OOP

Subject Code: SEHH2242
Subject Title: OBJECT ORIENTED PROGRAMMING
Class: 101C

Student ID: 20190875A
Student Name: KOK Siu Chung

Source Code

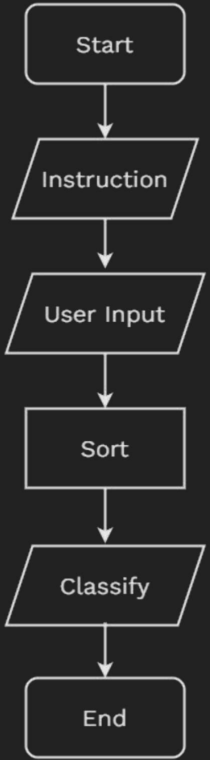
```java
/*
SEHH2242 OBJECT ORIENTED PROGRAMMING Assignment One
Stundent Name         Student Number Tutorial Group
KOK Siu Chung           20190875A        101C

2021/9/17
*/
import java.util.Scanner;

public class TriangleChecker_20190875A {
    static int side1, side2, side3; // User inputs

    public static void main(String args[]) {

        InstructionPrint(); // Print Instruction Message

        System.out.print("Please enter the first side of the triangle  : "); // Prompt for input
        side1 = userinput(); // user input side1
        System.out.print("Please enter the second side of the triangle : "); // Prompt for input
        side2 = userinput(); // user input side2
        System.out.print("Please enter the third side of the triangle  : "); // Prompt for input
        side3 = userinput(); // user input side3

        Sort(); // Sort the three inputs, descending order

        // Display Result
        System.out.printf("%S %n", "\n -- Result --");
        System.out.printf("Sides (descending order): %d %d %d%n", side1, side2, side3);

        Classify(); // Classify the triangle
        System.out.println(""); // Console spacing
    }

    public static void InstructionPrint() { // Print out instruction message
        System.out.print("\033[H\033[2J"); // Clear the console
        System.out.flush(); // Clear the console

        // Print out message
        System.out.printf("%S %n", "==========TRAIANGLE CHECKER==========");
        System.out.println("");
        System.out.println("         -- How to use --");
        System.out.printf("1. ONLY integer values are accepted.%n2. Type the three inputs one by one.%n");
        System.out.println("");
    }

    public static int userinput() { // User input validation
        while (true) { // While the input is not positive integer
            Scanner input = new Scanner(System.in);
            if (input.hasNextInt()) { // Check the input to see it is int or not
                int temp = input.nextInt(); // Save the int input to a temporary variable
                if (temp > 0) { // Check the input to see it is positive integer value or not
                    return temp; // Return the value
                } else { // The input is non-positive integer
                    System.out.print("Please a positive integer value: "); // Prompt for input
                }
            } else { // The input is not integer
                System.out.print("Please a positive integer value: "); // Prompt for input
            }
        }
    }
```

```java
public static void Classify() { // Classify the triangle
    if (TriangleCheck() == true) { // Check whether can these three sides can form a triangle
        if (EquilateralCheck() == false) { // Check whether the triangle is equilateral
            if (IsoscelesCheck() == false) { // Check whether the triangle is isosceles
                RightAngledCheck(); // Check whether the triangle is right-angled
            }
        }
    }
}

public static void RightAngledCheck() { // Check whether the triangle is right-angled
    if (side1 * side1 == (side2 * side2 + side3 * side3)) { // Determine whether the triangle is right-angled
        System.out.println("These three sides could form a Right-angled Triangle and Scalene Triangle"); // The triangle is Right-angled and Scalene
    } else { // The triangle is not right-angled
        System.out.println("These three sides could form a Scalene Triangle"); // The triangle is Scalene

    }
}

public static boolean IsoscelesCheck() { // Check whether the triangle is isosceles
    if ((side1 == side2) || (side1 == side2) || (side2 == side3)) { // Determine whether the triangle is isosceles
        System.out.println("These three sides could form a Isosceles Triangle");// The triangle is isosceles
        return true;
    } else { // The triangle is not isosceles
        return false;
    }
}

public static boolean EquilateralCheck() { // Check whether the triangle is equilateral
    if ((side1 == side2) && (side2 == side3)) { // Determine whether the triangle is equilateral
        System.out.println("These three sides could form a Equilateral Triangle.");// The triangle is equilateral
        return true;
    } else { // The triangle is not equilateral
        return false;
    }
}
```

```java
    public static boolean TriangleCheck() { // Check whether can these three sides can form a triangle
        if (side1 >= (side2 + side3)) { // Determine whether can these three sides can form a triangle
            System.out.println("These three sides could not form any triangle."); // These three sides can form a triangle
            return false;
        } else { // These three sides can not form a triangle
            return true;
        }
    }

    public static void Sort() { // Sort the three inputs, descending order
        int temp; // Temporary variable

        // Determine whether side1 or side2 is larger
        if (side1 < side2) {
            temp = side1;
            side1 = side2;
            side2 = temp;
        }

        // Determine whether side1 or side3 is larger
        if (side1 < side3) {
            temp = side1;
            side1 = side3;
            side3 = temp;
        }

        // Determine whether side2 or side3 is larger
        if (side2 < side3) {
            temp = side2;
            side2 = side3;
            side3 = temp;
        }
    }
}
```

Explanations

Program Structure



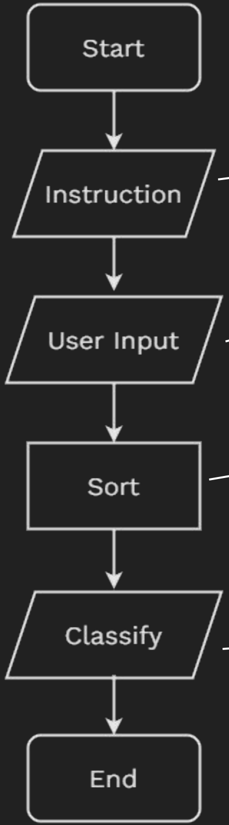
Instruction for user to use this triangle checker

