# **CIS 225 101 – Spring 2014**

# **Project #2**

**Name: Due Dates:**

|  |  |
| --- | --- |
| Documentation (30):  Cover Page (2)  Table of Content (2)  Abstract (5)  Design (UML diagram and Javadocs for Customer and Bank classes) (11)  Source Code (5)  Sample Input (5) |  |
| Sample Output(10) |  |
| Correct Logic(40):  Clean compile |  |
| Structure(20):  white spacing (5)  proper indentation(5)  meaningful names(5)  logical flow(5) |  |
| Total points: out of 100% |  |

**AT THE BEGINNING OF LAB (you must demonstrate to me that the program is working)**

Your contribution to the project in percent\_\_\_\_\_\_\_\_\_\_\_\_\_

Your partner’s contribution to the project in percent\_\_\_\_\_\_\_\_\_\_\_\_\_

Explain in Detail your contribution:

**Comments:**

# **CIS 225 Advanced Java – Spring 2014**

**Sara Wexler**

# Swing Lab Assignment: MenusClass

**Program Name**: Project 2

**Abstract:**

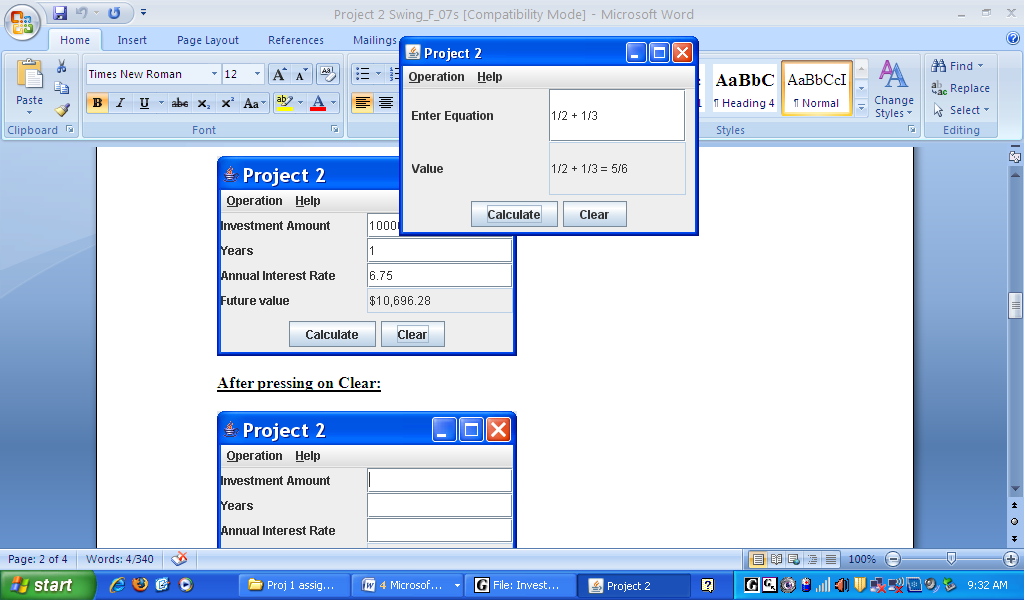
This program using Graphical User Interface process arithmetic and logical expressions containing pairs of rational numbers example: 1/3 + 1/5 = 8/15.

