Problem statement.

For this problem, it requires using a stack to perform operations on numbers similarly to how a machine would in it's natural state. Before performing the operations, I must first take an inflix expression and convert it to postfix, by sorting numbers from operators, and then sorting operators based on their precedence. Operators will be sored in a stack and popped from the stack to an output string along with the numbers accordingly, based on their precedence and the order in which they will be applied to the operands. Operators will be given corresponding integer numbers to state their precedence amongst eachother, amd to sort the order in which they are placed in the postfix expression. This problem will have many conditions and loops to help separate the characters from the input inflix string from numbers, operators, and paranthesis, and also to filter the input to ensure it is submitted in the correct form, only digits 0-9 and *, +, -, / , ^ and (,).than to perform operations on the postflix expression, I will use the algorithm given from the assignmnent pdf, by pushing each number to the stack and when the current character of the postfix expression is a operator, pop 2 operands from the stack, perform the required operation on them and then pop the result back to the stack, until there is one number left on the stack, which will give me the answer.

Analysis and design notes.

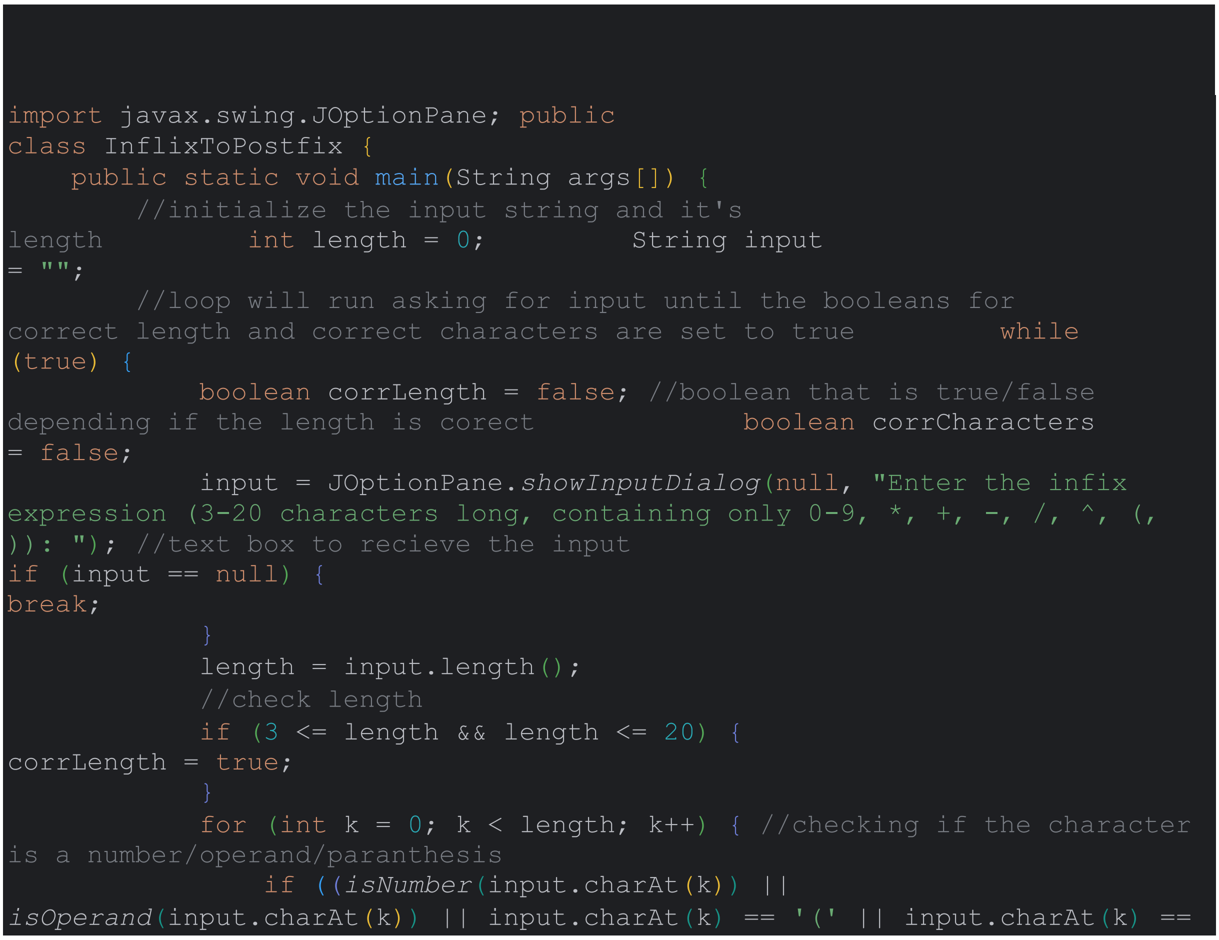
For this assignment I will use 1 class. I will firstly in the main method create a while loop that will request for an input of an inflix expression, and will keep running unless the expression contains the correct characters, 0-9 and ^, *, /, +, -, (,), and if it contains any other numbers except these numbers it will keep requesting a new input, and it will keep running if the inflix expression contains less than 3, or more than 20 characters. After receiving the correct input, I will store the input string into a new array of characters called 'inflix' and pass this array of characters into a method which will convert it to postflix. This method will output an output postfix string. I will initialize a new arraystack object here and the flow of the method will go as followed:

-loop through each character in the array.
1.if it's a number (checked through a Boolean method called 'isNumber', move directly to output string.
2.else, if its an operator (checked through Boolean method called 'isOperator'), call another method called 'operandValue' to get it's precedence.
3.if the stack is empty, or the currant operand has a higher precedence than that at the top of the stack, ill push the operand integer value to the stack.
4.else if the operand at the top of the stack has a higher precedence, while the stack isn't empty, ill pop every operand that only has a higher precedence than the current operator to the output string, ill only stop if the stack is empty, the currant operator is bigger than the one on top of the stack, or until we come across a paranthesis.

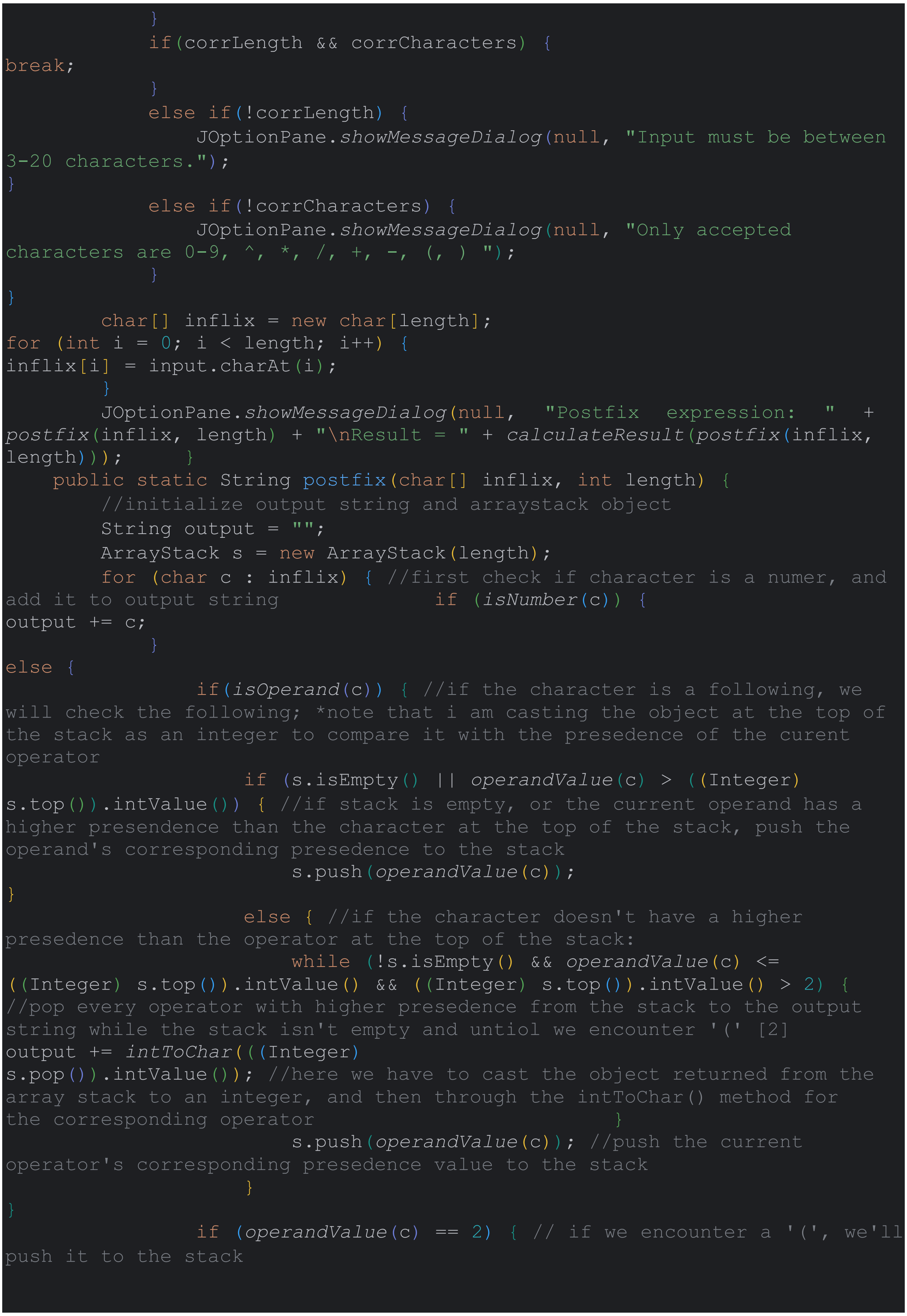
5. when popping the operator to the output string, ill call another method ‘intToChar’, which will convert the integer from the stack back to it’s corresponding character operator.
Ill also have to cast the object from the stack to an integer, and then from the intToChar method to a character.
6.after this while loop stops, ill push the current operator to the stack.
7.if the current operator is ‘(‘, ill push it to the stack.
8.if the current operator is ‘)’, ill pop everything from the stack while the stack isn’t empty and until we encounter another paranthesis to the output string, and then pop the top of the stack to discard the paranthesis we came across, and iterate to the next character to discard the paranthesis’.
8. once every character has been scanned, well pop everything to the output string, and for extra measures, add an if statement that states if theres a paranthesis at the top of the stack, we just pop it and discard of it. 9.method then returns the output string.

