Southern Luzon State University
College Of Engineering
Computer Engineering Department

IMPLEMENTATION OF PROGRAMMING PROJECT

CpE05 - Object Oriented Programming
SY 2023-2024

Name: Dave Jhared G. Paduada   Date: April 29, 2024

Section/Schedule: IF T & TH 10:30 to 13:30   Score:______________

SOURCE CODE:

```
//PADUADA, Dave Jhared G.
// BSCpE II - IF
// OOP Method used: Encapsulation
// T & TH 10:30 - 13:30

import javax.swing.*;
import java.io.File;
import java.util.Scanner;
import java.io.PrintWriter;
import java.io.FileWriter;

public class mainsolverplus {
    public static void main(String[] args) throws Exception {
        new WelcomeMessage(); // Go to the WelcomeMessage class
    }
}

// The WelcomeMessage class will display a welcome message and ask the user to click
START or HISTORY or EXIT
class WelcomeMessage {
    WelcomeMessage() throws Exception {
        String[] options = { "START","HISTORY", "EXIT" };

        int choice = JOptionPane.showOptionDialog(null,
                "Welcome to the Main Solver+ !\nThis program will help you solve your right
triangle\n\nClick START to continue",
                "Main Solver+", JOptionPane.DEFAULT_OPTION,
JOptionPane.INFORMATION_MESSAGE, null, options, options[0]);
        if (choice == 0) {
            new Triangle(); // Go to the Triangle class
        } else if (choice == 1) {
           new HistoryReader();
        } else {
            System.exit(0);
        }
    }
}

// The Triangle class will ask the user to select the type of given values
class Triangle {
    Triangle() throws Exception {

        String[] choices = { "-- Please select --", "1 Side and 1 Angle", "2 Sides with or without
Angles"};

        String part = (String) JOptionPane.showInputDialog(null, "How many givens?",
                "Type of Given", JOptionPane.QUESTION_MESSAGE, null, choices, choices[0]);


        if ("-- Please select --".equals(part)) {
            new Triangle(); // Re-prompt.
        } else if ("1 Side and 1 Angle".equals(part)) {
            new OneAngleOneSide();
        } else if ("2 Sides with or without Angles".equals(part)) {
            new Sideside();
        } else {
            new WelcomeMessage();
        }
    }
}

class OneAngleOneSide { // Contains all the methods for solving a right triangle with one angle
and one side.

    // Initialize the sides and angles as -1 to indicate that they are missing
    // Why -1? Because the user can input 0 as a value for the sides and angles.
    // Also -1 acts like a placeholder to tell the compiler that the value is missing.
    private double sideA; // Opposite
    private double sideB; // Adjacent
    private double sideC; // Hypotenuse
    private double angleA;
    private double angleB;
    // angle C is given as 90 degrees since my topic is about 'Solutions of Right Triangle'

    // Constructor
    OneAngleOneSide() throws Exception {
        getInput();
        if (areAllSidesMissing()) {// If true, show an error message and re-prompt for input...
                                    If false, proceed to calculate the missing values
            JOptionPane.showMessageDialog(null, "At least one side must contain a value.", "Input
Error", JOptionPane.ERROR_MESSAGE);
            getInput();
        }
        checkForZeroValues();
        oneSideWithAngle();
        displayResults();
    }

    // Get the input from the user
    // The user can input -1 to indicate that the side is missing
    private void getInput() throws Exception {
        sideA = getDoubleFromInput("Enter the value of the opposite side (a): ", "Opposite Side");
        sideB = getDoubleFromInput("Enter the value of the adjacent side (b): ", "Adjacent Side");
        sideC = getDoubleFromInput("Enter the value of the hypotenuse (c): ", "Hypotenuse");
        angleA = getDoubleFromInput("Enter the value of angle A (degrees): ", "Angle A");
        angleB = getDoubleFromInput("Enter the value of angle B (degrees): ", "Angle B");
    }

    private double getDoubleFromInput(String message, String title) throws Exception {

        Object[] options = { "OK", "MISSING", "GO BACK", "Cancel" };
        int option = JOptionPane.showOptionDialog(null, message, title,
            JOptionPane.DEFAULT_OPTION, JOptionPane.PLAIN_MESSAGE, null, options,
            options[0]);

            if (option == 0) { // OK
            while (true) { // Loop until the user enters a valid value
                String input = JOptionPane.showInputDialog(null, "Enter a value:");
                if (input.equals("")) {
                    JOptionPane.showMessageDialog(null, "You must enter a value", "Error",
JOptionPane.ERROR_MESSAGE);
                    getInput(); // Re-prompt for input after showing the error message.
                } else {
                    return Double.parseDouble(input);
                }
            }
        } else if (option == 1) { // MISSING, when clicked it will input -1
            return -1;
        } else if (option == 2) { // GO BACK
            new Triangle();
            return -1;
        } else { // Cancel or close
            System.exit(0);
            return -1; // To avoid the error "missing return statement"
        }
    }
    // Check if any entered value is 0 and display an error message
    private void checkForZeroValues() throws Exception {
        if (sideA == 0 || sideB == 0 || sideC == 0 || angleA == 0 || angleB == 0) {
            JOptionPane.showMessageDialog(null, "Zero values are not allowed. Try again.", "Input
Error", JOptionPane.ERROR_MESSAGE);
            getInput(); // Re-prompt for input after showing the error message.
        }
    }

    // Check if all sides are marked as missing (-1)
    // If all sides are missing, the program will show an error message and re-prompt for input
    private boolean areAllSidesMissing() {
        return sideA == -1 && sideB == -1 && sideC == -1;
    }

    // this method will calculate the missing sides and angles
    private void oneSideWithAngle() throws Exception {

        // Check if the sides form a valid right triangle
        if (sideC != -1 && (sideA > sideC || sideB > sideC)) {
            JOptionPane.showMessageDialog(null, "The hypotenuse must be greater than the other
sides.", "Invalid Triangle", JOptionPane.ERROR_MESSAGE);
            getInput(); // Re-prompt for input after showing the error message.
        }

        // Now use trigonometry to find missing sides:

        // Case when hypotenuse is known
        if (angleA == -1 && angleB != -1) {
            angleA = 90 - angleB; // Since it's a right triangle
        } else if (angleB == -1 && angleA != -1) {
            angleB = 90 - angleA;
        }

        // Trigonometric calculations for known hypotenuse