The interface contains text fields for the arithmetic or logical expression

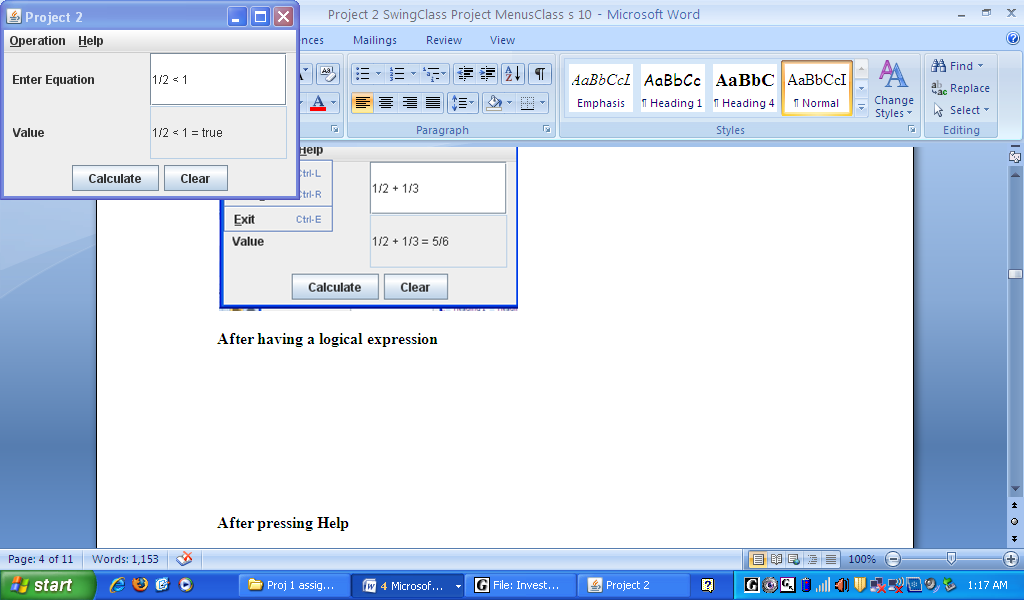
Display the result in a text field when the user clicks the calculate button. Or choose Calculate from the Operation menu. Show a message dialog boxes when the user clicks the About menu item from the Help menu, or Input Information menu item

**Sample of running the program.**

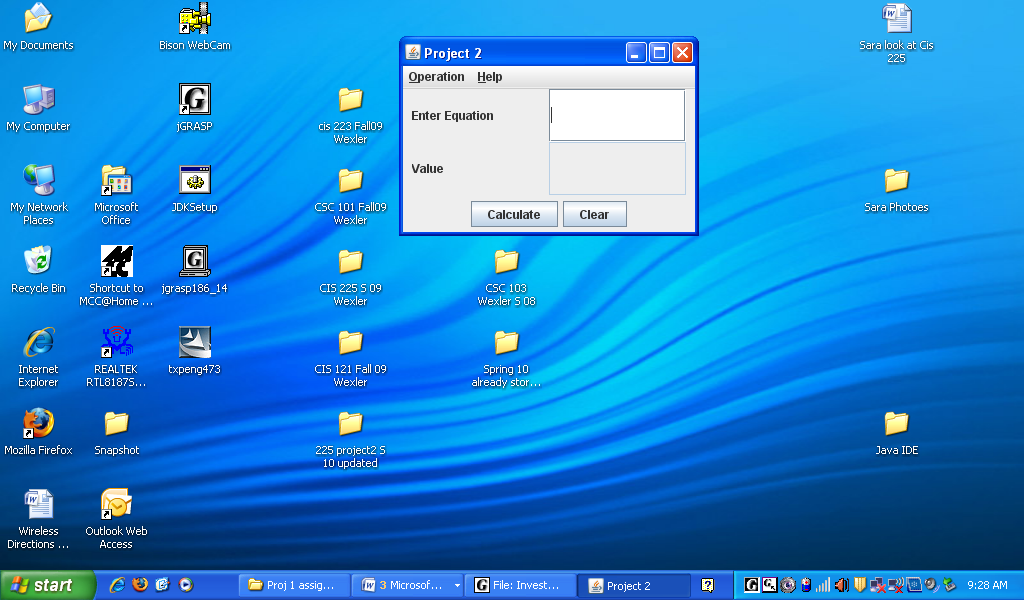
**After entering an arithmetic expression:**