Ill then have another method to calculate the result. This method will take in the postfix string as an input. It will then create a new arrayStack object. It will iterate through each character in the array, and check if it’s a number, it will get the numerical value of that character and push it to the stack, and then if it’s an operator, pop the first 2 numbers in the stack, perform the operations, and push the result back to the stack. When the entire string has been checked, the method will then pop the last number left in the stack, and return it as the result. The main method will then print this result and the postfix expression to the screen.

Coding



```
import javax.swing.JOptionPane; public
class InflixToPostfix {
    public static void main(String args[]) {
        //initialize the input string and it's
length          int length = 0;          String input
= "";
        //loop will run asking for input until the booleans for
correct length and correct characters are set to true          while
(true) {
            boolean corrLength = false; //boolean that is true/false
depending if the length is corect                boolean corrCharacters
= false;
            input = JOptionPane.showInputDialog(null, "Enter the infix
expression (3-20 characters long, containing only 0-9, *, +, -, /, ^, (,
)): "); //text box to recieve the input
if (input == null) {
break;
            }
            length = input.length();
            //check length
            if (3 <= length && length <= 20) {
corrLength = true;
            }
            for (int k = 0; k < length; k++) { //checking if the character
is a number/operand/paranthesis
                if ((isNumber(input.charAt(k)) ||
isOperand(input.charAt(k)) || input.charAt(k) == '(' || input.charAt(k) ==
')')) {
                    corrCharacters = true;
                }
```



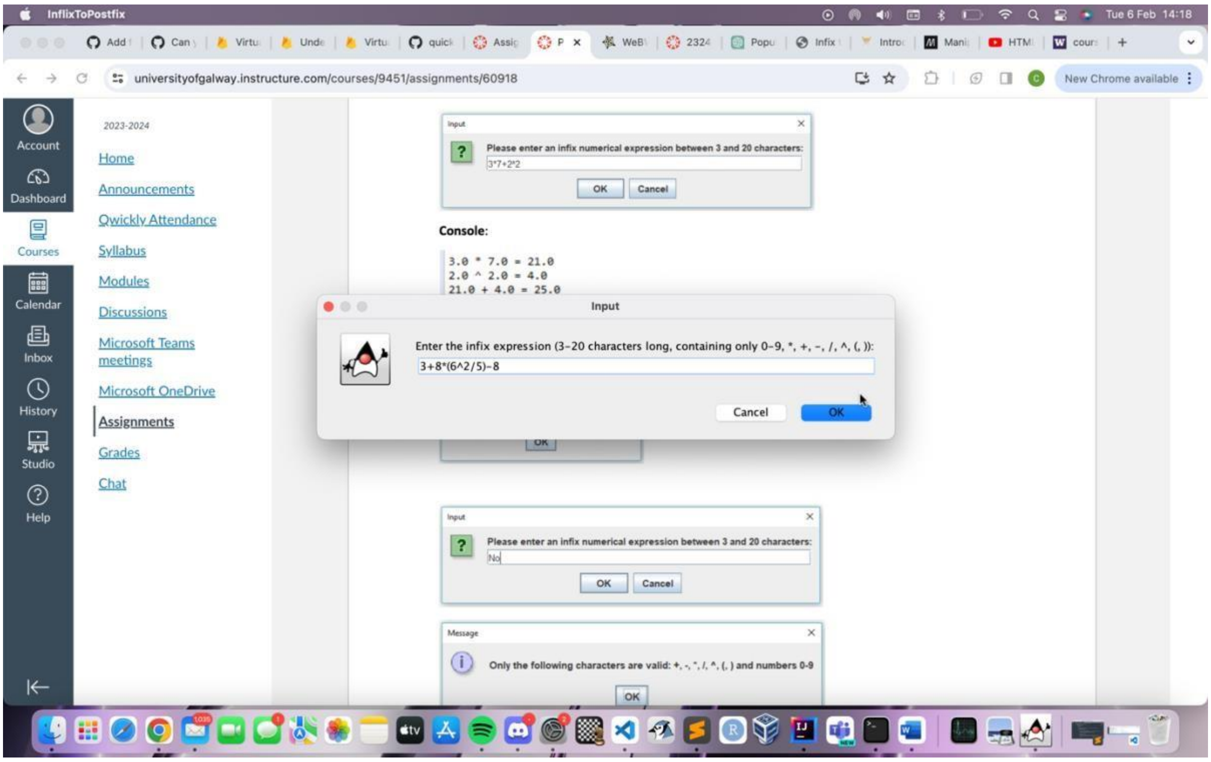
```
            }
            if(corrLength && corrCharacters) {
break;
            }
            else if(!corrLength) {
                JOptionPane.showMessageDialog(null, "Input must be between
3-20 characters.");
}
            else if(!corrCharacters) {
                JOptionPane.showMessageDialog(null, "Only accepted
characters are 0-9, ^, *, /, +, -, (, ) ");
            }
}
        char[] inflix = new char[length];
for (int i = 0; i < length; i++) {
inflix[i] = input.charAt(i);
        }
        JOptionPane.showMessageDialog(null,  "Postfix  expression:  "  +
postfix(inflix, length) + "\nResult = " + calculateResult(postfix(inflix,
length)));      }
    public static String postfix(char[] inflix, int length) {
        //initialize output string and arraystack object
        String output = "";
        ArrayStack s = new ArrayStack(length);
        for (char c : inflix) { //first check if character is a numer, and
add it to output string             if (isNumber(c)) {
output += c;
            }
else {
                if(isOperand(c)) { //if the character is a following, we
will check the following; *note that i am casting the object at the top of
the stack as an integer to compare it with the presedence of the curent
operator
                    if (s.isEmpty() || operandValue(c) > ((Integer)
s.top()).intValue()) { //if stack is empty, or the current operand has a
higher presendence than the character at the top of the stack, push the
operand's corresponding presedence to the stack
                        s.push(operandValue(c));
}
                    else { //if the character doesn't have a higher
presedence than the operator at the top of the stack:
                        while (!s.isEmpty() && operandValue(c) <=
((Integer) s.top()).intValue() && ((Integer) s.top()).intValue() > 2) {
//pop every operator with higher presedence from the stack to the output
string while the stack isn't empty and untiol we encounter '(' [2]
output += intToChar(((Integer)
s.pop()).intValue()); //here we have to cast the object returned from the
array stack to an integer, and then through the intToChar() method for
the corresponding operator                          }
                        s.push(operandValue(c)); //push the current
operator's corresponding presedence value to the stack
                    }
}
                if (operandValue(c) == 2) { // if we encounter a '(', we'll
push it to the stack
```

```
                                s.push(operandValue(c));
}
                          else if (operandValue(c) == 1) { //if we encounter ')',
we'll pop everything from the stack to the output string until we encounter
another paranthesis or the stack is empty
                     while(!s.isEmpty() && ((Integer) s.top()).intValue() >
2) {
                         output += intToChar(((Integer)
s.pop()).intValue());
}
                     s.pop(); //discard the paranthesis
                 }
             }
         }
         //pop everything left in the stack to the output string, but if we
encounter '(' or ')' , we discard them          while (!s.isEmpty()) {
             if (((Integer) s.top()).intValue() <= 2) {
                 s.pop();
}              else {
                 output += intToChar(((Integer) s.pop()).intValue());
             }
}
         return output; //return the output
     }
     public static boolean isNumber (char c) {
         if (c == '0' || c == '1' || c == '2' || c == '3' || c == '4' || c
== '5' || c == '6' || c == '7' || c == '8' || c == '9') {
return true;
         }
else {
             return false;
         }
}
     public static boolean isOperand (char c) {
         if (c == '^' || c == '*' || c == '/' || c == '+' || c == '-') {
return true;
         }
else {
             return false;
         }
     }
     //method to get the presedence of each Symbol
public static int operandValue(char c) {
if (c == '^') {              return 7;
         }
         else if (c == '*') {
return 6;
         }
         else if (c == '/') {
return 5;
         }
         else if (c == '+') {
return 4;
         }
         else if (c == '-') {
return 3;
         }
         else if (c == '(') {
            return 2;
        }          else if (c
== ')') {
```

```
            return 1;
        }
        return 0;
    }
    //method to retrieve the corresponding character from each symbol's
presedence value
    public static char intToChar(int value) {
if (value == 7) {            return '^';
        }
        else if (value == 6) {
return '*';
        }
        else if (value == 5) {
return '/';
        }
        else if (value == 4) {
return '+';
        }
        else if (value == 3) {
return '-';
        }
        else if (value == 2) {
return '(';
        }
        else if (value == 1) {
return ')';
        }
        return ' ';
    }
    private static float calculateResult(String output) {
float number1, number2, result= 0; //initialize variables
        int aLength = output.length(); //length of output
ArrayStack stack = new ArrayStack(aLength);
        for (int i = 0; i < aLength; i++) { //loop to check every character
in the string
            if (isNumber(output.charAt(i))) { //if the current character is
a number:
                float numValue =
Character.getNumericValue(output.charAt(i)); //get the numerical value of
the character and push it to the stack
stack.push(numValue);
            }
            else if(isOperand(output.charAt(i))) { //if the current
character is an operand, pop the two numbers at the top of the stack
number2 = (float)stack.pop(); //here i'm casting the objects as
floating point numbers to avoid compution errors
number1 = (float)stack.pop();
                //performing various operations on the two numbers
depending on what the current operand is, and pushing the result to the
stack
                if(output.charAt(i) == '^') {
                    stack.push((float)Math.pow(number1, number2));;
                }
                else if(output.charAt(i) == '*') {
stack.push((float)number1*number2);
                }
                else if(output.charAt(i) == '/') {
```

```
                         stack.push((float)number1/number2);
                    }
                    else if(output.charAt(i) == '+') {
stack.push((float)number1+number2);
                    }
                    else if(output.charAt(i) == '-') {
stack.push((float)number1-number2);
                    }
               }
}
          result = (float)stack.pop(); //when every character in the ouput
string has been checked, pop the result from the stack and return it
return result;
     }
}
```