        if (sideC != -1) {
            if (sideA == -1 && angleA != -1) { // Find opposite using sine
            sideA = Math.sin(Math.toRadians(angleA)) * sideC;
            }
            if (sideB == -1 && angleA != -1) { // Find adjacent using cosine
            sideB = Math.cos(Math.toRadians(angleA)) * sideC;
            }
        }

     // Trigonometric calculations for known adjacent (side B)
        if (sideB != -1) {
            if (sideC == -1 && angleA != -1) { // Find hypotenuse using cosine
            sideC = sideB / Math.cos(Math.toRadians(angleA));
            }
            if (sideA == -1 && angleA != -1) { // Find opposite using tangent
            sideA = Math.tan(Math.toRadians(angleA)) * sideB;
            }
        }

        // Trigonometric calculations for known opposite (side A)
        if (sideA != -1) {
            if (sideC == -1 && angleA != -1) { // Find hypotenuse using sine
            sideC = sideA / Math.sin(Math.toRadians(angleA));
            }
            if (sideB == -1 && angleA != -1) { // Find adjacent using tangent
            sideB = sideA / Math.tan(Math.toRadians(angleA));
            }
        }

        // Calculations when one angle (B) and its opposite side (B) is known:
        if (sideB != -1 && angleB != -1) {
            if (sideC == -1) { // Find hypotenuse using sine of angle B
            sideC = sideB / Math.sin(Math.toRadians(angleB));
            }
            if (sideA == -1) { // Find opposite using cosine of angle B (since angle B = 90 - angleA)
            sideA = Math.cos(Math.toRadians(angleB)) * sideB;
            }
        }

        // Round the values to two decimal places
        sideA = Math.round(sideA * 100.0) / 100.0;
        sideB = Math.round(sideB * 100.0) / 100.0;
        sideC = Math.round(sideC * 100.0) / 100.0;
        angleA = Math.round(angleA * 100.0) / 100.0;
        angleB = Math.round(angleB * 100.0) / 100.0;

        // If angle A or B is not present and angle B or A is present, we can calculate angle  A or B
using the formula:  90 - angle B or A
        if (angleA != -1 && angleB == -1) {
            angleB = 90 - angleA;
        } else if (angleB != -1 && angleA == -1) {
            angleA = 90 - angleB;
        }
        }



        // Display the results to the user
        private void displayResults() throws Exception {
        String results = "Results:\n\n" +
                "Side a (opposite): " + sideA + "\n" +
                "Side b (adjacent): " + sideB + "\n" +
                "Side c (hypotenuse): " + sideC + "\n\n" +
                "Angle A: " + angleA + " degrees\n" + // Assume angleA is already in degrees
                "Angle B: " + angleB + " degrees\n" + // Assume angleB is already in degrees
                "Angle C: 90.0 degrees\n" +
                "Total Angle: " + (angleA + angleB + 90.0) + " degrees\n\n";

            Object[] options = { "OK", "SAVE", "HOME" };
            int option = JOptionPane.showOptionDialog(null, results, "Calculated Values",
                JOptionPane.DEFAULT_OPTION, JOptionPane.INFORMATION_MESSAGE, null,
                options, options[0]);

            if (option == 0) {
            int calculateAgainOption = JOptionPane.showOptionDialog(null, "Would you
like to calculate again?", "Calculate again?",
JOptionPane.YES_NO_OPTIONJOptionPane.QUESTION_MESSAGE, null, null,null);
                if (calculateAgainOption == JOptionPane.YES_OPTION) {
                new Triangle();
                } else {
                JOptionPane.showMessageDialog(null, "Happy coding!", "Cheers",
JOptionPane.INFORMATION_MESSAGE);
                System.exit(0);
                }
            } else if (option == 1) { // SAVE selected
                saveResultsOutside();
            } else if (option == 2) { // Return to Welcome Message
                new WelcomeMessage();
            }
        }

    // Save the results to a file
    // The file will be stacked with the results of each calculation
    // with the use of the PrintWriter class to write to a file.
    // If I use FileWriter, it will overwrite the file each time the program is run
    private void saveResultsOutside() throws Exception {
        File file = new File("One Side One Angle.txt");


        // When you're writing to a file, you often wrap PrintWriter around a FileWriter to handle file
creation and character encoding
        // The second argument true in FileWriter constructor is for enabling the append mode,
which allows you to add to the existing content of the file instead of overwriting it.
        PrintWriter writer = new PrintWriter(new FileWriter(file, true));
        writer.println("Hypotenuse: " + sideC);
        writer.println("Adjacent: " + sideB);
        writer.println("Opposite: " + sideA + "\n");
        writer.println("Angle A: " + angleA + " degrees");
        writer.println("Angle B: " + angleB + " degrees\n");
        writer.println("Total Angles: " + (angleA + angleB + 90.0) + "degrees\n");
        writer.println("Last updated on: " + new java.util.Date());
        writer.println(); // Adds a newline for separation between entries

writer.println("================================================================");


        writer.close(); // After writing data, I should close the PrintWriter object to release the
resources.


        JOptionPane.showMessageDialog(null, "Results have been saved to \"One Side One
Angle\".txt",
                "Results Saved", JOptionPane.INFORMATION_MESSAGE);
        int option = JOptionPane.showOptionDialog(null, "Would you like to calculate again?",
"Calculate again?",
                JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE, null, null,
null);
        if (option == JOptionPane.YES_OPTION) {
            new Triangle();
        } else {
            JOptionPane.showMessageDialog(null, "Happy coding!", "Cheers",
JOptionPane.INFORMATION_MESSAGE);
        }
    }
}

// The Sideside class will calculate the missing sides and angles of a right triangle with two sides
given
// Same as the OneAngleOneSide class, but this time, we have two sides given,
// the only difference in this Sideside class is the method used to calculate the missing values.

class Sideside {

    // Initialize the sides and angles as -1 to indicate that they are missing

    private double sideA; // Opposite
    private double sideB; // Adjacent
    private double sideC; // Hypotenuse
    private double angleA;
    private double angleB;
// angle C is given as 90 degrees since my topic is about 'Solutions of Right
// Triangle'

    Sideside() throws Exception {
        getInput();
        if (areAllSidesMissing()) { // If true, show an error message and re-prompt for input... If
false, proceed
```

