package zad4;  
  
import zad3.QueueS;  
  
public class Main {  
 public static void main(String[] args){  
 String equation = EquationReader.*readFromFile*("eq.txt");  
 System.*out*.println("Infix:**\n**" + equation);  
  
 QueueS<String> s = InfixOperations.*covertToPostfix*(equation);  
 System.*out*.println("**\n**Postfix:");  
 InfixOperations.*printPostfix*(s);  
 }  
}

package zad4;  
  
import java.io.BufferedReader;  
import java.io.FileReader;  
import java.io.IOException;  
  
public class EquationReader {  
 public static String readFromFile(String path){  
 try(BufferedReader reader = new BufferedReader(new FileReader(path))){  
 return reader.readLine();  
 } catch(IOException exception){  
 System.*out*.println("Failed to read from file: " + path);  
 }  
  
 return null;  
 }  
}

package zad4;  
  
import zad1.Queue;  
import zad2.Stack;  
import zad3.QueueS;  
  
public class InfixOperations {  
 public static QueueS<String> covertToPostfix(String equation){  
 String[] symbols = equation.split("**\\**s++");  
  
 QueueS<String> q = new QueueS<>();  
 Stack<String> characters = new Stack<>();  
  
 for(int i = 0; i < symbols.length; i++){  
 if(*isNumber*(symbols[i]))  
 q.enqueue(symbols[i]);  
 else if(symbols[i].equals("("))  
 characters.push("(");  
 else if(symbols[i].equals(")")){  
 String s = characters.pop();  
  
 while(!(s.equals("("))){  
 q.enqueue(s);  
  
 s = characters.pop();  
 }  
 }  
 else{  
 if(characters.isEmpty()) {  
 characters.push(symbols[i]);  
 } else if(*hasHigherPrio*(characters.top(), symbols[i])){  
 characters.push(symbols[i]);  
 } else {  
 if(!characters.top().equals("(")){  
 q.enqueue(characters.pop());  
 }  
 characters.push(symbols[i]);  
 }  
  
 }  
 }  
  
 while(!characters.isEmpty()){  
 String s = characters.pop();  
 q.enqueue(s);  
 }  
  
 return q;  
 }  
  
 public static void printPostfix(QueueS<String> equation){  
 try{  
 QueueS<String> copy = equation.clone();  
  
 while(!copy.isEmpty()){  
 System.*out*.printf("%s ", copy.dequeue());  
 }  
 } catch(CloneNotSupportedException e){  
 System.*out*.println("Can't clone equation");  
 return;  
 }  
  
  
 }  
  
 private static boolean isNumber(String c){  
 if(c.equals("/") || c.equals("\*") || c.equals("+") ||c.equals("-") || c.equals("(") || c.equals(")"))  
 return false;  
  
 return true;  
 }  
  
 private static boolean hasHigherPrio(String last, String current){  
 if(current.equals("\*") || current.equals("/")){  
 if(last.equals("+") || last.equals("-"))  
 return true;  
 }  
  
 return false;  
 }  
}

Wyjście programu dla przykładu:( 4 + 8 ) \* ( 6 - 5 ) / ( ( 3 - 2 ) \* ( 2 + 2 ) )

Infix:

( 4 + 8 ) \* ( 6 - 5 ) / ( ( 3 - 2 ) \* ( 2 + 2 ) )

Postfix:

4 8 + 6 5 - \* 3 2 - 2 2 + \* /