testing:



Input

Please enter an infix numerical expression between 3 and 20 characters:

3*7+2*2

OK Cancel

Console:

```
3.0 * 7.0 = 21.0
2.0 ^ 2.0 = 4.0
21.0 + 4.0 = 25.0
```

Input

Enter the infix expression (3-20 characters long, containing only 0-9, *, +, -, /, ^, (,)):

3+8*(6^2/5)-8

Cancel OK

Input

Please enter an infix numerical expression between 3 and 20 characters:

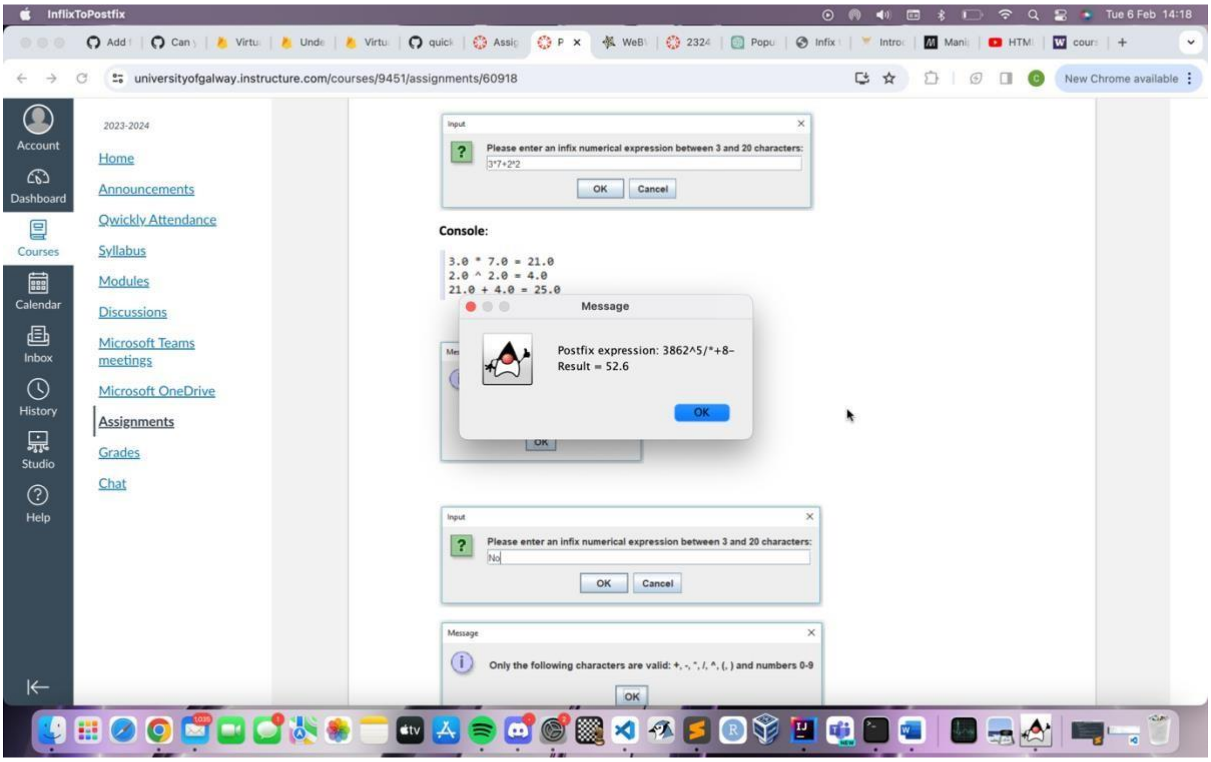
No

OK Cancel

Message

Only the following characters are valid: +, -, *, /, ^, (,) and numbers 0-9

OK



Input

Please enter an infix numerical expression between 3 and 20 characters:

3*7+2^2

OK Cancel

Console:

```
3.0 * 7.0 = 21.0
2.0 ^ 2.0 = 4.0
21.0 + 4.0 = 25.0
```

Message

Postfix expression: 3862^5/*+8-
Result = 52.6

OK

Input

Please enter an infix numerical expression between 3 and 20 characters:

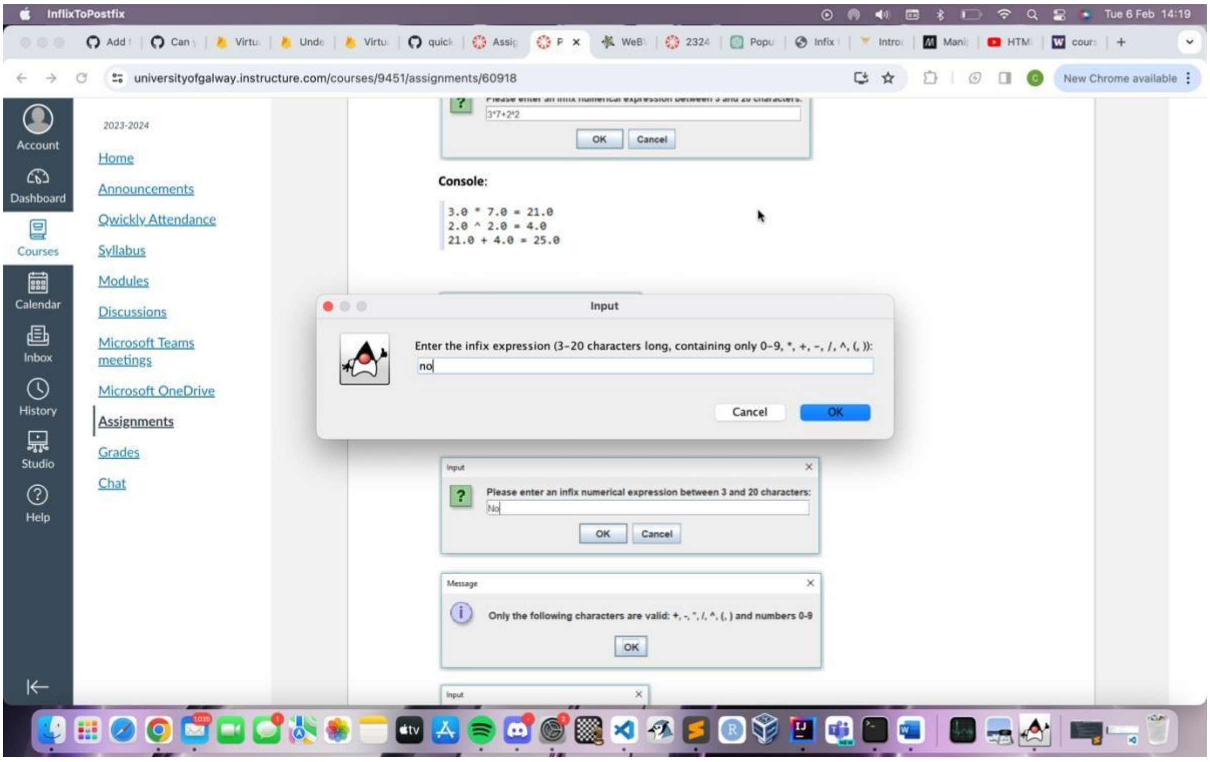
No

OK Cancel

Message

Only the following characters are valid: +, -, *, /, ^, (,) and numbers 0-9

OK



3*7+2^2

OK Cancel

Console:

```
3.0 * 7.0 = 21.0
2.0 ^ 2.0 = 4.0
21.0 + 4.0 = 25.0
```

Input

Enter the infix expression (3-20 characters long, containing only 0-9, *, +, -, /, ^, (,)):

no

Cancel OK

Input

Please enter an infix numerical expression between 3 and 20 characters:

No

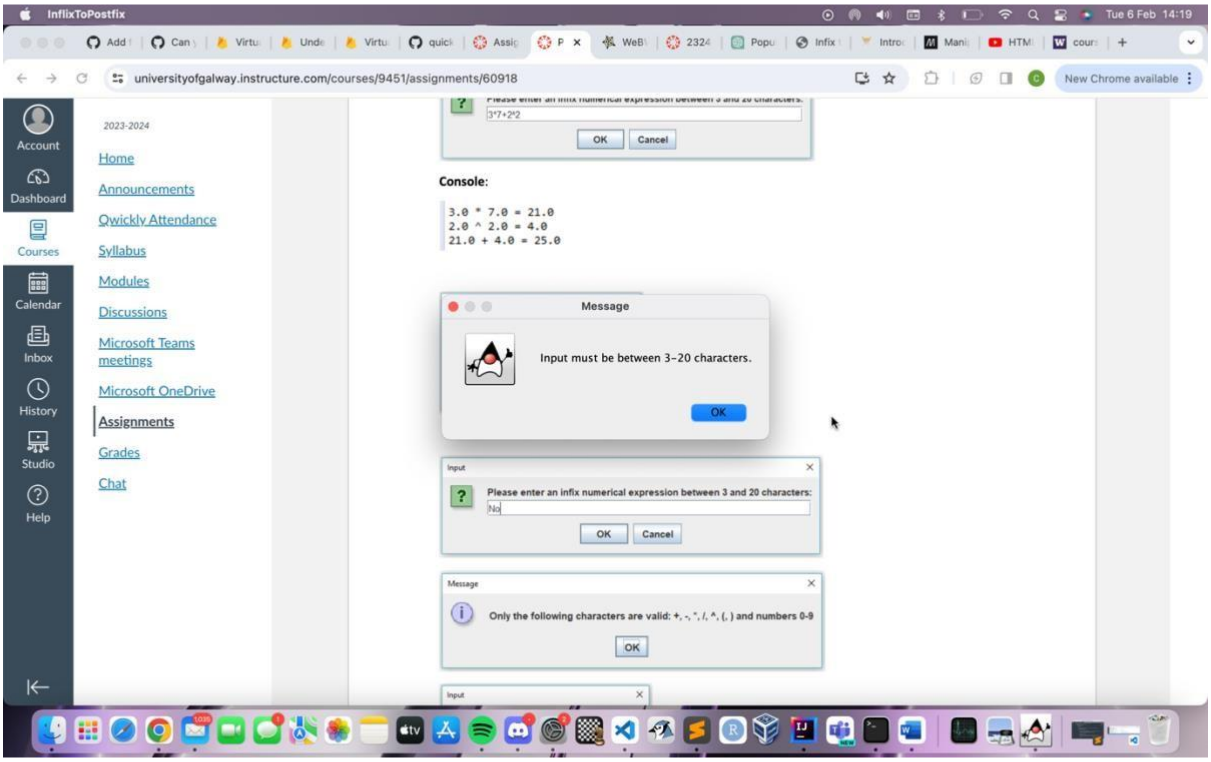
OK Cancel

Message

Only the following characters are valid: +, -, *, /, ^, (,) and numbers 0-9

OK

Input



Please enter an infix numerical expression between 3 and 20 characters:

3*7+2*2

OK Cancel

Console:

```
3.0 * 7.0 = 21.0
2.0 ^ 2.0 = 4.0
21.0 + 4.0 = 25.0
```

Message

Input must be between 3-20 characters.