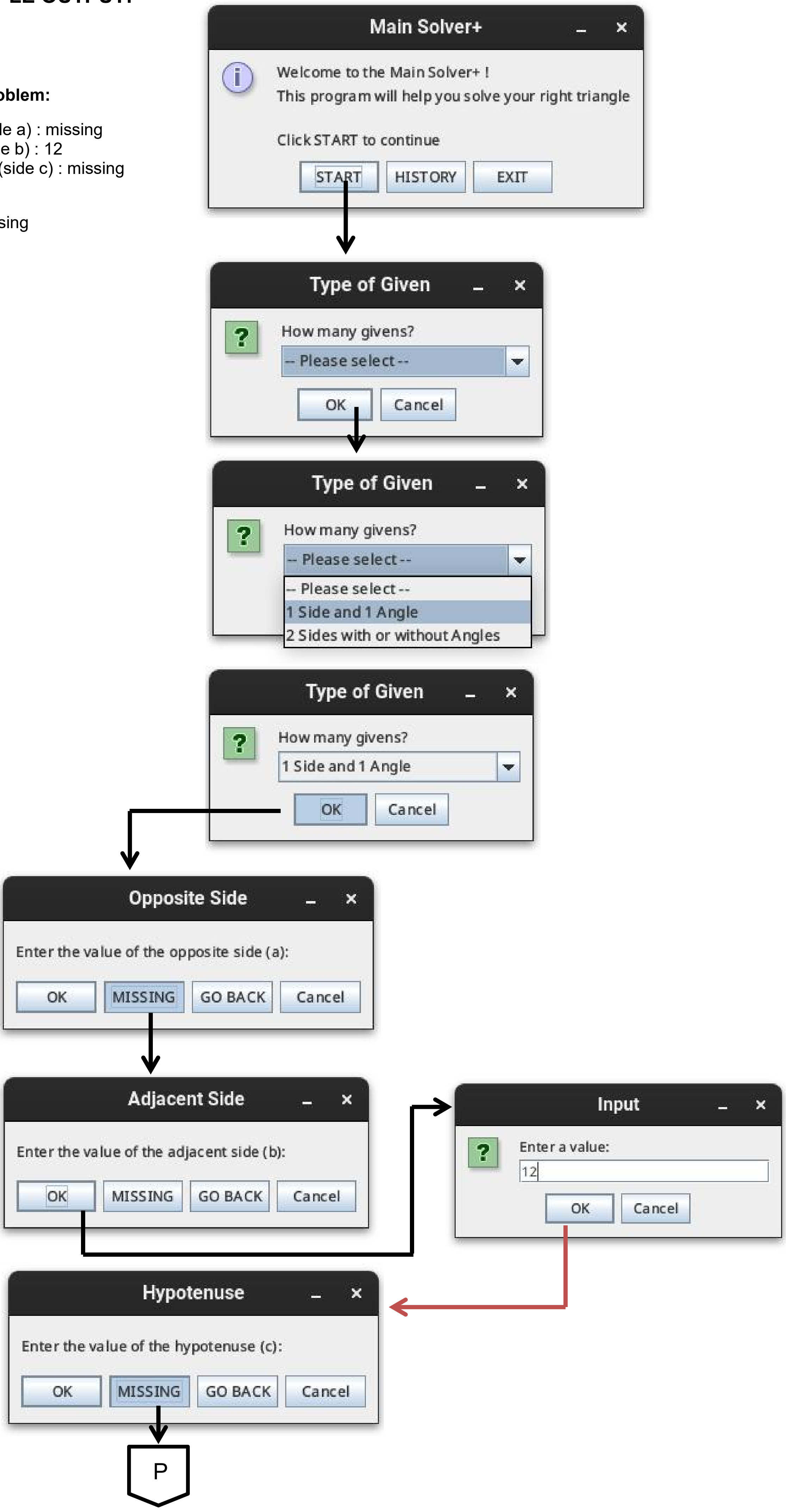
```
                                    // to calculate the missing values
            JOptionPane.showMessageDialog(null, "At least one side must contain a value.", "Input
Error",
                    JOptionPane.ERROR_MESSAGE);
            getInput(); // Re-prompt for input after showing the error message.
        }
        checkForZeroValues();
        twoSidesWithorWithoutAngle();
        displayResults();
    }

    private void getInput() throws Exception {
        sideA = getDoubleFromInput("Enter the value of the opposite side (a): ", "Opposite Side");
        sideB = getDoubleFromInput("Enter the value of the adjacent side (b): ", "Adjacent Side");
        sideC = getDoubleFromInput("Enter the value of the hypotenuse (c): ", "Hypotenuse");
        angleA = getDoubleFromInput("Enter the value of angle A (degrees): ", "Angle A");
        angleB = getDoubleFromInput("Enter the value of angle B (degrees): ", "Angle B");
    }

    private double getDoubleFromInput(String message, String title) throws Exception {
        Object[] options = { "OK", "MISSING", "GO BACK", "Cancel" };

        int option = JOptionPane.showOptionDialog(null, message, title,
JOptionPane.DEFAULT_OPTION,
                JOptionPane.PLAIN_MESSAGE, null, options, options[0]);
        if (option == 0) { // OK
            while (true) {
                String input = JOptionPane.showInputDialog(null, "Enter a value:");
                if (input.equals("")) {
                    JOptionPane.showMessageDialog(null, "You must enter a value", "Error",
JOptionPane.ERROR_MESSAGE);
                    getInput(); // Re-prompt for input after showing the error message.
                } else {
                    return Double.parseDouble(input);
                }
            }
        } else if (option == 1) { // MISSING, when clicked it will input -1
            return -1;
        } else if (option == 2) { // GO BACK
            new Triangle();
            return -1;
        } else { // Cancel or close
            System.exit(0);
            return -1;
        }
    }
    // Check if any entered value is 0 and display an error message
    private void checkForZeroValues() throws Exception {
        if (sideA == 0 || sideB == 0 || sideC == 0 || angleA == 0 || angleB == 0) {
            JOptionPane.showMessageDialog(null, "Zero values are not allowed. Try again.", "Input
Error", JOptionPane.ERROR_MESSAGE);
            getInput(); // Re-prompt for input after showing the error message.
        }
    }

    // Check if all sides are marked as missing (-1)
    private boolean areAllSidesMissing() {
        return sideA == -1 && sideB == -1 && sideC == -1;
    }

    private void twoSidesWithorWithoutAngle() throws Exception{

        // Check if the sides form a valid right triangle
        if (sideC != -1 && (sideA > sideC || sideB > sideC)) {
            JOptionPane.showMessageDialog(null, "The hypotenuse must be greater than the other
sides.", "Invalid Triangle", JOptionPane.ERROR_MESSAGE);
            getInput(); // Re-prompt for input after showing the error message.
        }

        //iF the user enters a value of -1 or missing, the program will calculate the missing    value
and it will use Pythagorean theorem.
          != means if the side# has values.

        // Pythagorean theorem
        if (sideA == -1 && sideB != -1 && sideC != -1) {
            sideA = Math.round(Math.sqrt(Math.pow(sideC, 2) - Math.pow(sideB, 2)) * 100) / 100.0;
// We use -(negative) here because we transposed the formula

        } else if (sideB == -1 && sideA != -1 && sideC != -1) {
            sideB = Math.round(Math.sqrt(Math.pow(sideC, 2) - Math.pow(sideA, 2)) * 100) / 100.0;

        } else if (sideC == -1 && sideA != -1 && sideB != -1) {
             /* If side C is missing, Calculate sideC using the Pythagorean theorem
             By adding the calculation for side C as shown, the method now covers the case where
side C is missing and ensures that all sides and angles can be determined from just two known
sides. */
            sideC = Math.round(Math.sqrt(Math.pow(sideA, 2) + Math.pow(sideB, 2)) * 100) / 100.0;
// Standard a^2 + b^2 = c^2 then square root to get c
         }

        // After calculating the missing side, we can calculate the missing angle using
trigonometric functions.
        // TOA Tangent = Opposite side A /Adjacent side B
        // I will use TOA since I have the opposite and adjacent sides.
        // I will use Math.atan to get the angle in radians then convert it to degrees using
Math.toDegrees.

        // Availability of Sides: If two sides of a right triangle are known and they are the opposite
and
        // adjacent sides relative to one of the non-right angles, then the tangent function is the
direct choice because it uses those two sides.

        if (angleA == -1 && sideA != -1 && sideB != -1) { // If angle A is missing and side A and side
B are present, we can calculate angle A
            angleA = Math.round(Math.toDegrees(Math.atan(sideA / sideB)) * 100) / 100.0;
            angleB = Math.round((90 - angleA) * 100) / 100.0;
        }