****

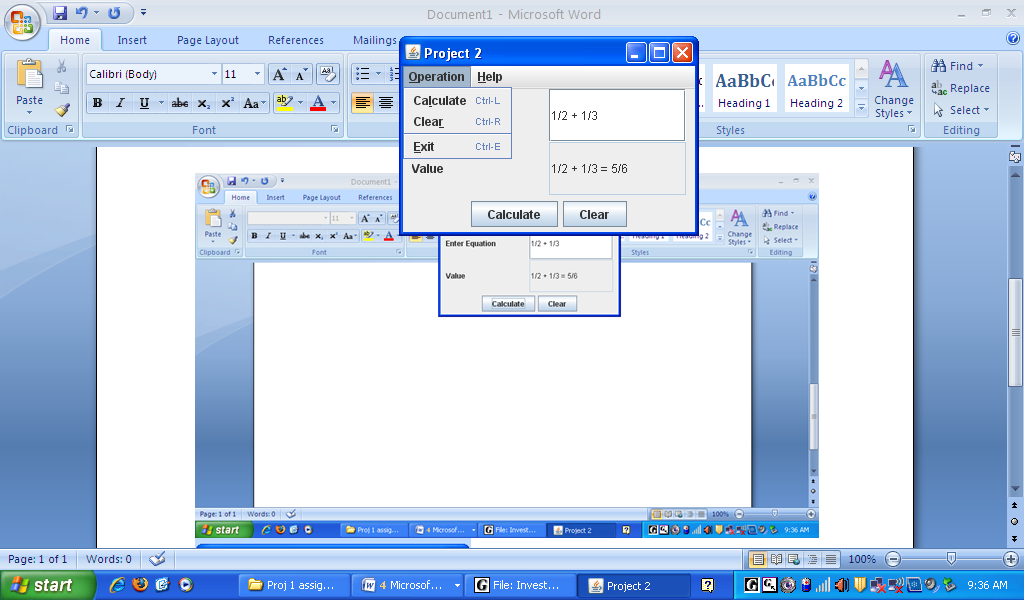
**After entering an arithmetic expression:**

