**Министерство цифрового развития, связи и массовых коммуникаций**

**Российской Федерации Ордена Трудового Красного Знамени**

**Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский технический университет связи и информатики»**

Кафедра математической кибернетики и информационных технологий

Отчет по лабораторной работе №4

на тему «Реализация стека/дека»

по дисциплине «Структуры и алгоритмы обработки данных»

Выполнила: студентка группы БВТ1904

Кремер К.С.

# 

Москва 2021

**Цель работы**

Реализовать следующие структуры данных:

* Стек (stack): операции для стека: инициализация, проверка на пустоту, добавление нового элемента в начало, извлечение элемента из начала;
* Дек (двусторонняя очередь, deque): операции для дека: инициализация, проверка на пустоту, добавление нового элемента в начало, добавление нового элемента в конец, извлечение элемента из начала, извлечение элемента из конца.

Задания:

1. Отсортировать строки файла, содержащие названия книг, в алфавитном порядке с использованием двух деков.
2. Дек содержит последовательность символов для шифровки сообщений. Дан текстовый файл, содержащий зашифрованное сообщение. Пользуясь деком, расшифровать текст. Известно, что при шифровке каждый символ сообщения заменялся следующим за ним в деке по часовой стрелке через один.
3. Даны три стержня и n дисков различного размера. Диски можно надевать на стержни, образуя из них башни. Перенести n дисков со стержня А на стержень С, сохранив их первоначальный порядок. При переносе дисков необходимо соблюдать следующие правила:

* на каждом шаге со стержня на стержень переносить только один диск;
* диск нельзя помещать на диск меньшего размера;
* для промежуточного хранения можно использовать стержень В.

Реализовать алгоритм, используя три стека вместо стержней А, В, С. Информация о дисках хранится в исходном файле.

1. Дан текстовый файл с программой на алгоритмическом языке. За один просмотр файла проверить баланс круглых скобок в тексте, используя стек.
2. Дан текстовый файл с программой на алгоритмическом языке. За один просмотр файла проверить баланс квадратных скобок в тексте, используя дек.
3. Дан файл из символов. Используя стек, за один просмотр файла напечатать сначала все цифры, затем все буквы, и, наконец, все остальные символы, сохраняя исходный порядок в каждой группе символов.
4. Дан файл из целых чисел. Используя дек, за один просмотр файла напечатать сначала все отрицательные числа, затем все положительные числа, сохраняя исходный порядок в каждой группе.
5. Дан текстовый файл. Используя стек, сформировать новый текстовый файл, содержащий строки исходного файла, записанные в обратном порядке: первая строка становится последней, вторая – предпоследней и т.д.
6. Дан текстовый файл. Используя стек, вычислить значение логического выражения, записанного в текстовом файле в следующей форме:

< ЛВ > ::= T | F | (N<ЛВ>) | (<ЛВ>A<ЛВ>) | (<ЛВ>X<ЛВ>) | (<ЛВ>O<ЛВ>), где буквами обозначены логические константы и операции:

T – True, F – False, N – Not, A – And, X – Xor, O – Or.

1. Дан текстовый файл. В текстовом файле записана формула следующего вида:

<Формула> ::= <Цифра> | M(<Формула>,<Формула>) | N(Формула>,<Формула>)

< Цифра > ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

где буквами обозначены функции:

M – определение максимума, N – определение минимума.

Используя стек, вычислить значение заданного выражения.