        if (angleB == -1 && sideA != -1 && sideB != -1) { // If angle B is missing and side A and side
B are present, we can calculate angle B
            angleB = Math.round(Math.toDegrees(Math.atan(sideB / sideA)) * 100) / 100.0;
            angleA = Math.round((90 - angleB) * 100) / 100.0;
        }

        // If angle A or B is not present and angle B or A is present, we can calculate angle A or B
using the formula:  90 - angle B or A
        if (angleA != -1 && angleB == -1) {
            angleB = 90 - angleA;
        } else if (angleB != -1 && angleA == -1) {
            angleA = 90 - angleB;
        }
    }

    private void displayResults() throws Exception {
        String results = "Results:\n\n" +
                "Side a (opposite): " + sideA + "\n" +
                "Side b (adjacent): " + sideB + "\n" +
                "Side c (hypotenuse): " + sideC + "\n\n" +
                "Angle A: " + angleA + " degrees\n" +
                "Angle B: " + angleB + " degrees\n" +
                "Angle C: 90.00 degrees\n" +
                "Total Angle: " + (angleA + angleB + 90.0) + " degrees\n\n";


        Object[] options = { "OK", "SAVE", "HOME" };
        int option = JOptionPane.showOptionDialog(null, results, "Calculated Values",
            JOptionPane.DEFAULT_OPTION, JOptionPane.INFORMATION_MESSAGE, null,
options, options[0]);

        if (option == 0) {
            int calculateAgainOption = JOptionPane.showOptionDialog(null, "Would you like to
calculate again?", "Calculate again?",
            JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE, null, null, null);
            if (calculateAgainOption == JOptionPane.YES_OPTION) {
            new Triangle();
            } else {
            JOptionPane.showMessageDialog(null, "Happy coding!", "Cheers",
JOptionPane.INFORMATION_MESSAGE);
            System.exit(0);
            }
        } else if (option == 1) { // SAVE selected
            saveResultsOutside();
        } else if (option == 2) { // Returns to Home page
            new WelcomeMessage();
        }
    }

    private void saveResultsOutside() throws Exception {
        File file = new File("Two sides with or without Angles.txt");

        PrintWriter writer = new PrintWriter(new FileWriter(file, true));
        writer.println("Hypotenuse: " + sideC);
        writer.println("Adjacent: " + sideB);
        writer.println("Opposite: " + sideA + "\n");
        writer.println("Angle A: " + angleA + " degrees");
        writer.println("Angle B: " + angleB + " degrees\n");
        writer.println("Total Angles: " + (angleA + angleB + 90.0) + "degrees\n");
        writer.println("Last updated on: " + new java.util.Date());
        writer.println(); // Adds a newline for separation between entries

writer.println("====================================================================");

        writer.close();
        JOptionPane.showMessageDialog(null, "Results have been saved to \"Two sides with or
without Angles.txt\"",
                "Results Saved", JOptionPane.INFORMATION_MESSAGE);
        int option = JOptionPane.showOptionDialog(null, "Would you like to calculate again?",
"Calculate again?", JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE,
null, null, null);
        if (option == JOptionPane.YES_OPTION) {
            new Triangle();
        } else {
            JOptionPane.showMessageDialog(null, "Happy coding!", "Cheers",
JOptionPane.INFORMATION_MESSAGE);
        }
    }
}

// The HistoryReader class will read the history of the calculations made by the user
// and produces the output in a JOptionPane dialog box.
class HistoryReader {
    HistoryReader() throws Exception {

        int choice = JOptionPane.showOptionDialog(null, "Which history would you like to read?", 
"History",
                JOptionPane.DEFAULT_OPTION, JOptionPane.QUESTION_MESSAGE, null,
                new String[] {"One Side One Angle", "Two sides with or without Angles", "GO BACK" , 
"EXIT" }, "One Side One Angle");

        if (choice == 0) {
            File file = new File("One Side One Angle.txt");
            Scanner scanner = new Scanner(file);
            String history = ""; // Initialize an empty string to store the history
            while (scanner.hasNextLine()) {
                history += scanner.nextLine() + "\n"; // Read the file line by line and concatenate the 
lines
            }
            scanner.close();
            JOptionPane.showMessageDialog(null, history, "One Side One Angle History", 
JOptionPane.INFORMATION_MESSAGE);
            new WelcomeMessage(); // Go back to WelcomeMessage class

        } else if (choice == 1) {
            File file = new File("Two sides with or without Angles.txt");
            Scanner scanner = new Scanner(file);
            String history = "";
            while (scanner.hasNextLine()) {
                history += scanner.nextLine() + "\n";
            }
            scanner.close();
            JOptionPane.showMessageDialog(null, history, "Two sides with or without Angles 
History", JOptionPane.INFORMATION_MESSAGE);
            new HistoryReader();

        } else if (choice == 2) {
            new WelcomeMessage();
        } else {
            System.exit(0);
        }
    }
}
```