User input the three sides of the triangle

Sort the three sides in descending order

Classify the triangle and print out the result

Program Structure



Code

```
public static void main(String args[]) {

        InstructionPrint(); // Print Instruction Message

        System.out.print("Please enter the first side of the triangle  : "); // Prompt for input
        side1 = userinput(); // user input side1
        System.out.print("Please enter the second side of the triangle : "); // Prompt for input
        side2 = userinput(); // user input side2
        System.out.print("Please enter the third side of the triangle  : "); // Prompt for input
        side3 = userinput(); // user input side3

        Sort(); // Sort the three inputs, descending order

        // Display Result
        System.out.printf("%S %n", "\n -- Result --");
        System.out.printf("Sides (descending order): %d %d %d%n", side1, side2, side3);

        Classify(); // Classify the triangle

        System.out.println(""); // Console spacing
    }
```

Instruction

Code

```
public class TriangleChecker_20190875A  {
          ⋮
    public static void main(String args[]) {
         ⋮
         InstructionPrint (); //Print Instruction Message
         ⋮
    }
}
```

```
public static void InstructionPrint () { // Print out instruction message
    System.out.print("\033[H\033[2J"); // Clear the console
    System.out.flush(); // Clear the console

    // Print out message
    System.out.printf("%S %n", "==========TRAIANGLE CHECKER==========  ");
    System.out.println("");
    System.out.println("         -- How to use -- ");
    System.out.printf("1. ONLY integer values are accepted.%n2. Type the three inputs one by one.%n   ");
    System.out.println("");
}
```

Description

`InstructionPrint` is a function for printing out the instruction about how to use this triangle checker.

Instruction

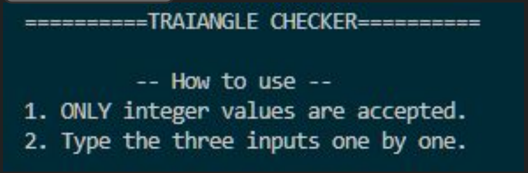
```java
System.out.print("\033[H\033[2J"); // Clear the console
System.out.flush(); // Clear the console
```

All Human-computer interaction of this program are using the console. In order to enhance the user experience, before printing any instruction to the console. The console will earse all the past messages.

```java
// Print out message
System.out.printf("%S %n", "==========TRAIANGLE CHECKER==========  ");
System.out.println("");
System.out.println("          -- How to use -- ");
System.out.printf("1. ONLY integer values are accepted.%n2. Type the three inputs one by one.%n   ");
System.out.println("");
```

After clear the console, Instruction will be printed out to the console. Since our program only accepts specific types of inputs (positive integer), we have to tell the user "what" they can input and "how" they can input.

Output



```
==========TRAIANGLE CHECKER==========

         -- How to use --
1. ONLY integer values are accepted.
2. Type the three inputs one by one.
```

User Input

Code

```java
public class TriangleChecker_20190875A  {
    static int side1, side2, side3;  // User inputs
    public static void main(String args[]) {
        ⋮
          System.out.print("Please enter the first side of the triangle  :   "); // Prompt for input
          side1  = userinput(); // user input side1
          System.out.print("Please enter the second side of the triangle :   "); // Prompt for input
          side2  = userinput(); // user input side2
          System.out.print("Please enter the third side of the triangle  :   "); // Prompt for input
          side3  = userinput(); // user input side3
        ⋮
    }
}
```

Description

`side1, side2, side3` are the variables for storing user input of the three sides. Since we only accept integer values, we can just set the variable type to int.

The program will prompt for input first. Then store user inputs into the three variables.

`userinput` is a function for getting user input with validation. It will return the integer value that the user has input, so we just put the value straight into the three variables mentioned above.

User Input

Code

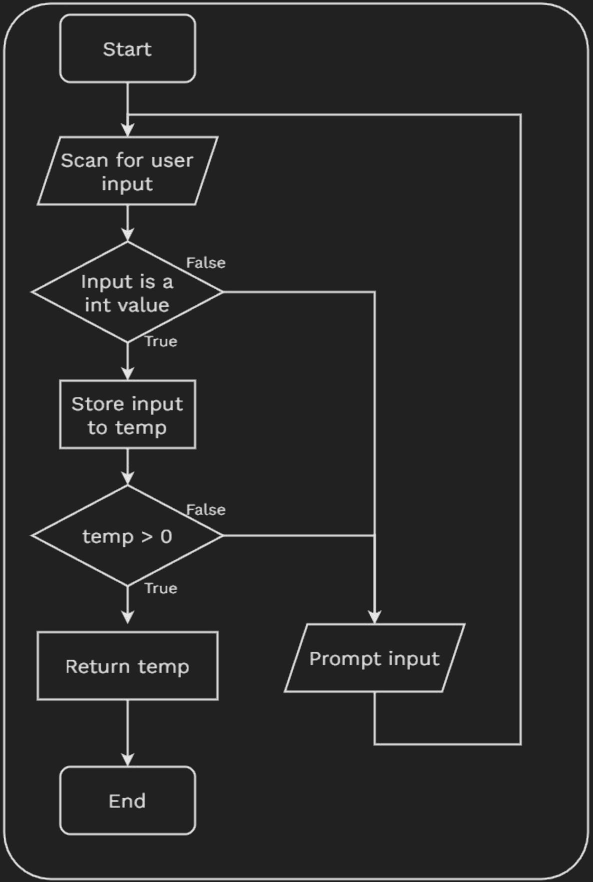
```java
public static int userinput () { // User input validation
    while (true) {  // While the input is not positive integer
        Scanner input = new Scanner(System.in);
        if (input.hasNextInt()) { // Check the input to see it is int or not
            int temp = input.nextInt(); // Save the int input to a temporary variable
            if (temp > 0) { // Check the input to see it is positive integer value or not
                return temp;  // Return the value
            } else {  // The input is non-positive integer
                System.out.print("Please a positive integer value:  "); // Prompt for input
            }
        } else {  // The input is not integer
            System.out.print("Please a positive integer value:  "); // Prompt for input
        }
    }
}
```