1. Дан текстовый файл. Используя стек, проверить, является ли содержимое текстового файла правильной записью формулы вида:

< Формула > ::= < Терм > | < Терм > + < Формула > | < Терм > - < Формула >

< Терм > ::= < Имя > | (< Формула >)

< Имя > ::= x | y | z

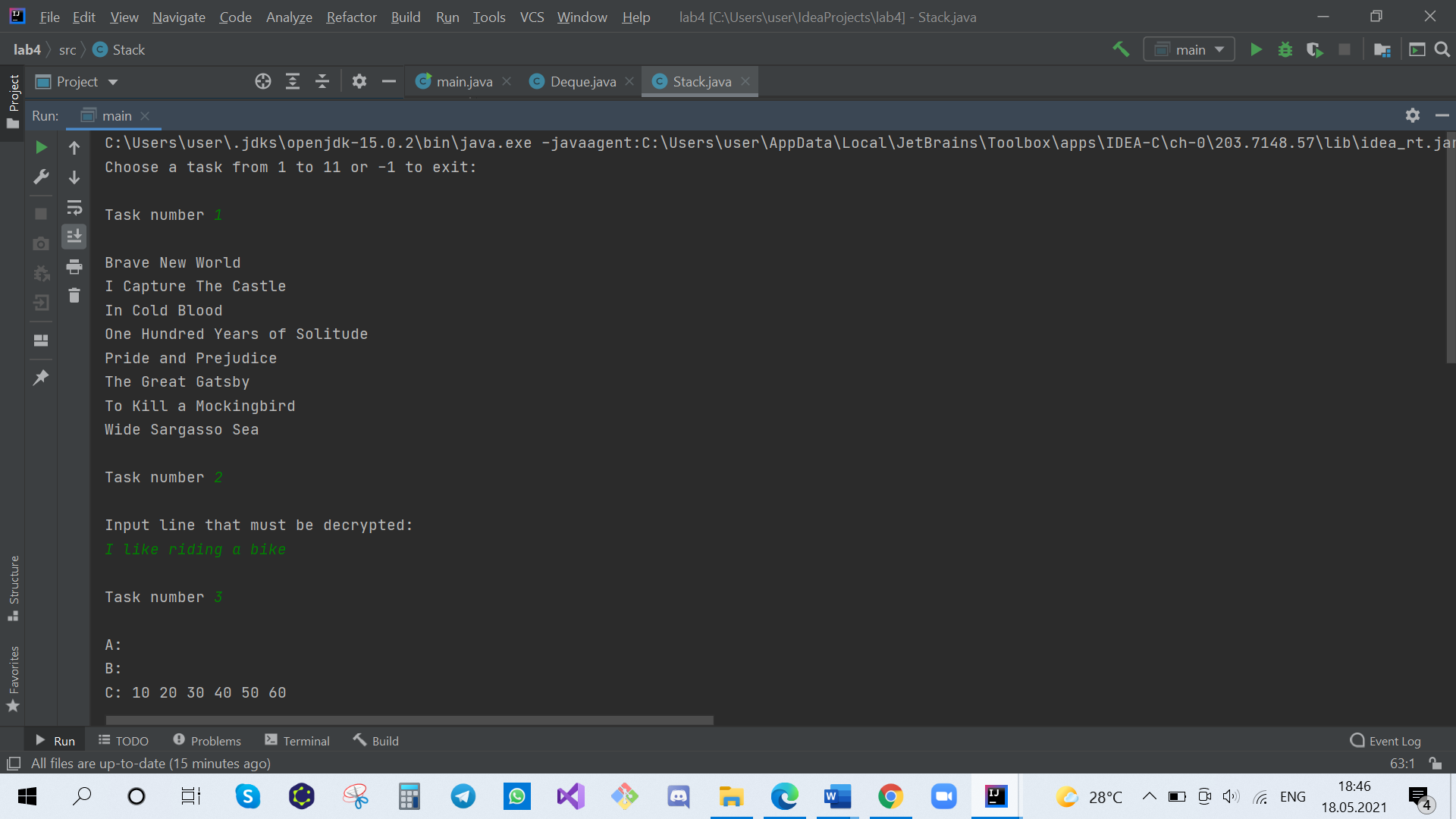
**Код программы**

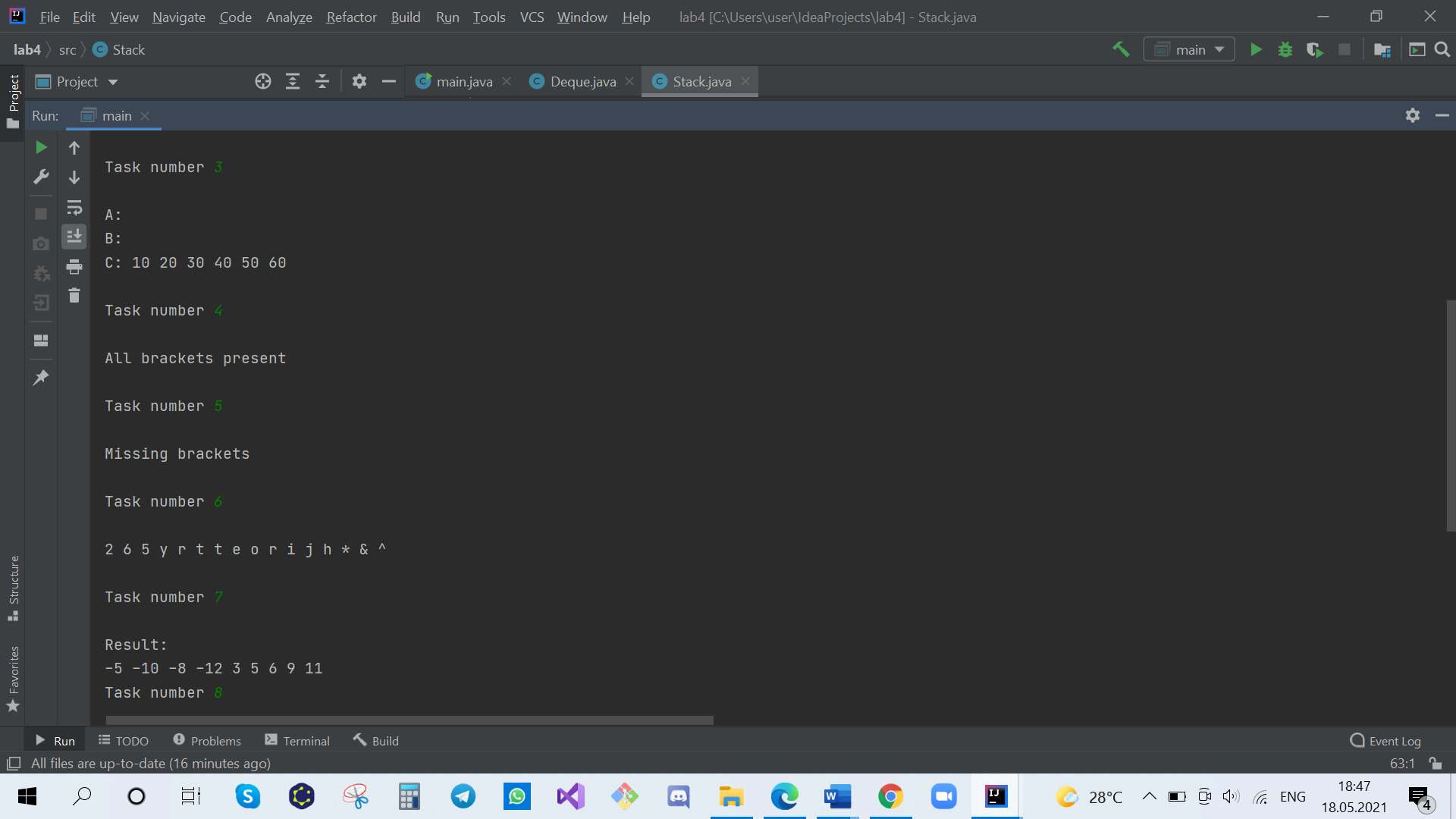
import java.io. \*;  
import java.util.\*;  
import java.lang. \*;  
  