SAMPLE OUTPUT:

Example Problem:

Opposite (side a) : missing
Adjacent (side b) : 12
Hypotenuse (side c) : missing

Angle A: 30
Angle B: Missing



Main Solver+

Welcome to the Main Solver+ !
This program will help you solve your right triangle

Click START to continue

Type of Given

How many givens?
-- Please select --

Type of Given

How many givens?
-- Please select --
-- Please select --
1 Side and 1 Angle
2 Sides with or without Angles

Type of Given

How many givens?
1 Side and 1 Angle

Opposite Side

Enter the value of the opposite side (a):

Adjacent Side

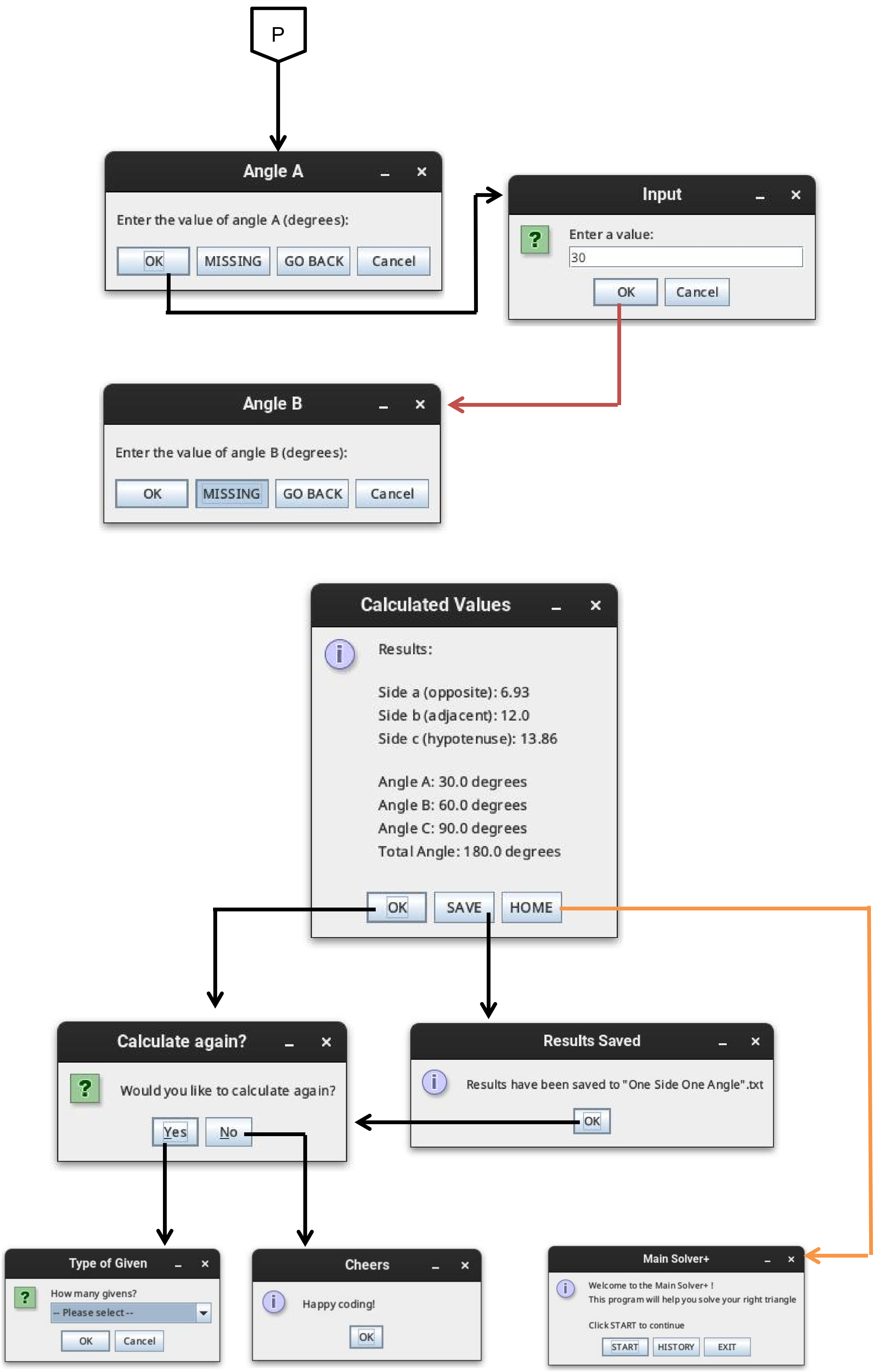
Enter the value of the adjacent side (b):

Input

Enter a value:
12

Hypotenuse

Enter the value of the hypotenuse (c):



Enter the value of angle A (degrees):

Enter a value:
30

Enter the value of angle B (degrees):

```
Results:

Side a (opposite): 6.93
Side b (adjacent): 12.0
Side c (hypotenuse): 13.86

Angle A: 30.0 degrees
Angle B: 60.0 degrees
Angle C: 90.0 degrees
Total Angle: 180.0 degrees
```

Would you like to calculate again?