Description

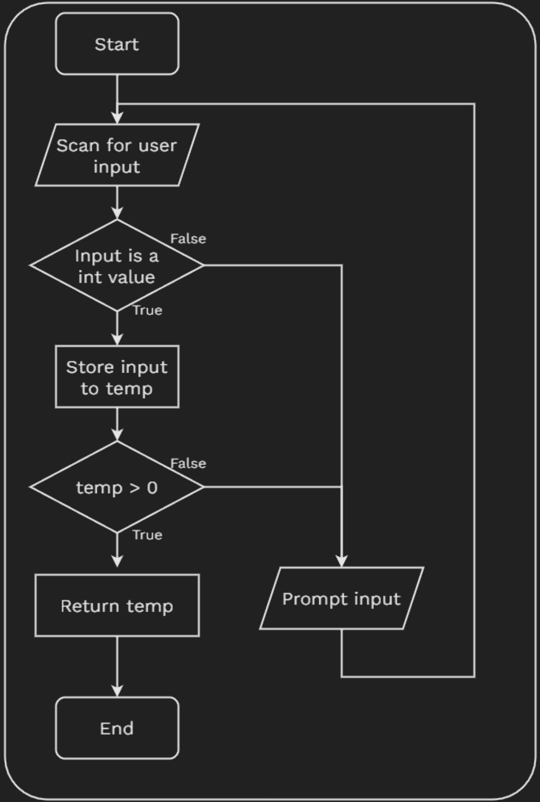
`userinput` is a function for getting user input with validation.

Since we only accept positive integer values from the user. We have to run through validation for all inputs. The program will keep asking users for acceptable inputs, if the user has input not acceptable inputs.

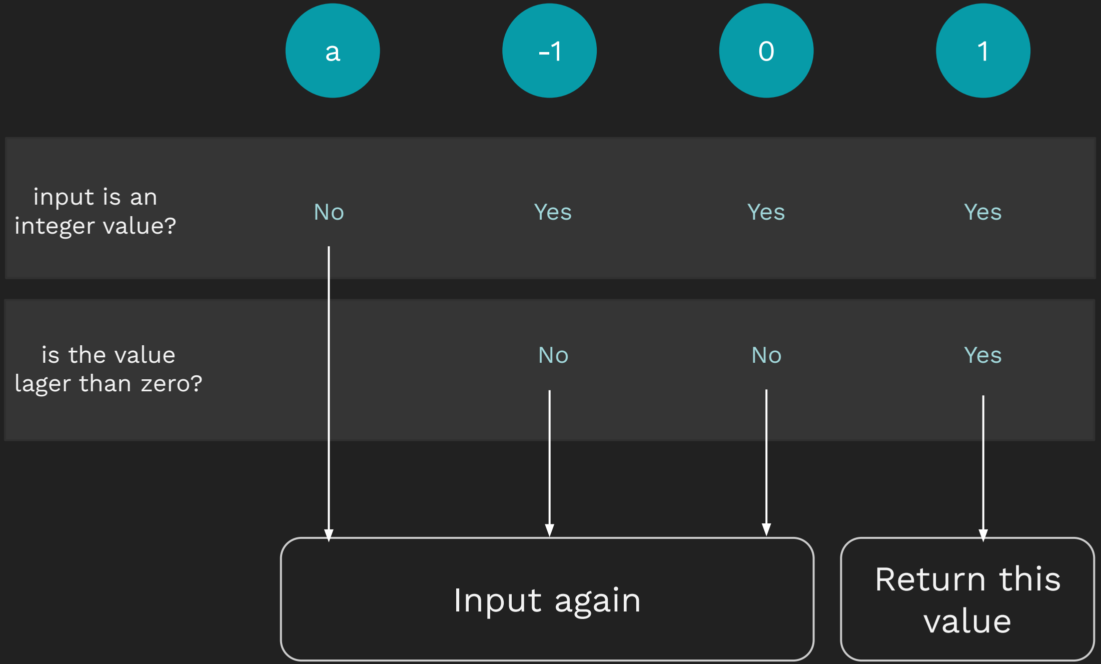
After the validation, the function will return the verified input value.



User Input



Demonstrate the validation with some different input.



After the validation, only positive integer values would be left.

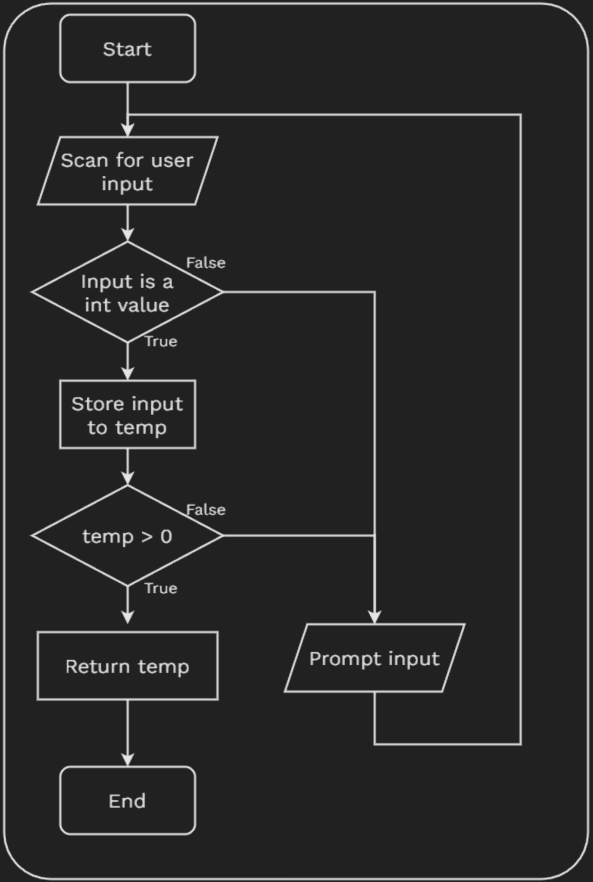
User Input

```
⋮
49     while (true) {  // While the input is not positive integer
50         Scanner input = new Scanner(System.in);
51         if (input.hasNextInt()) {  // Check the input to see it is int or not
52             int temp = input.nextInt(); // Save the int input to a temporary variable
⋮
58         } else {  // The input is not integer
59             System.out.print("Please a positive integer value:  "); // Prompt for input
60         }
61     }
⋮
```