public class main{  
 public static void main(String[] args){  
 Scanner in = new Scanner(System.*in*);  
 System.*out*.println("Choose a task from 1 to 11 or -1 to exit: ");  
 int num = 0;  
 while(num != -1){  
 System.*out*.print("\nTask number ");  
 num = in.nextInt();  
 System.*out*.println();  
 switch(num){  
 case 1:  
 *task1*();  
 break;  
 case 2:  
 *task2*();  
 break;  
 case 3:  
 *task3*();  
 break;  
 case 4:  
 *task4*();  
 break;  
 case 5:  
 *task5*();  
 break;  
 case 6:  
 *task6*();  
 break;  
 case 7:  
 *task7*();  
 break;  
 case 8:  
 *task8*();  
 break;  
 case 9:  
 *task9*();  
 break;  
 case 10:  
 *task10*();  
 break;  
 case 11:  
 *task11*();  
 default:  
 break;  
 }  
 }  
  
 }  
  
 static void task1(){  
 Deque<String> deque1 = new Deque<String>();  
 Deque<String> deque2 = new Deque<String>();  
  
 try{  
 FileReader fr = new FileReader("task1.txt");  
 BufferedReader reader = new BufferedReader(fr);  
 String line;  
  
  
 while((line = reader.readLine()) != null){  
  
 if(deque1.isEmpty())  
 deque1.addFirst(line);  
 else{  
 *help1*(line, deque1, deque2);  
 while(!deque2.isEmpty()){  
 deque1.addFirst(deque2.removeFirst());  
 }  
  
 }  
 }  
  
 while(!deque1.isEmpty()){  
 System.*out*.println(deque1.removeFirst());  
 }  
 }  
 catch(IOException ex){  
  
 System.*out*.println(ex.getMessage());  
 }  
 }  
  
 static void help1(String trueLine, Deque<String> deque1, Deque<String> deque2){  
  
 if(deque1.isEmpty()){  
 deque1.addFirst(trueLine);  
 return;  
 }else{  
 String first = deque1.getFirst();  
 String line = trueLine.toLowerCase();  
 first = first.toLowerCase();  
 line = line.replace(" ","");  
 first = first.replace(" ","");  
  
 for(int i = 0; i < line.length() && i < first.length(); i++){  
 if((int) line.charAt(i) < (int) first.charAt(i)){  
 deque1.addFirst(trueLine);  
 break;  
 }  
 else if((int) line.charAt(i) > (int) first.charAt(i)){  
 deque2.addFirst(deque1.removeFirst());  
 if(!deque1.isEmpty()){  
 first = deque1.getFirst().toLowerCase();  
 }  
 *help1*(trueLine, deque1, deque2);  
 break;  
 }  
 }  
 }  
 }  
  
 static void task2(){  
 Scanner in = new Scanner(System.*in*);  
  
 Deque<Character> deque = new Deque<Character>();  
 int iter = 0;  
 for(int i = 126; i > 31; i--){  
 deque.addFirst((char) i);  
 ++iter;  
 }  
 //получаем на вход строку, которая будет расшифровкой  
 System.*out*.println("Input line that must be decrypted: ");  
 String line = in.nextLine();  
 String sentence = "";  
 //шифруем предложение и записываем в файл  
 for(int i = 0; i < line.length(); i++){  
 while(deque.getFirst() != line.charAt(i)){  
 deque.addLast(deque.removeFirst());  
 }  
 for(int j = 0; j < 2; j++){  
 deque.addFirst(deque.removeLast());  
 }  
 sentence += deque.getFirst();  
 }  
  
 try(FileWriter writer = new FileWriter("task2\_1.txt", false)){  
 writer.write(sentence);  
  
 writer.flush();  
 }  
 catch(IOException ex){  
  
 System.*out*.println(ex.getMessage());  
 }  
 //читаем созданный нами файл  
 String line1 = "";  
 try{  
 FileReader fr = new FileReader("task2\_1.txt");  
 BufferedReader reader = new BufferedReader(fr);  
 line1 = reader.readLine();  
  