OK

Input

Please enter an infix numerical expression between 3 and 20 characters:

No

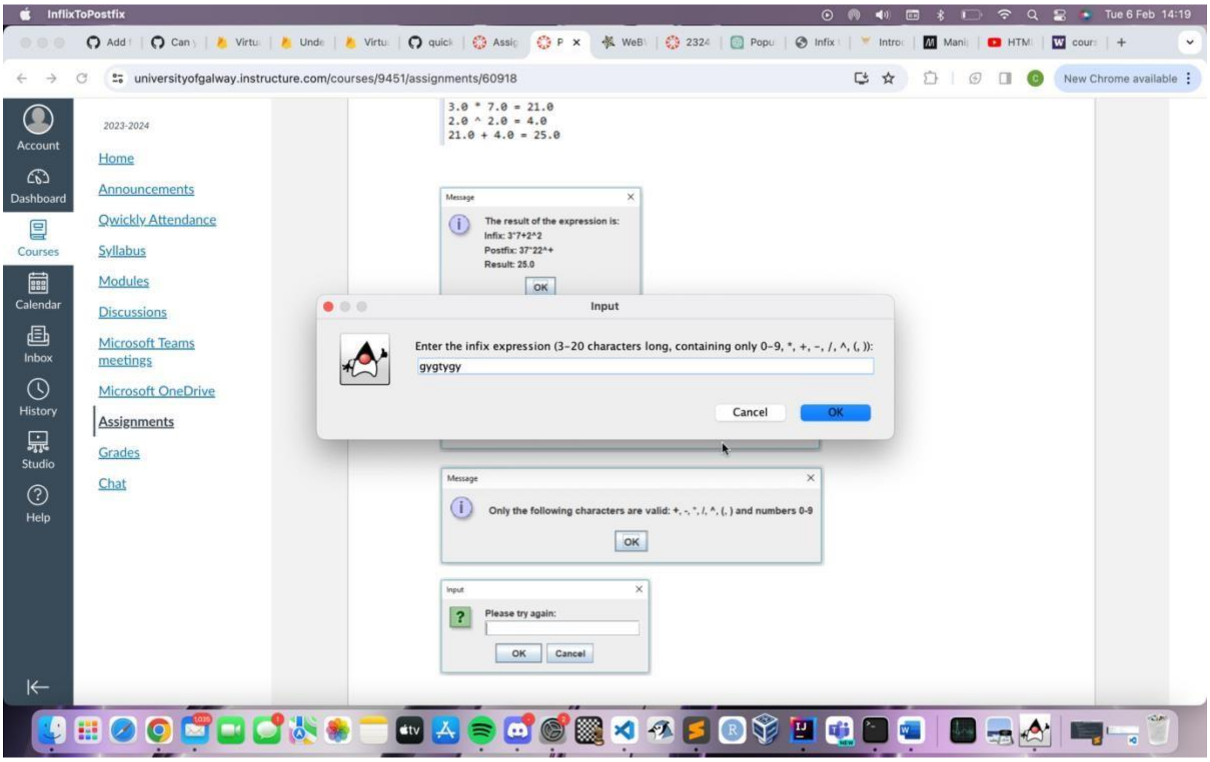
OK Cancel

Message

Only the following characters are valid: +, -, *, /, ^, (,) and numbers 0-9

OK

Input



```
3.0 * 7.0 = 21.0
2.0 ^ 2.0 = 4.0
21.0 + 4.0 = 25.0
```

Message

The result of the expression is:
Infix: 3*7+2^2
Postfix: 37*22^+
Result: 25.0

OK

Input

Enter the infix expression (3-20 characters long, containing only 0-9, *, +, -, /, ^, (,)):

gygtygy

Cancel OK

Message

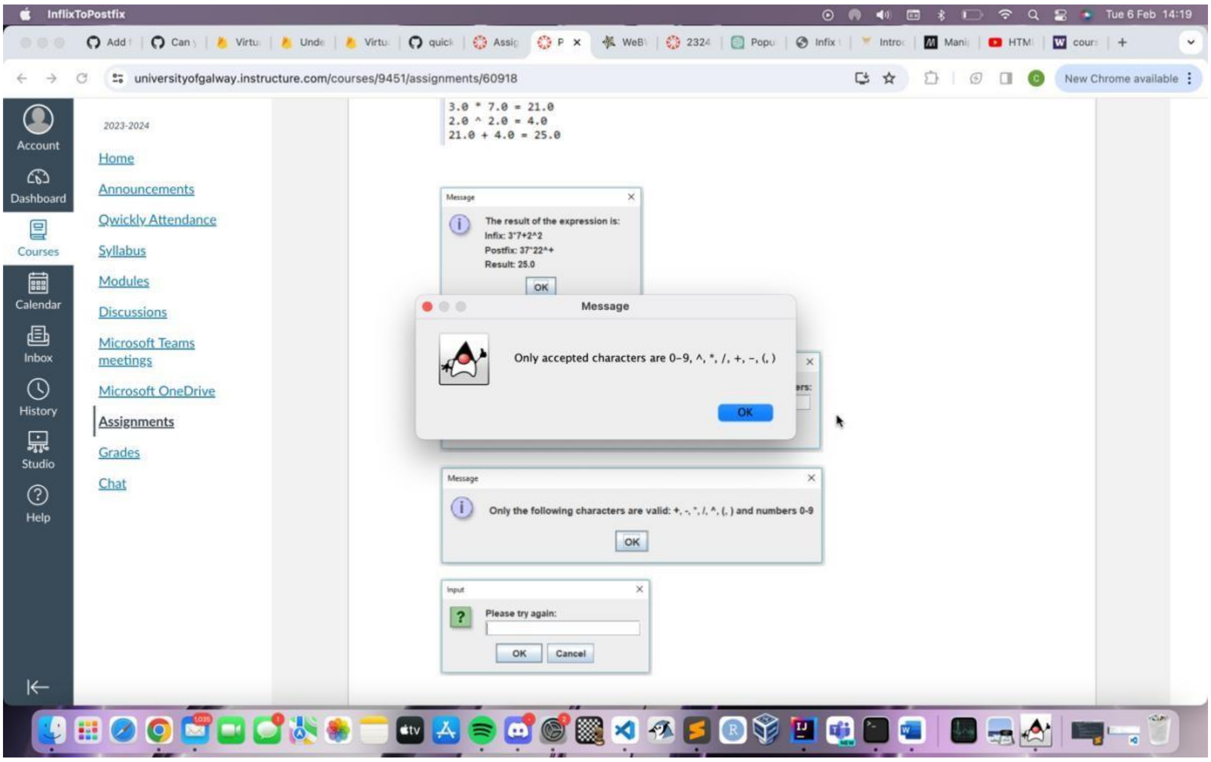
Only the following characters are valid: +, -, *, /, ^, (,) and numbers 0-9

OK

Input

Please try again:

OK Cancel



```
3.0 * 7.0 = 21.0
2.0 ^ 2.0 = 4.0
21.0 + 4.0 = 25.0
```

Message

The result of the expression is:
Infix: 3*7+2^2
Postfix: 37*22^+
Result: 25.0

OK

Message

Only accepted characters are 0-9, ^, *, /, +, -, (,)

OK

Message

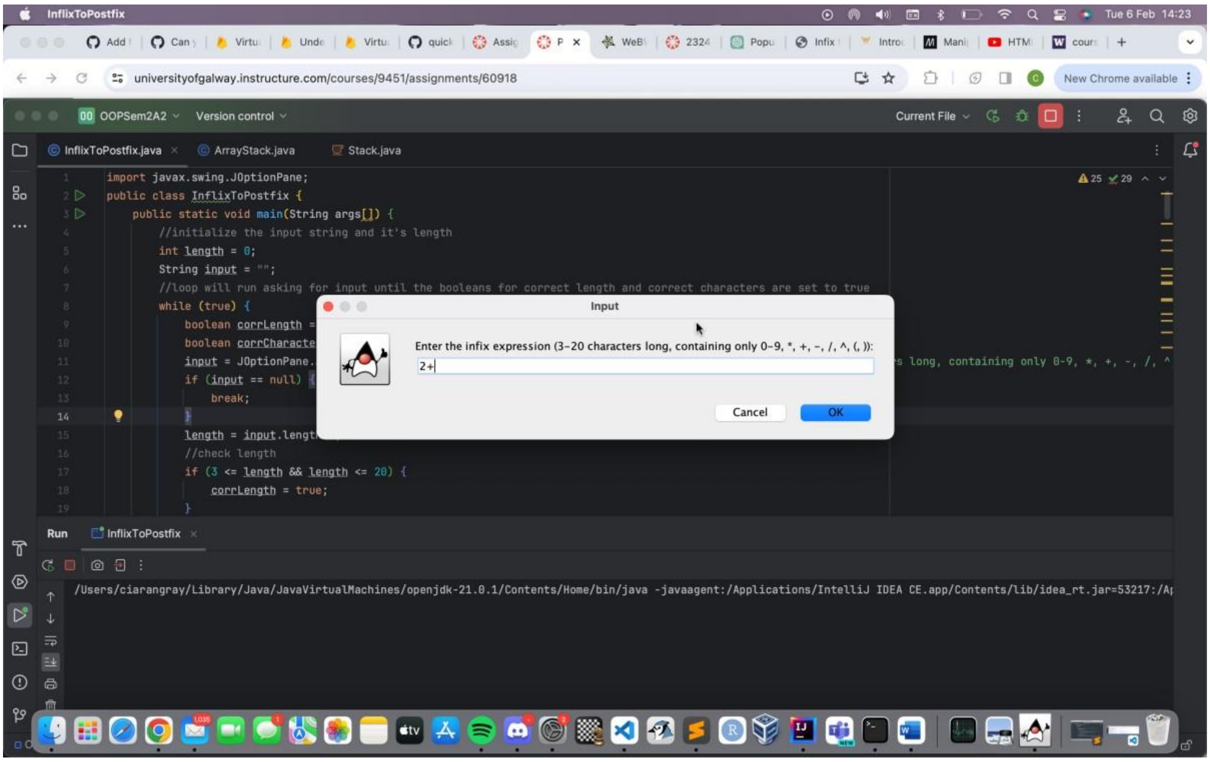
Only the following characters are valid: +, -, *, /, ^, (,) and numbers 0-9