Start the scanner at line 50, scan for user input.
if the next input is an integer, store the value to *“temp”*, then execute line 53-57
if not an integer, prompt for input and go back to line 49, run the scanner again.

```
⋮
53             if (temp > 0) { // Check the input to see it is positive integer value or not
54                 return temp;  // Return the value
55             } else {  // The input is non-positive integer
56                 System.out.print("Please a positive integer value:  "); // Prompt for input
57             }
⋮
```

If temp (user input) is larger than 0, return the value.
If temp (user input) is not larger than 0, prompt for input and go back to line 49, run the scanner again.



Sort

Code

```
public class TriangleChecker_20190875A  {
      ⋮
    public static void main(String args[]) {
      ⋮
        Sort(); // Sort the three inputs, descending order
      ⋮
    }
}
```

Description

Sort is a function to sort the three sides in descending order.

After sorting, the largest value will store in `side1`, the smallest will store in `side3`.

Code

```
public static void Sort() { // Sort the three inputs, descending order
    int temp; // Temporary variable

    // Determine whether side1 or side2 is larger
    if (side1 < side2) {
        temp = side1;
        side1 = side2;
        side2 = temp;
    }
    // Determine whether side1 or side3 is larger
    if (side1 < side3) {
        temp = side1;
        side1 = side3;
        side3 = temp;
    }
    // Determine whether side2 or side3 is larger
    if (side2 < side3) {
        temp = side2;
        side2 = side3;
        side3 = temp;
    }
}
```

Description

Sort is a function to sort the three sides in descending order.
Compare `side1` and `side2`, Swap if `side2` is larger.
Compare `side1` and `side3`, Swap if `side3` is larger.
Compare `side2` and `side3`, Swap if `side3` is larger.

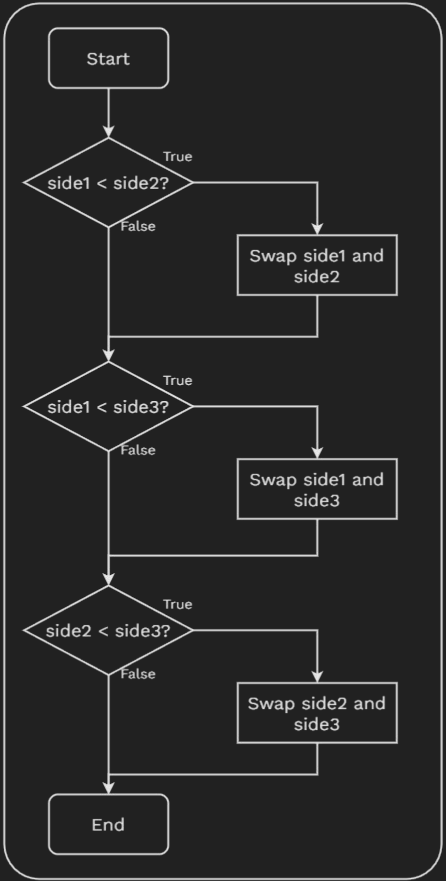
Sort

Code

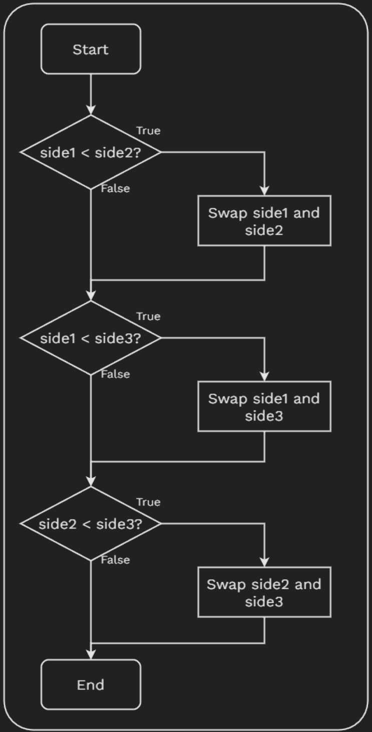
```
public static void Sort() { // Sort the three inputs, descending order
    int temp; // Temporary variable

    // Determine whether side1 or side2 is larger
    if (side1 < side2) {
        temp = side1;
        side1 = side2;
        side2 = temp;
    }
    // Determine whether side1 or side3 is larger
    if (side1 < side3) {
        temp = side1;
        side1 = side3;
        side3 = temp;
    }
    // Determine whether side2 or side3 is larger
    if (side2 < side3) {
        temp = side2;
        side2 = side3;
        side3 = temp;
    }
}
```

Description

`Sort` is a function to sort the three sides in descending order.
Compare `side1` and `side2`, Swap if `side2` is larger.
Compare `side1` and `side3`, Swap if `side3` is larger.
Compare `side2` and `side3`, Swap if `side3` is larger.