Results have been saved to "One Side One Angle".txt

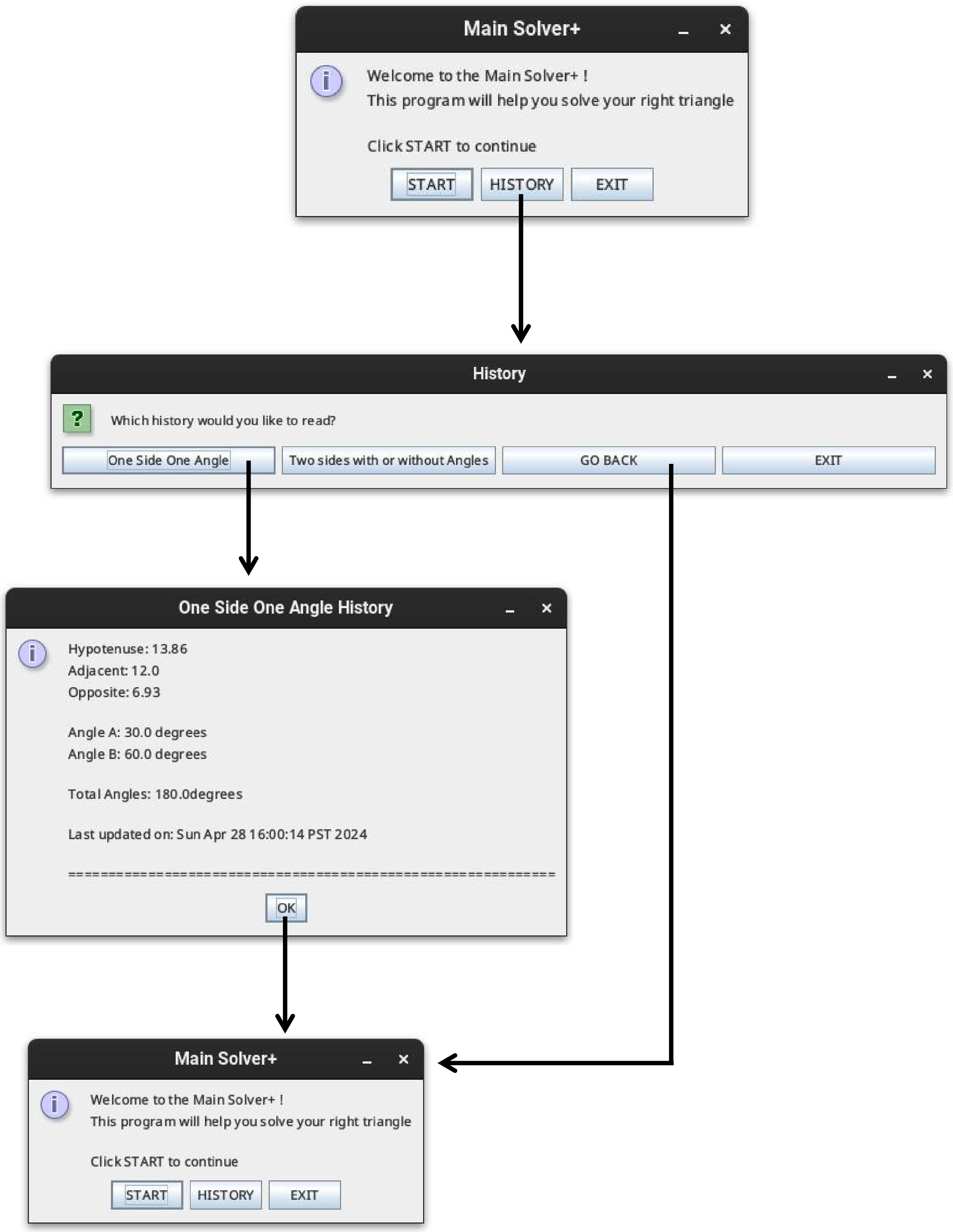
How many givens?

Happy coding!

Welcome to the Main Solver+ !
This program will help you solve your right triangle

Click START to continue

When viewing the history for One side One Angle:



Welcome to the Main Solver+ !
This program will help you solve your right triangle

Click START to continue

Which history would you like to read?

```
Hypotenuse: 13.86
Adjacent: 12.0
Opposite: 6.93

Angle A: 30.0 degrees
Angle B: 60.0 degrees

Total Angles: 180.0degrees

Last updated on: Sun Apr 28 16:00:14 PST 2024

=============================================================
```