OK

Input

Please try again:

OK Cancel



```
import javax.swing.JOptionPane;
public class InflixToPostfix {
    public static void main(String args[]) {
        //initialize the input string and it's length
        int length = 0;
        String input = "";
        //loop will run asking for input until the booleans for correct length and correct characters are set to true
        while (true) {
            boolean corrLength =
            boolean corrCharacte
            input = JOptionPane.                                                    s long, containing only 0-9, *, +, -, /, ^
            if (input == null) {
                break;
            }
            length = input.lengt
            //check length
            if (3 <= length && length <= 20) {
                corrLength = true;
            }
```

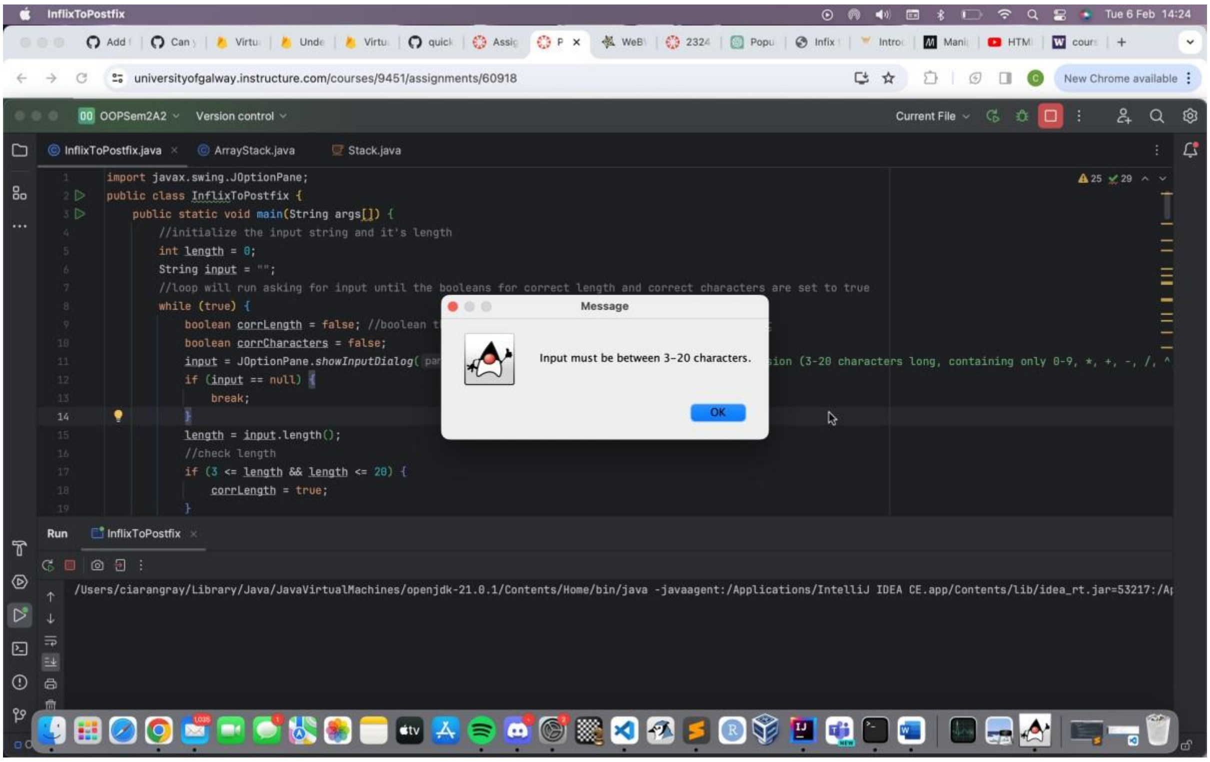
Input

Enter the infix expression (3-20 characters long, containing only 0-9, *, +, -, /, ^, (,)):

2+

Cancel OK

```
/Users/ciarangray/Library/Java/JavaVirtualMachines/openjdk-21.0.1/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA CE.app/Contents/lib/idea_rt.jar=53217:/Ap
```



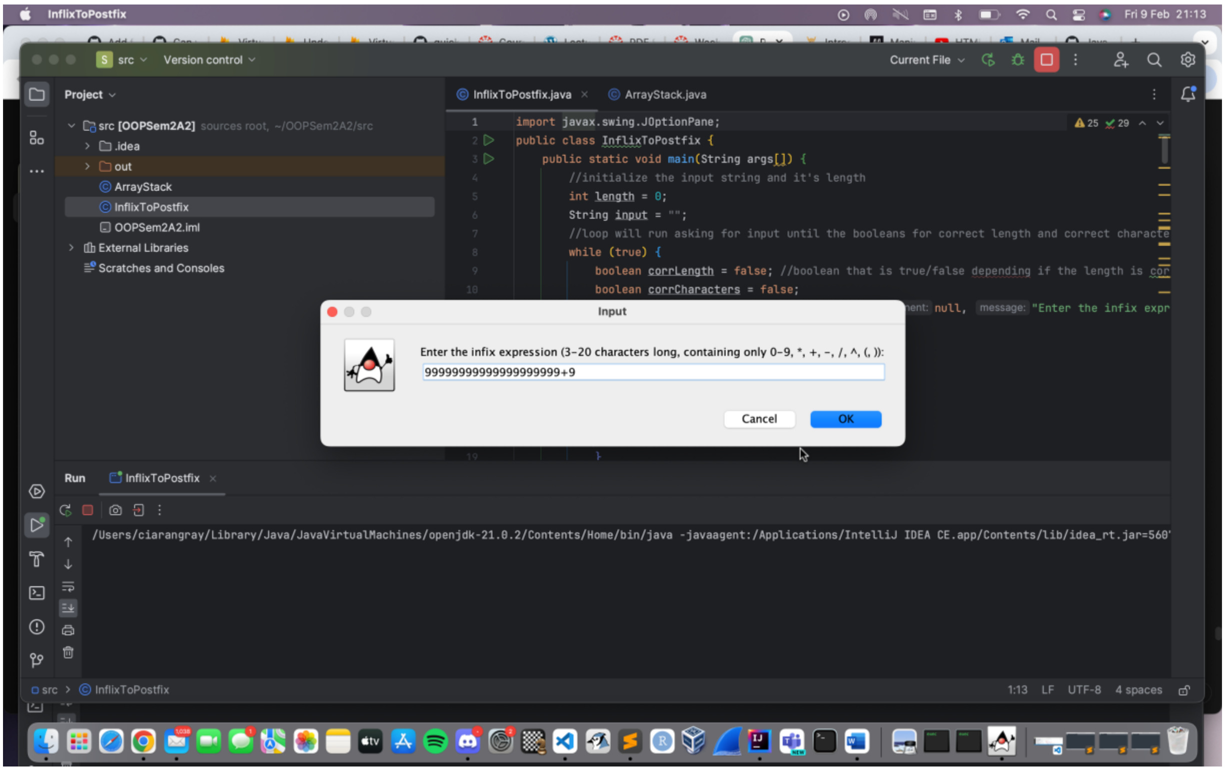
```
import javax.swing.JOptionPane;
public class InflixToPostfix {
    public static void main(String args[]) {
        //initialize the input string and it's length
        int length = 0;
        String input = "";
        //loop will run asking for input until the booleans for correct length and correct characters are set to true
        while (true) {
            boolean corrLength = false; //boolean t
            boolean corrCharacters = false;
            input = JOptionPane.showInputDialog( par                     sion (3-20 characters long, containing only 0-9, *, +, -, /, ^
            if (input == null) {
                break;
            }
            length = input.length();
            //check length
            if (3 <= length && length <= 20) {
                corrLength = true;
            }
```

Message

Input must be between 3-20 characters.