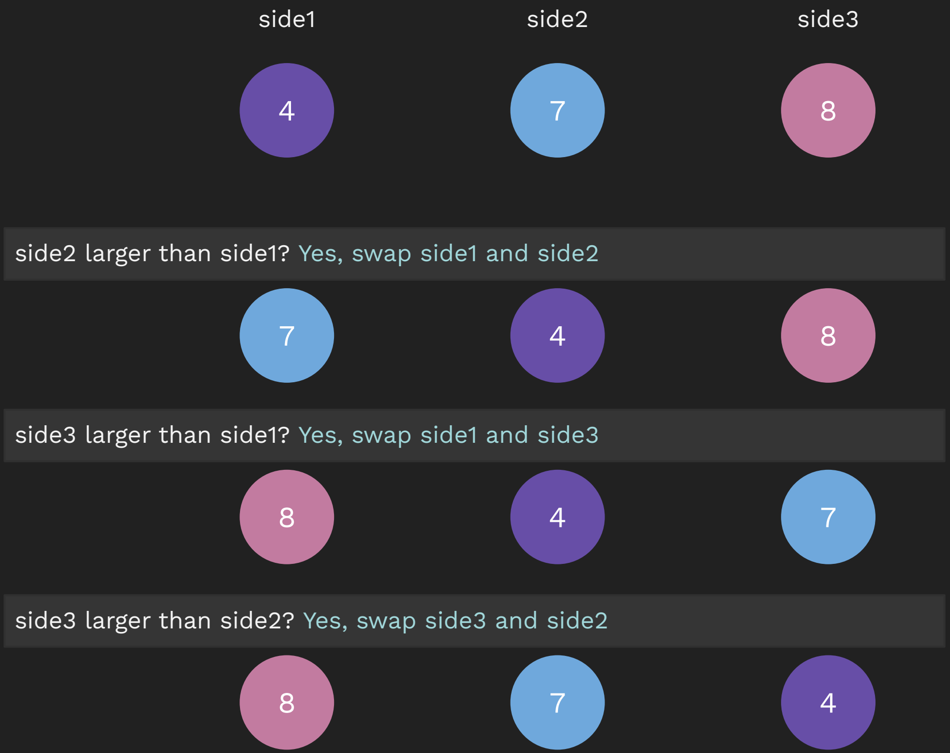
Sort



Description

Compare `side1` and `side2` , Swap if `side2` is larger.
Compare `side1` and `side3` , Swap if `side3` is larger.
Compare `side2` and `side3` , Swap if `side3` is larger.

Demonstrate Sort with 3 integer.



side2 larger than side1? Yes, swap side1 and side2

side3 larger than side1? Yes, swap side1 and side3

side3 larger than side2? Yes, swap side3 and side2

After `Sort`, three sides are in descending order.

Sort (Swap)

Code

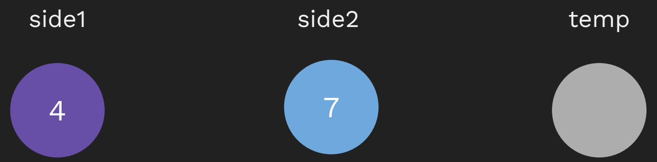
```
110 public static void Sort() { // Sort the three inputs, descending order
111     int temp; // Temporary variable
          ⋮
113     // Determine whether side1 or side2 is larger
114     if (side1 < side2) {
115         temp = side1;
116         side1 = side2;
117         side2 = temp;
118     }
          ⋮
133 }
```

Description

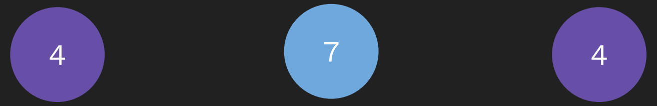
line 115-117 will perform swapping value of `side1` and `side2`.

line 115: Store `side1` in `temp`, a temporary variable
line 116: Store `side2` in `side1`.
line 117: Store `temp` in `side2`.

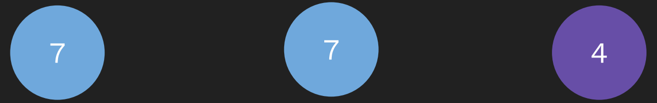
Demonstrate swap with 2 integer.



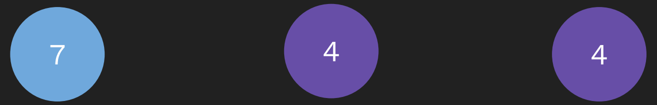
Store side1 in temp , a temporary variable



Store `side2` in `side1`.

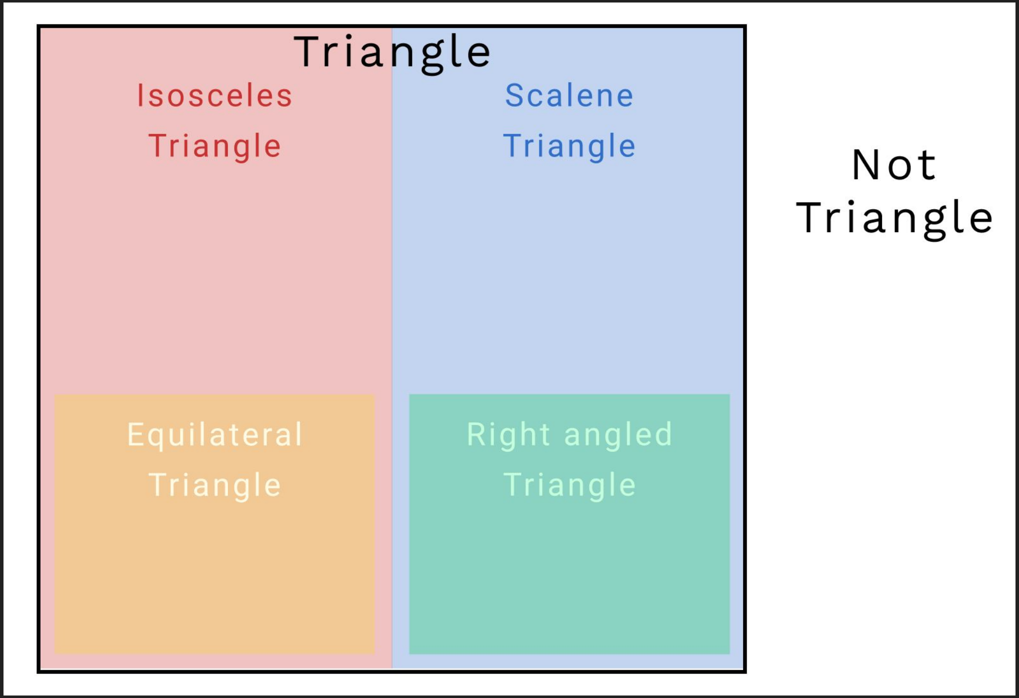


Store `temp` in `side2`.