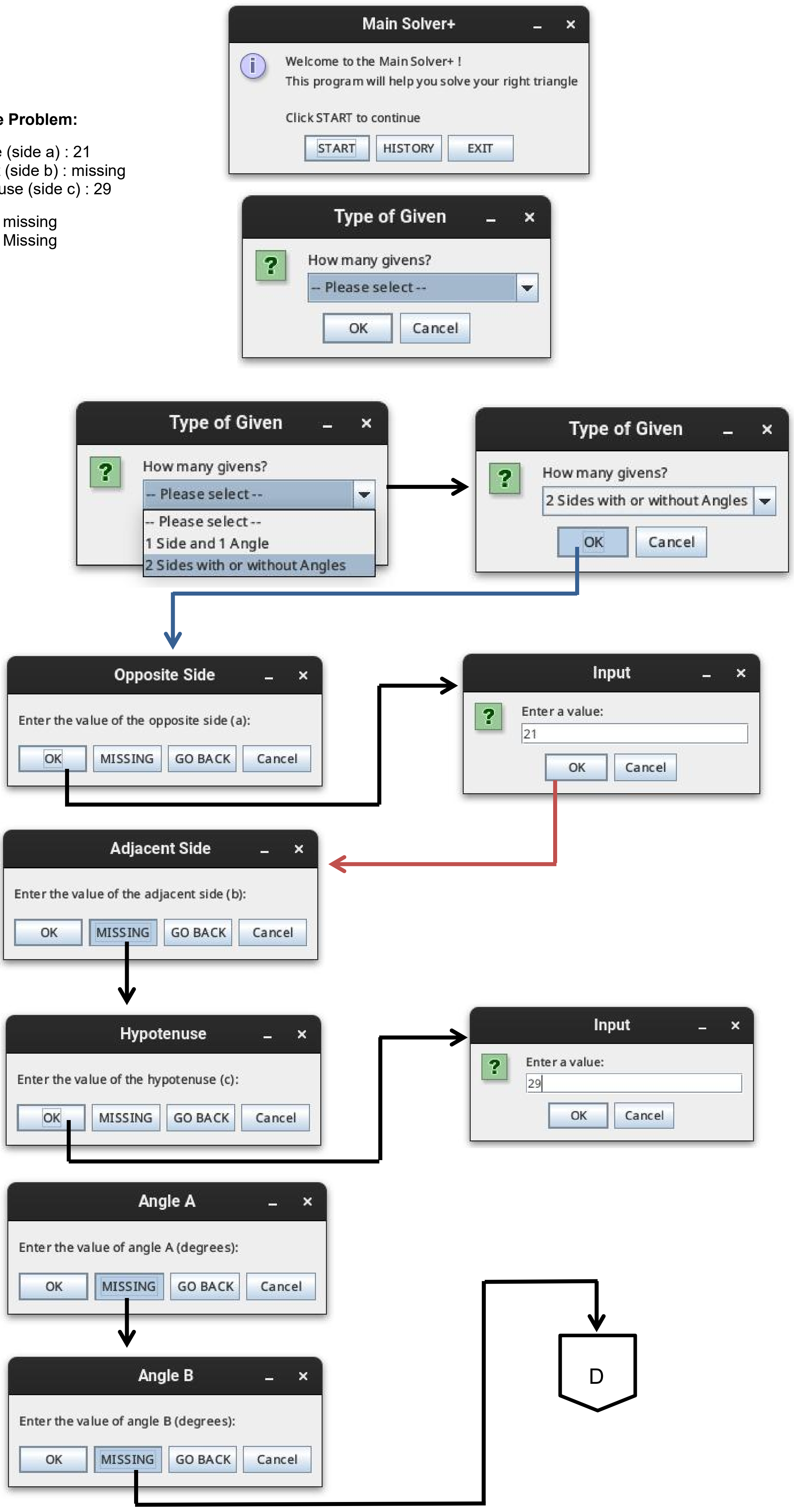
Welcome to the Main Solver+ !
This program will help you solve your right triangle

Click START to continue

Example Problem:

Opposite (side a) : 21
Adjacent (side b) : missing
Hypotenuse (side c) : 29

Angle A: missing
Angle B: Missing


Main Solver+

Welcome to the Main Solver+ !
This program will help you solve your right triangle

Click START to continue

Type of Given

How many givens?
-- Please select --

Type of Given

How many givens?
-- Please select --
-- Please select --
1 Side and 1 Angle
2 Sides with or without Angles

Type of Given

How many givens?
2 Sides with or without Angles

Opposite Side

Enter the value of the opposite side (a):

Input

Enter a value:
21

Adjacent Side

Enter the value of the adjacent side (b):

Hypotenuse

Enter the value of the hypotenuse (c):