OK

```
/Users/ciarangray/Library/Java/JavaVirtualMachines/openjdk-21.0.1/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA CE.app/Contents/lib/idea_rt.jar=53217:/A
```



```
import javax.swing.JOptionPane;
public class InflixToPostfix {
    public static void main(String args[]) {
        //initialize the input string and it's length
        int length = 0;
        String input = "";
        //loop will run asking for input until the booleans for correct length and correct characte
        while (true) {
            boolean corrLength = false; //boolean that is true/false depending if the length is cor
            boolean corrCharacters = false;
                                                  ent: null, message: "Enter the infix expr
            }
```

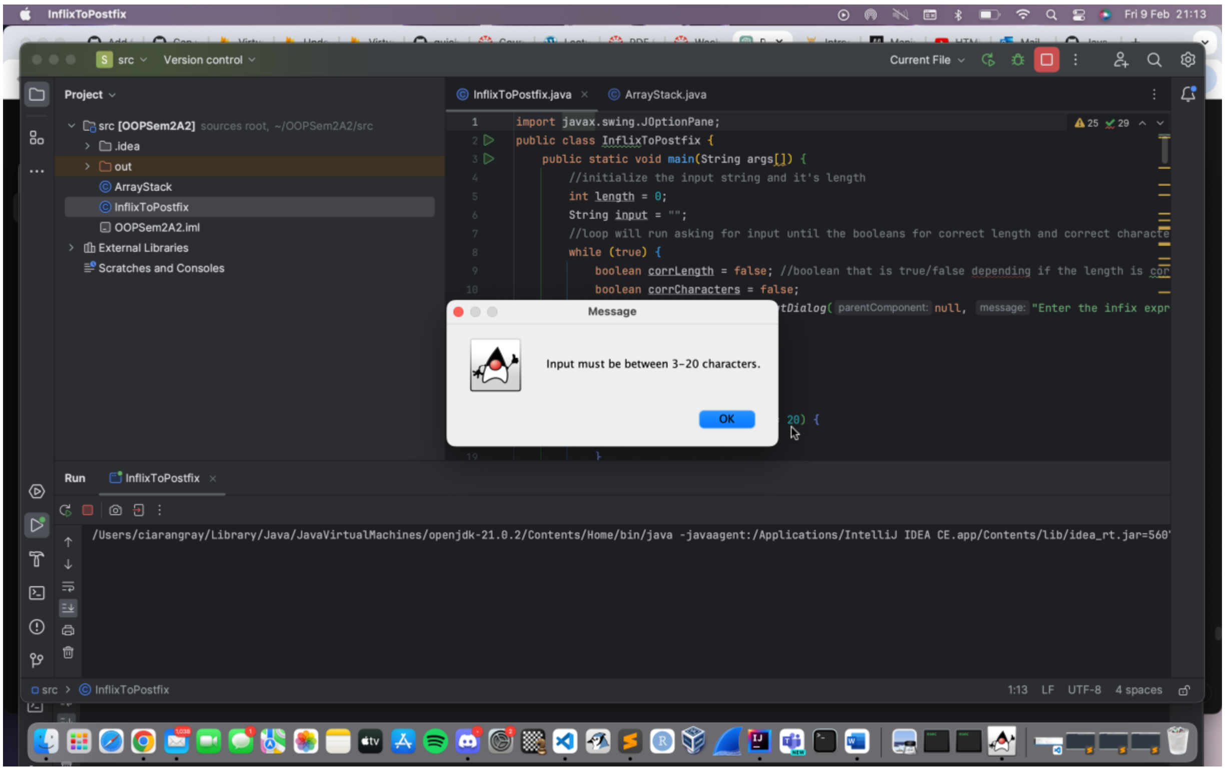
Input

Enter the infix expression (3-20 characters long, containing only 0-9, *, +, -, /, ^, (,)):

99999999999999999999+9

Cancel OK

```
/Users/ciarangray/Library/Java/JavaVirtualMachines/openjdk-21.0.2/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA CE.app/Contents/lib/idea_rt.jar=560
```



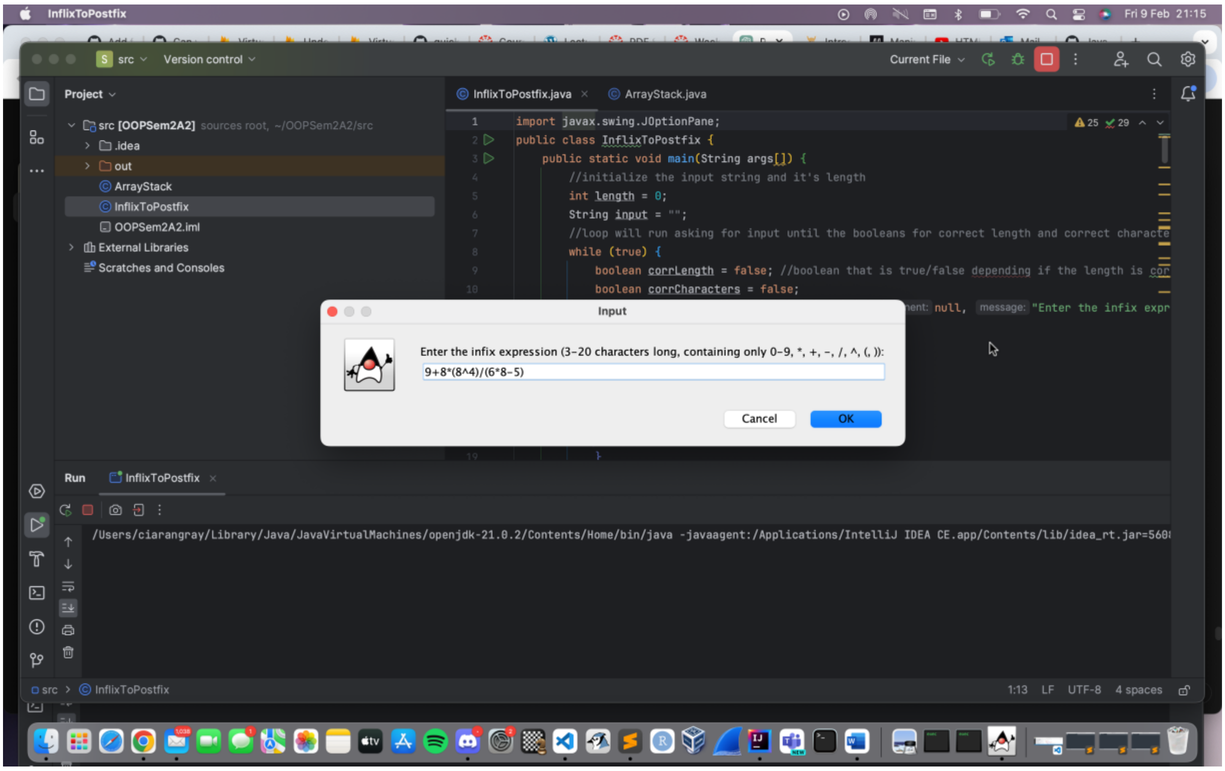
```java
import javax.swing.JOptionPane;
public class InflixToPostfix {
    public static void main(String args[]) {
        //initialize the input string and it's length
        int length = 0;
        String input = "";
        //loop will run asking for input until the booleans for correct length and correct characte
        while (true) {
            boolean corrLength = false; //boolean that is true/false depending if the length is cor
            boolean corrCharacters = false;
            tDialog( parentComponent: null, message: "Enter the infix expr
            20) {
            }
```

Message

Input must be between 3-20 characters.