****

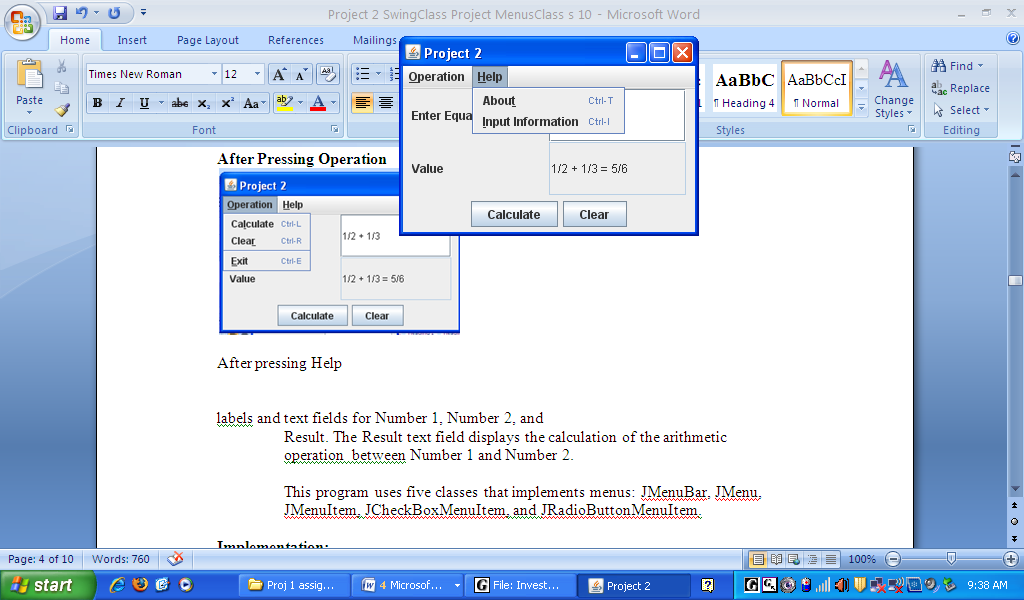
**After pressing Clear**



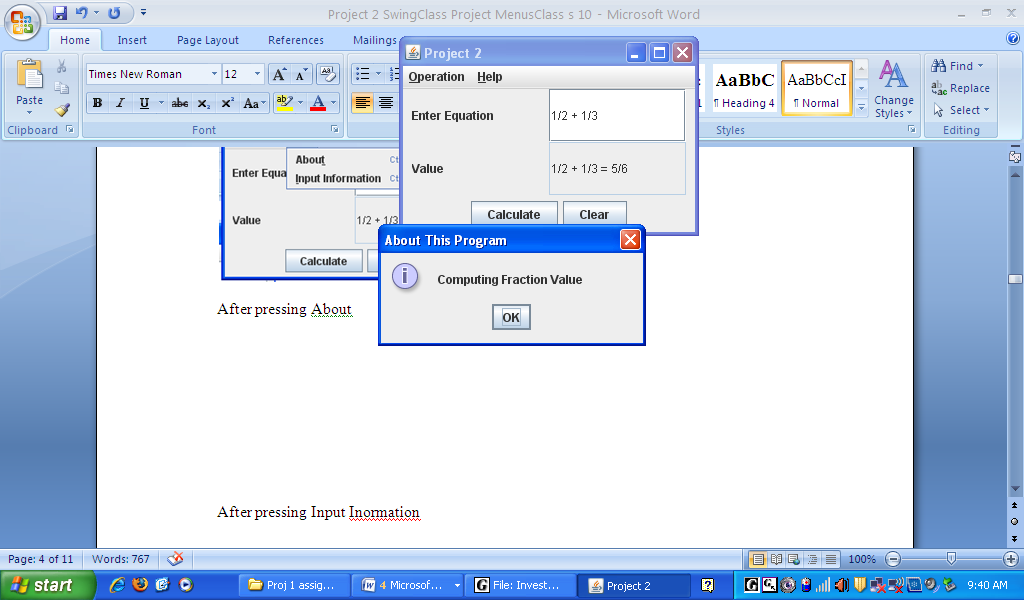
**After Pressing Operation**



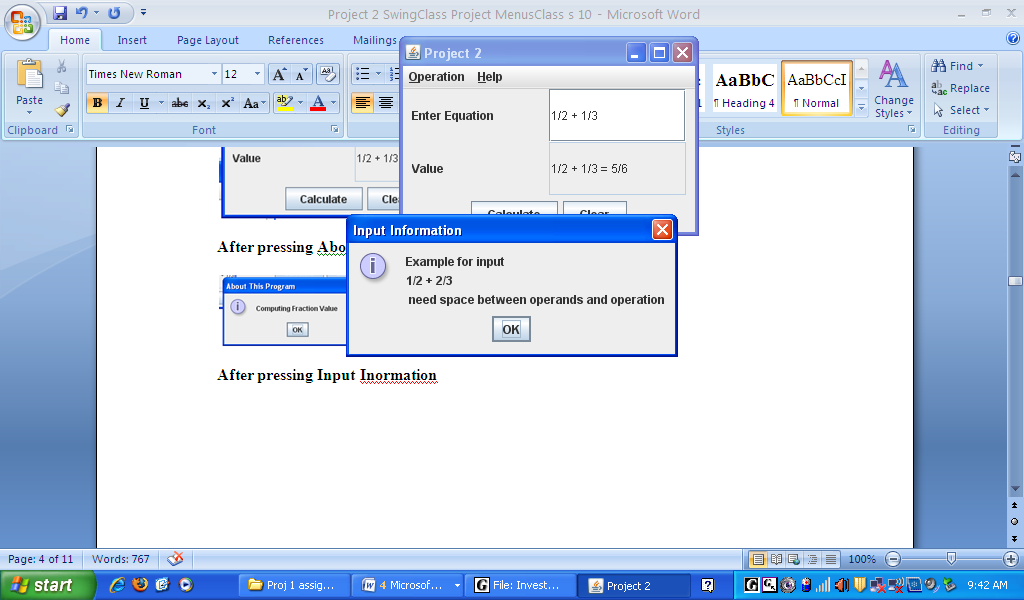
**After pressing Help**



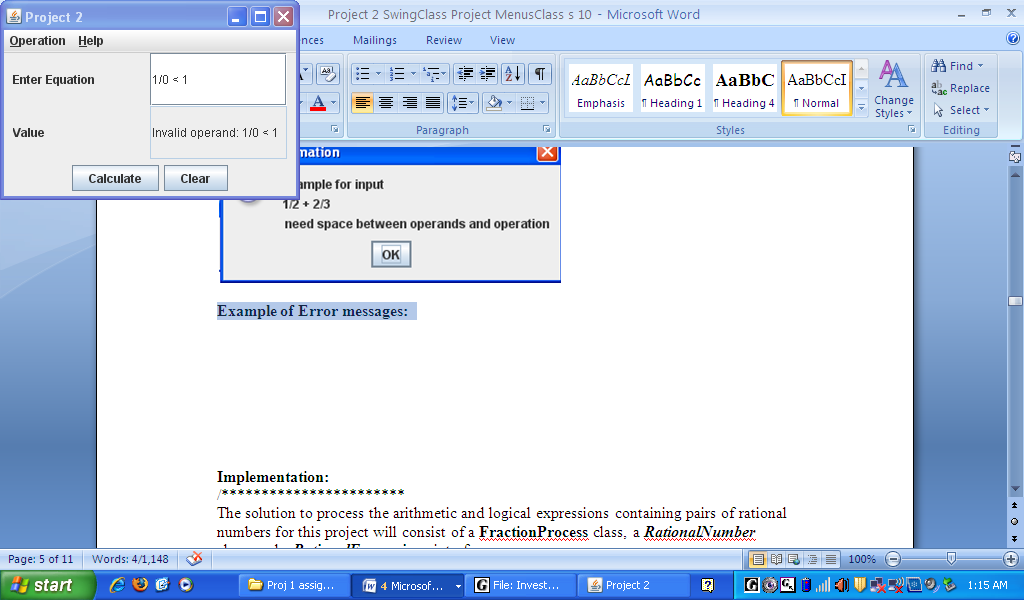
**After pressing About**



**After pressing Input Information**



**Example of Error messages:**



# Program Specifications

1. **The program interface:**

* Create the main menu bar ( JMenuBar) ,
* Create Jmenu items for Operation with Mnemonic ‘O’ and Help Mnemonic ‘H’ and add to the menu bar.
* To the Operation menu add calculate, clear, exit , add a Separator before Exit (look at the picture)
* To the help menu add About and Input Information
* Set accelerators (Cntl) to the menu keys
* Create a new panel with GridLayout to hold labels and text boxes
* Create a Panel to hold the Calculate and Clear Button
* Add the panels to the frame using border layout
* Register listener

1. **Program specification:**

The solution to process the arithmetic and logical expressions containing pairs of rational numbers for this project will consist of a **FractionProcess** class, a ***RationalNumber*** class, and a ***RationalExpressions*** interface.

The **FractionProcess** processes a one-line infix notation expression. For an arithmetic operators, the program should display the calculated result. For a logical operators, the program should display either TRUE or FALSE depending upon the truth-value of the expression.

The following cases should generate an error message and the program should continue:

* If an input expression is malformed (not of the form ***operand operator operand*** )
* If the operator or operand(s) entered are invalid,

A concrete ***RationalNumber*** class should be created. This class must implement the ***RationalExpressions*** interface. A rational object should be created for each rational number read in as input and should be used for processing within the application. Selection logic is used to process the operands based on the operator value.

