other.__denominator +
other.__numerator * self.__denominator
        new_denominator = self.__denominator * other.__denominator
        return Rational(new_numerator, new_denominator)

    def subtract(self, other):
        new_numerator = self.__numerator * other.__denominator -
other.__numerator * self.__denominator
        new_denominator = self.__denominator * other.__denominator
        return Rational(new_numerator, new_denominator)

    def multiply(self, other):
        new_numerator = self.__numerator * other.__numerator
        new_denominator = self.__denominator * other.__denominator
        return Rational(new_numerator, new_denominator)

    def divide(self, other):
        if other.__numerator == 0:
            raise ValueError("Division by zero is not allowed.")
        new_numerator = self.__numerator * other.__denominator
        new_denominator = self.__denominator * other.__numerator
        return Rational(new_numerator, new_denominator)

    def get_numerator(self):
        return self.__numerator

    def get_denominator(self):
        return self.__denominator
"""
Author: Coral S. Schmidt Montilla
Student Number: 148830

This application provides a graphical user interface for performing
arithmetic
operations on rational numbers. It accepts input as fractions,
performs addition, subtraction, multiplication, and division based on user
selection,
and displays the results.
"""

import sys
from PyQt5 import uic
from PyQt5.QtWidgets import QMainWindow, QApplication
from Rational import Rational


class RationalDriver(QMainWindow):
    def __init__(self):
        super().__init__()
        uic.loadUi('Asig_5.ui', self)
        self.GetResultButton.clicked.connect(self.calculate_result)

    def calculate_result(self):
        try:
```

```
            num1 = int(self.rational1_lineEdit.text().split('/')[0])
            den1 = int(self.rational1_lineEdit.text().split('/')[1])
            num2 = int(self.rational2_lineEdit.text().split('/')[0])
            den2 = int(self.rational2_lineEdit.text().split('/')[1])

            r1 = Rational(num1, den1)
            r2 = Rational(num2, den2)

            if self.addition_button.isChecked():
                result = r1.add(r2)
            elif self.subs_button.isChecked():
                result = r1.subtract(r2)
            elif self.multiplication_button.isChecked():
                result = r1.multiply(r2)
            elif self.div_button.isChecked():
                result = r1.divide(r2)
            else:
                raise ValueError("Please select an operation.")

            if result.get_numerator() == result.get_denominator():
                self.result_label.setText(str(result.get_numerator()))
            else:
                self.result_label.setText(str(result.to_mixed_fraction()))
        except Exception as e:
            self.result_label.setText("Error: " + str(e))


if __name__ == "__main__":
    app = QApplication(sys.argv)
    window = RationalDriver()
    window.show()
    sys.exit(app.exec_())
```