After swapping, value of `side1` and `side2` are swapped.

Classify



Description

We need to identify our goal first. After we have three sides, we should have one of these five outputs.

The three sides might not form any triangle of one of the four kinds of triangles.

There are two main categories -- "Isosceles triangle" and "Scalene triangle".

"Isosceles triangle" included "Equilateral Triangle".

All isosceles right-angled triangles cannot have sides with integer values. Therefore, only scalene right-angled triangle would be an output. "Scalene triangle" included "Right-angled triangle".

Classify

Code

```java
public class TriangleChecker_20190875A  {
          ⋮
    public static void main(String args[]) {
          ⋮
          Classify(); // Classify the triangle
          ⋮
    }
}
```

Description

`Classify` is a function to classify the three sides to one of the five possible categories.

Code

```java
public static void Classify() { // Classify the triangle
    if (TriangleCheck () == true) { // Check whether can these three sides can form a triangle
        if (EqualateralCheck () == false) { // Check whether the triangle is equilateral
            if ( IsoscelesCheck () == false) { // Check whether the triangle is isosceles
                RightAngledCheck (); // Check whether the triangle is right-angled
            }
        }
    }
}
```

Description

`TriangleCheck`, `EqualateralCheck`, `IsoscelesCheck`, `RightAngledCheck` are the functions to identify the three sides to the possible categories.

Classify (TriangleCheck)

Code

```
public static boolean TriangleCheck () { // Check whether can these three sides can form a triangle
    if (side1 >= (side2 + side3)) { // Determine whether can these three sides can form a triangle
        System.out.println("These three sides could not form any triangle.  "); // These three sides can form a triangle
        return false;
    } else { // These three sides can not form a triangle
        return true;
    }
}
```

Description

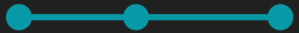
To form a triangle, the longest side should not be larger than the sum of the other two sides.
If the three sides could form a triangle, the program will return true.

There are two more possible outcomes and both of them could not form any triangle.

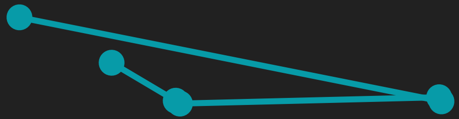
1. If the largest side is equal to the sum of the other two sides, it simply becomes a line.
2. If the largest side is larger than the sum of the other two sides, the three sides could not connect themselves.

For these two cases, the program will print out the result, that these three sides could not form any triangle and return false.

1.



2.



Classify (EquilateralCheck & IsoscelesCheck)

Code

```
public static boolean EquilateralCheck () { // Check whether the triangle is equilateral
    if ((side1 == side2) && (side2 == side3)) { // Determine whether the triangle is equilateral
        System.out.println("These three sides could form a Equilateral Triangle.  ");// The
triangle is equilateral
        return true;
    } else {  // The triangle is not equilateral
        return false;
    }
}
```

Description

A Equilateral Triangle has the same value of all three sides. The program will compare the value of all side, to see are the values the same.

If yes, it will print out the result -- Equilateral Triangle and return true.
If no, it will return false.

Code

```
public static boolean IsoscelesCheck () { // Check whether the triangle is isosceles
    if ((side1 == side2) || (side1 == side2) || (side2 == side3)) {  // Determine whether the
triangle is isosceles
        System.out.println("These three sides could form a Isosceles Triangle  ");// The triangle
is isosceles
        return true;
    } else {  // The triangle is not isosceles
        return false;
    }
}
```

Description

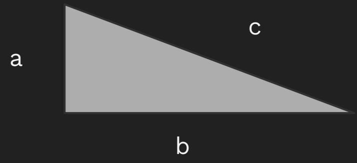
A Isosceles Triangle has the same value in two of the three sides. The program will compare the three sides, see if there are any same values.

If yes, it will print out the result -- Equilateral Triangle and return true.
If no. it will return false.

Classify (RightAngledCheck)

Code

```
public static void RightAngledCheck () { // Check whether the triangle is right-angled
    if (side1  * side1  == (side2  * side2  + side3  * side3)) {  // Determine whether the triangle is right-angled
        System.out.println ("These three sides could form a  Right-angled Triangle and Scalene Triangle  "); // The triangle is Right-angled and Scalene
    } else {  // The triangle is not right-angled
        System.out.println ("These three sides could form a  Scalene Triangle "); // The triangle is Scalene

    }
}
```



Description

To form a right-angled triangle, the three sides need to follow the Pythagorean theorem. (a^2+b^2 = c^2)
The program would check if the square of the longest side (c^2) is equal to the square of themselves of the other two sides (a^2 + b^2).

On top of that, All isosceles right-angled triangles cannot have sides with integer values. Therefore, only scalene right-angled triangle would be output. "Scalene triangle" included "Right-angled triangle".

If the three sides follow the Pythagorean theorem, the program will print the result -- Right-angles Triangle and Scalene Triangle.
If not, there is only one possible output for the triangle, the program will print the result -- Scalene Triangle.