**Design a program with the following specifications:**

* A rational number is of the form **a / b**, where the division is only indicated not performed. The numerator and denominator are both integers. The denominator may not be zero. An integer can be interpreted as a rational number by assigning the integer to the numerator and setting the denominator to one (i.e.: a ≡ **a / 1**).
* For this project, it is not necessary to reduce or normalize the rational numbers for output.
* Extraneous data in any expression can be ignored.
* For arithmetic expressions (+, /, etc), display the result of the expression.
* For logical expressions (=, !=, <, >, <=, >=), display the Boolean result (true or false). The case of the answer doesn’t matter.
* The program must trap and handle the following errors:
* Invalid operators
* Invalid operands
* Malformed expressions

# Program Implementation

* Implement the program as a Java application separated into three compilation units:
* a main class which holds the graphics interface,
* the ***RationalExpression*** interface.
* a ***RationalNumber*** class,
* Name the main class **Project2 .**
* For internal documentation, use plenty of comments throughout the program.
* For external documentation use a **UML** class diagram, **Javadoc** comments and run the source code through the javadoc utility.

**RationalExpressions Interface**

**interface RationalExpressions**

{

public RationalNumber **add**(RationalNumber r);

public RationalNumber **sub**(RationalNumber r);

public RationalNumber **mlt**(RationalNumber r);

public RationalNumber **div**(RationalNumber r);

public boolean **lt**(RationalNumber r);

public boolean **gt**(RationalNumber r);

public boolean **eq**(RationalNumber r);

public boolean **le**(RationalNumber r);

public boolean **ge**(RationalNumber r);

public int **getNumerator**();

public int **geDenominator**();

}

**Test the program** with the following data, and print your output for each case

1/2 + 2/3

a + 3

1 - 3

1/2 + 1/2

4/6 \* 10

1/2 - 5/6

a/b / c

1/2 \* 1/3

1/2 / 1/2

1/2 < 5/9

1/2 > 1/4

1/2 <= 1

1/2 >= 1/2

1/2 = 1/3

16 >= 10

1 <= 1

5 = 4

1 v 5

1/2

See the attached Coding Styles for your program.

**Due Date** :

**LAB (you have to demonstrate to me that the lab is working)**

**Hand In and Grading Criteria:**

For grading submit the following items in a **closeable** large envelop (no folders) with your name clearly stated outside:

Everything must be typed - ***no handwritten external documentation will be accepted.***

* A cover sheet with your name, project number, and section
* Your program design – UML, Javadocs, pseudocode, etc **(hard copies**)
* A **hard copy** of the source files
* A **hard Copy** of the **test cases** stapled together with your name marked on top
* A **hard copy** of the program output
* You will be required to show me that the programs are working

The following criteria will be used to grade your work.

* Program correctness – program runs successfully and as expected
* Adherence to specifications – appropriate control structures, modular design, etc
* Documentation – internal and external, meaningful variable names, etc
* Neatness / Organization – indentation, block formatting, etc

See the evaluation sheet attached to these specifications.

**Coding Style Example:**

Note the indentation, white space, and meaningful names.

/\*\* Program CIS 225 Prog1

\* Course Title: Introduction to Computer Science

\* Course number: CIS 225-xxx

\* Instructor: Sara Wexler

\* @author Your Name

\* @version 1.0, 1/10/200x

\*

\* Description: Program CIS 225 Prog1

\* This class computes the area of a circle. The radius is hard coded in the class.

\*

\* Input: radius

\*

\* Compute: area

\* radius times radius times PI (3.14159)

\*

\* @author Your Name

\* @version 1.0, 9/6/2003

\*/

public class ComputeArea

{

/\*\* Main method \*/

public static void main(String[] args) {

double radius;

double area;

// Assign a radius

radius = 20;

// Calculate the area

area = radius \* radius \* 3.14159;

// Display results

System.out.println("The area for the circle of radius " +

radius + " is " + area);

} // end for main method

} // end for class ComputeArea