Input

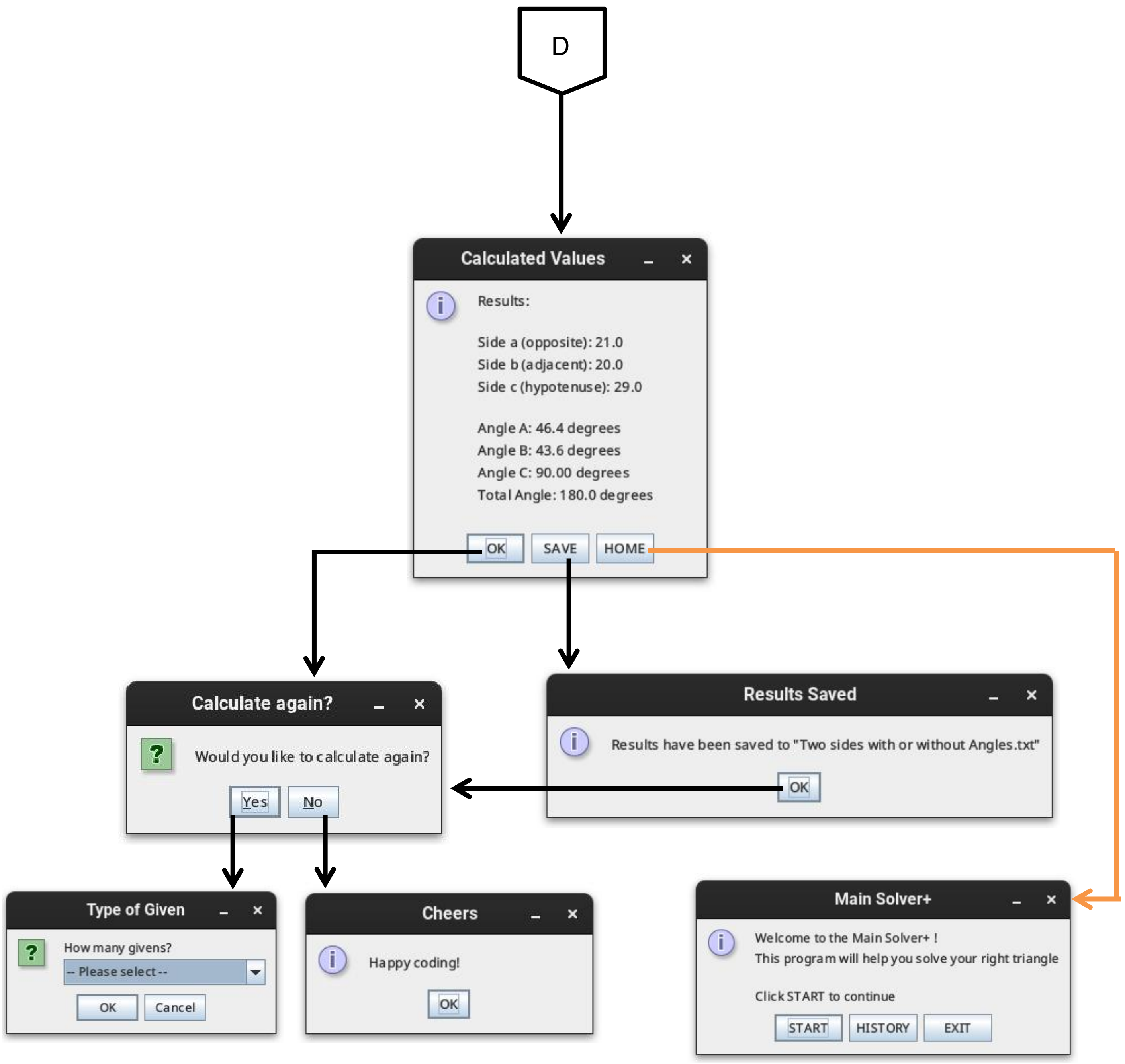
Enter a value:
29

Angle A

Enter the value of angle A (degrees):

Angle B

Enter the value of angle B (degrees):



Results:

Side a (opposite): 21.0
Side b (adjacent): 20.0
Side c (hypotenuse): 29.0

Angle A: 46.4 degrees
Angle B: 43.6 degrees
Angle C: 90.00 degrees
Total Angle: 180.0 degrees

Would you like to calculate again?

Results have been saved to "Two sides with or without Angles.txt"

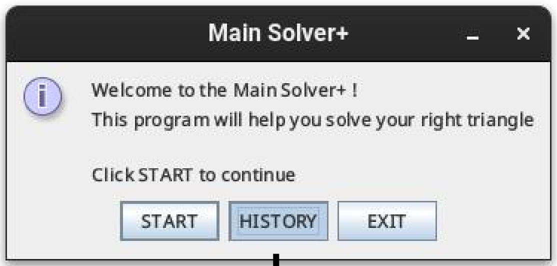
How many givens?
-- Please select --

Happy coding!

Welcome to the Main Solver+ !
This program will help you solve your right triangle

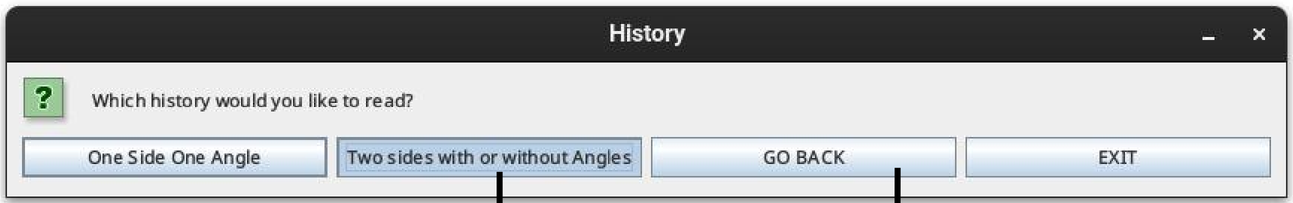
Click START to continue

Viewing history for Two Sides with or Without angles:

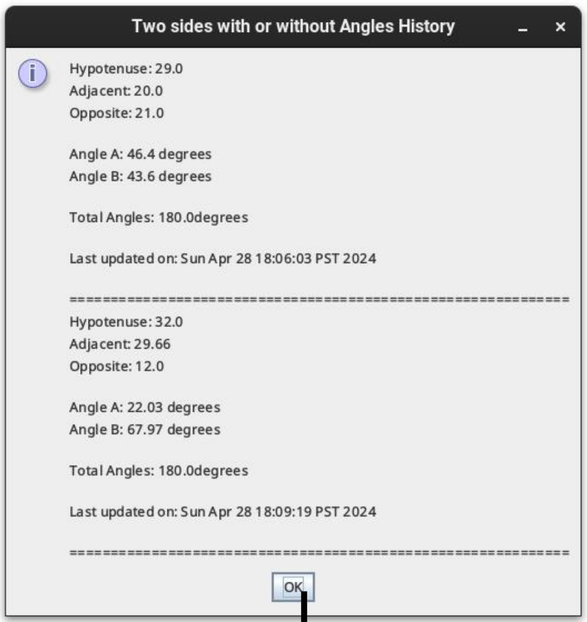


Welcome to the Main Solver+ !
This program will help you solve your right triangle

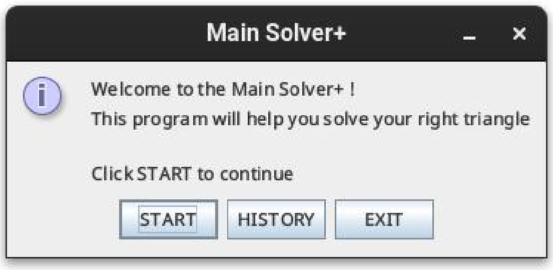
Click START to continue



Which history would you like to read?



```
Hypotenuse: 29.0
Adjacent: 20.0
Opposite: 21.0

Angle A: 46.4 degrees
Angle B: 43.6 degrees

Total Angles: 180.0degrees

Last updated on: Sun Apr 28 18:06:03 PST 2024

==========================================================
Hypotenuse: 32.0
Adjacent: 29.66
Opposite: 12.0

Angle A: 22.03 degrees
Angle B: 67.97 degrees

Total Angles: 180.0degrees

Last updated on: Sun Apr 28 18:09:19 PST 2024

==========================================================
```



Welcome to the Main Solver+ !
This program will help you solve your right triangle

Click START to continue