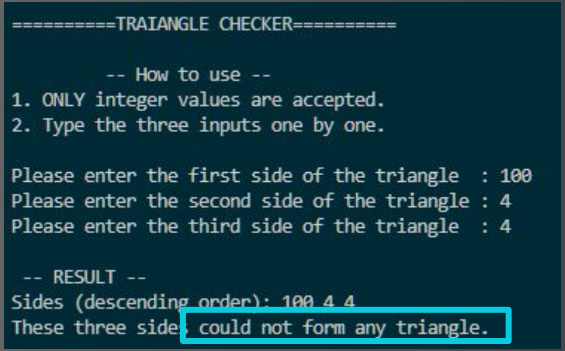
Testing Result

Test #1

	side1	side2	side3
INPUT	100	4	4

Expect Output Not a triangle

Program Output



```
==========TRAIANGLE CHECKER==========

         -- How to use --
1. ONLY integer values are accepted.
2. Type the three inputs one by one.

Please enter the first side of the triangle  : 100
Please enter the second side of the triangle : 4
Please enter the third side of the triangle  : 4

 -- RESULT --
Sides (descending order): 100 4 4
These three sides could not form any triangle.
```

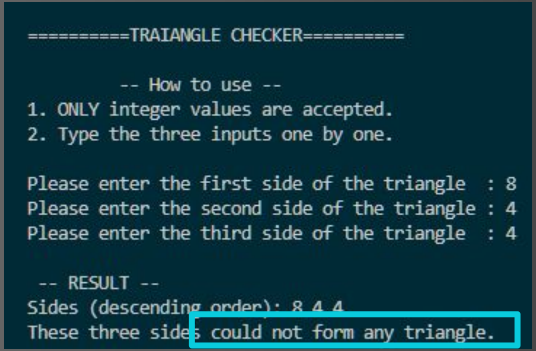
Expect Output = Program Output
Test pass.

Test #2

	side1	side2	side3
INPUT	8	4	4

Expect Output Not a triangle

Program Output

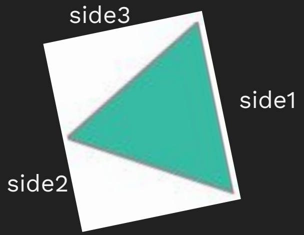


```
==========TRAIANGLE CHECKER==========

        -- How to use --
1. ONLY integer values are accepted.
2. Type the three inputs one by one.

Please enter the first side of the triangle  : 8
Please enter the second side of the triangle : 4
Please enter the third side of the triangle  : 4

 -- RESULT --
Sides (descending order): 8 4 4
These three sides could not form any triangle.
```

Expect Output = Program Output
Test pass.

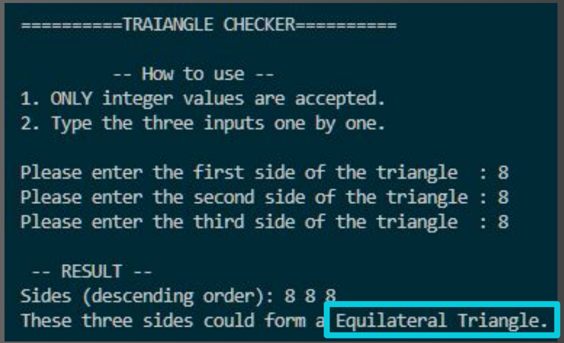
Test #3

	side1	side2	side3
INPUT	8	8	8



Expect Output	Equilateral Triangle

Program Output



```
==========TRAIANGLE CHECKER==========

        -- How to use --
1. ONLY integer values are accepted.
2. Type the three inputs one by one.

Please enter the first side of the triangle  : 8
Please enter the second side of the triangle : 8
Please enter the third side of the triangle  : 8

 -- RESULT --
Sides (descending order): 8 8 8
These three sides could form a Equilateral Triangle.
```

Expect Output = Program Output
Test pass.

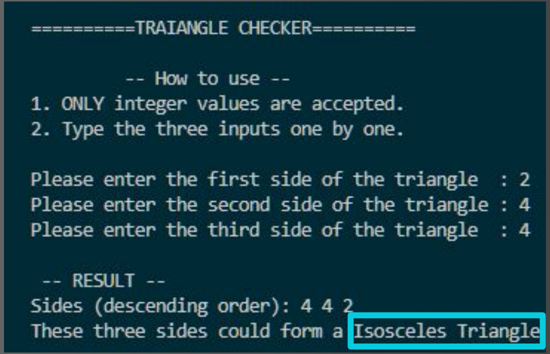
Test #4

	side1	side2	side3
INPUT	2	4	4

side3
side1
side2

Expect Output Isosceles Triangle

Program Output

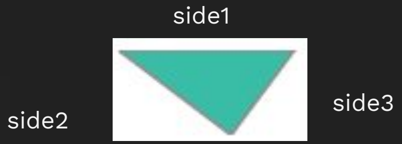


```
==========TRAIANGLE CHECKER==========

         -- How to use --
1. ONLY integer values are accepted.
2. Type the three inputs one by one.

Please enter the first side of the triangle  : 2
Please enter the second side of the triangle : 4
Please enter the third side of the triangle  : 4

 -- RESULT --
Sides (descending order): 4 4 2
These three sides could form a Isosceles Triangle
```

Expect Output = Program Output
Test pass.