OK

```
/Users/ciarangray/Library/Java/JavaVirtualMachines/openjdk-21.0.2/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA CE.app/Contents/lib/idea_rt.jar=560
```



```java
import javax.swing.JOptionPane;
public class InflixToPostfix {
    public static void main(String args[]) {
        //initialize the input string and it's length
        int length = 0;
        String input = "";
        //loop will run asking for input until the booleans for correct length and correct characte
        while (true) {
            boolean corrLength = false; //boolean that is true/false depending if the length is cor
            boolean corrCharacters = false;
            nent: null, message: "Enter the infix expr
            }
```

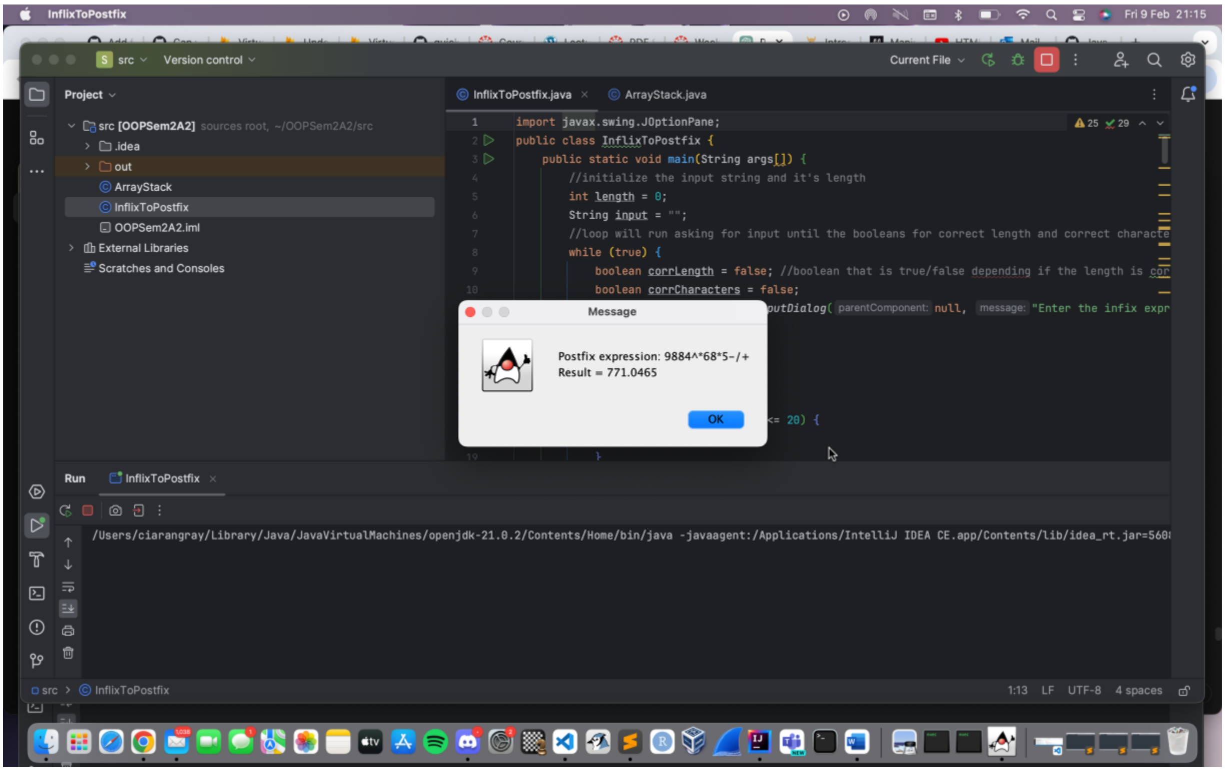
Input

Enter the infix expression (3-20 characters long, containing only 0-9, *, +, -, /, ^, (,)):

9+8*(8^4)/(6*8-5)

Cancel OK

```
/Users/ciarangray/Library/Java/JavaVirtualMachines/openjdk-21.0.2/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA CE.app/Contents/lib/idea_rt.jar=560
```



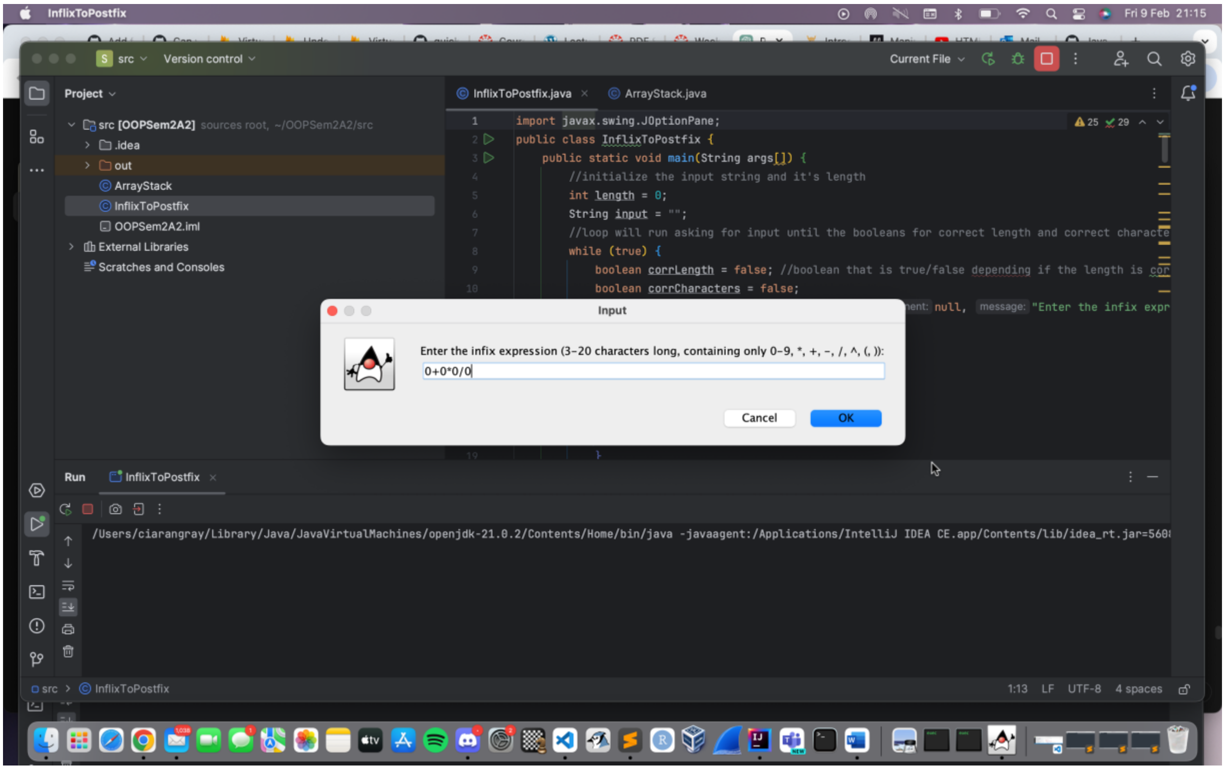
```java
import javax.swing.JOptionPane;
public class InflixToPostfix {
    public static void main(String args[]) {
        //initialize the input string and it's length
        int length = 0;
        String input = "";
        //loop will run asking for input until the booleans for correct length and correct characte
        while (true) {
            boolean corrLength = false; //boolean that is true/false depending if the length is cor
            boolean corrCharacters = false;
                                  putDialog( parentComponent: null,  message: "Enter the infix expr
```

Message

Postfix expression: 9884^*68*5-/+
Result = 771.0465

OK

```
/Users/ciarangray/Library/Java/JavaVirtualMachines/openjdk-21.0.2/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA CE.app/Contents/lib/idea_rt.jar=560
```



```java
import javax.swing.JOptionPane;
public class InflixToPostfix {
    public static void main(String args[]) {
        //initialize the input string and it's length
        int length = 0;
        String input = "";
        //loop will run asking for input until the booleans for correct length and correct characte
        while (true) {
            boolean corrLength = false; //boolean that is true/false depending if the length is cor
            boolean corrCharacters = false;
                                  ent: null,  message: "Enter the infix expr
```

Input

Enter the infix expression (3-20 characters long, containing only 0-9, *, +, -, /, ^, (,)):

0+0*0/0

Cancel OK

```
/Users/ciarangray/Library/Java/JavaVirtualMachines/openjdk-21.0.2/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA CE.app/Contents/lib/idea_rt.jar=560
```