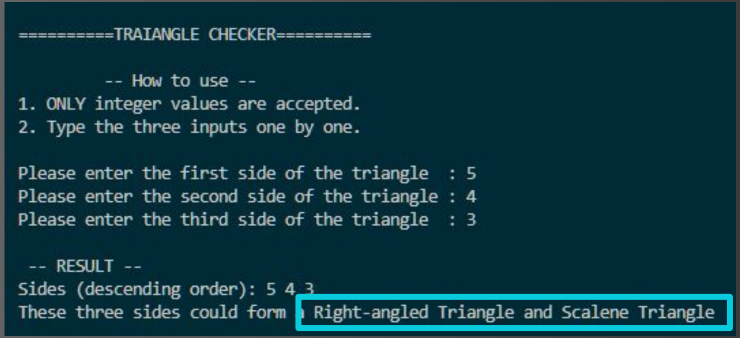
Test #5

	side1	side2	side3
INPUT	5	4	3



Expect Output Right-angled Triangle and Scalene Triangle

Program Output



```
==========TRAIANGLE CHECKER==========

         -- How to use --
1. ONLY integer values are accepted.
2. Type the three inputs one by one.

Please enter the first side of the triangle  : 5
Please enter the second side of the triangle : 4
Please enter the third side of the triangle  : 3

 -- RESULT --
Sides (descending order): 5 4 3
These three sides could form a Right-angled Triangle and Scalene Triangle
```

Expect Output = Program Output
Test pass.

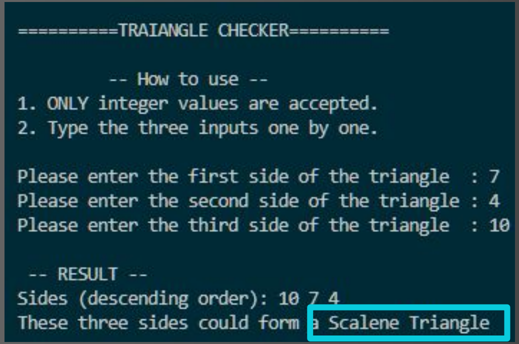
Test #6

side3
side1
side2

	side1	side2	side3
INPUT	7	4	10

Expect Output Scalene triangles

Program Output



```
===========TRAIANGLE CHECKER==========

         -- How to use --
1. ONLY integer values are accepted.
2. Type the three inputs one by one.

Please enter the first side of the triangle  : 7
Please enter the second side of the triangle : 4
Please enter the third side of the triangle  : 10

 -- RESULT --
Sides (descending order): 10 7 4
These three sides could form a Scalene Triangle
```

Expect Output = Program Output
Test pass.

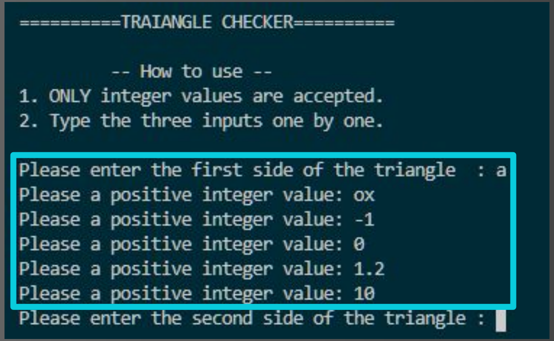
Test #7

side1

INPUT a ox -1 0 1.2 10

Expect Output keep asking for input, until user input positive integer (10)

Program Output



```
==========TRAIANGLE CHECKER==========

         -- How to use --
1. ONLY integer values are accepted.
2. Type the three inputs one by one.

Please enter the first side of the triangle  : a
Please a positive integer value: ox
Please a positive integer value: -1
Please a positive integer value: 0
Please a positive integer value: 1.2
Please a positive integer value: 10
Please enter the second side of the triangle :
```

Expect Output = Program Output
Test pass.

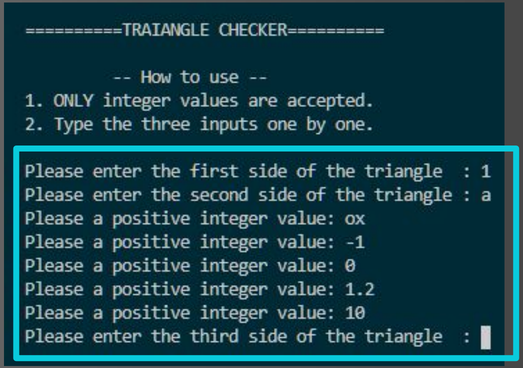
Test #8

side1 side2

INPUT 1 a ox -1 0 1.2 10

Expect Output keep asking side2 input, until user input positive integer (10)

Program Output



```
==========TRAIANGLE CHECKER==========

         -- How to use --
1. ONLY integer values are accepted.
2. Type the three inputs one by one.

Please enter the first side of the triangle  : 1
Please enter the second side of the triangle : a
Please a positive integer value: ox
Please a positive integer value: -1
Please a positive integer value: 0
Please a positive integer value: 1.2
Please a positive integer value: 10
Please enter the third side of the triangle  :
```

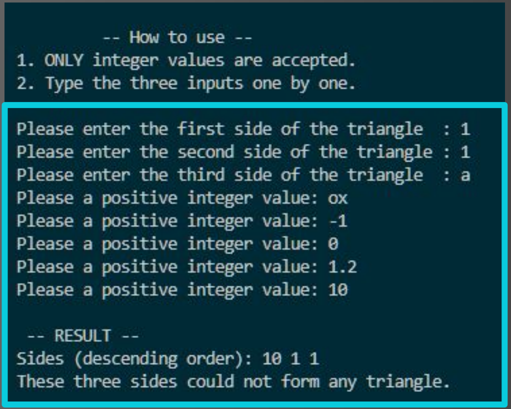
Expect Output = Program Output
Test pass.

Test #9

	side1	side2	side3					
INPUT	1	1	a	ox	-1	0	1.2	10

Expect Output keep asking side3 input, until user input positive integer (10)

Program Output



```
        -- How to use --
1. ONLY integer values are accepted.
2. Type the three inputs one by one.
Please enter the first side of the triangle  : 1
Please enter the second side of the triangle : 1
Please enter the third side of the triangle  : a
Please a positive integer value: ox
Please a positive integer value: -1
Please a positive integer value: 0
Please a positive integer value: 1.2
Please a positive integer value: 10

 -- RESULT --
Sides (descending order): 10 1 1
These three sides could not form any triangle.
```

Expect Output = Program Output
Test pass.

This is the end of the report.