 }  
 catch(IOException ex){  
  
 System.*out*.println(ex.getMessage());  
 }  
 //расшифровываем сообщение и записываем в новый файл  
 sentence = "";  
 for(int i = 0; i < line1.length(); i++){  
 while(deque.getFirst() != line1.charAt(i)){  
 deque.addLast(deque.removeFirst());  
 }  
 for(int j = 0; j < 2; j++){  
 deque.addLast(deque.removeFirst());  
 }  
 sentence += deque.getFirst();  
 }  
 sentence = sentence.substring(0, sentence.length());  
 sentence = sentence.replace("\"","");  
 try(FileWriter writer = new FileWriter("task2\_2.txt", false)){  
 writer.write(sentence);  
  
 writer.flush();  
 }  
 catch(IOException ex){  
  
 System.*out*.println(ex.getMessage());  
 }  
  
 }  
  
 static void task3(){  
 Integer k = 0;  
 Stack<Integer> A = new Stack<>();  
 Stack<Integer> B = new Stack<>();  
 Stack<Integer> C = new Stack<>();  
  
 File f = new File("task3.txt");  
 Scanner in = new Scanner(System.*in*);  
 try{  
 in = new Scanner(f);  
 }catch(Exception e){  
 System.*out*.println("Error: " + e.getMessage());  
 return;  
 }  
 while(in.hasNextInt()){  
 A.push(in.nextInt());  
 ++k;  
 }  
 *hanoi\_tower*(A, C, B, k);  
 System.*out*.print("A: ");  
 while(!A.isEmpty())  
 System.*out*.print(A.pop() + " ");  
  
 System.*out*.print("\nB: ");  
 while(!B.isEmpty())  
 System.*out*.print(B.pop() + " ");  
 System.*out*.print("\nC: ");  
 while(!C.isEmpty())  
 System.*out*.print(C.pop() + " ");  
 System.*out*.println();  
 }  
 static void hanoi\_tower(Stack<Integer> A, Stack<Integer> B, Stack<Integer> C, int n){  
 if(n <= 0){  
 return;  
 }  
 *hanoi\_tower*(A, C, B, n - 1);  
 B.push(A.pop());  
 *hanoi\_tower*(C, B, A, n - 1);  
 }  
  
  
 static void task4(){  
 try(FileReader reader = new FileReader("task4.txt")){  
 int c;  
 Stack<Integer> stack = new Stack<>();  
 while((c = reader.read()) > 0){  
 if(c == '('){  
 stack.push(c);  
 }  
 else if(c == ')'){  
 if(stack.isEmpty()){  
 System.*out*.println("Missing brackets");  
 return;  
 }  
 else{  
 stack.pop();  
 }  
 }  
 }  
 if(stack.isEmpty()){  
 System.*out*.println("All brackets present");  
 return;  
 }  
 else{  
 System.*out*.println("Missing brackets");  
 return;  
 }  
 }  
 catch(IOException ex){  
  
 System.*out*.println(ex.getMessage());  
 }  
 }  
 static void task5(){  
 try(FileReader reader = new FileReader("task5.txt")){  
 int c;  
 Deque<Integer> deque = new Deque<Integer>();  
 while((c = reader.read()) > 0){  
 if(c == '['){  
 deque.addFirst(c);  
 }  
 else if(c == ']'){  
 if(deque.isEmpty()){  
 System.*out*.println("Missing brackets");  
 return;  
 }  
 else{  
 deque.removeFirst();  
 }  
 }  
 }  
 if(deque.isEmpty()){  
 System.*out*.println("All brackets present");  
 return;  
 }  
 else{  
 System.*out*.println("Missing brackets");  
 return;  
 }  
 }  
 catch(IOException ex){  
  
 System.*out*.println(ex.getMessage());  
 }  
 }  
 static void task6(){  
 int c;  
 Deque<Character> deque = new Deque<Character>();  
  
 try(FileReader reader = new FileReader("task6.txt")){  
 while((c = reader.read()) > 0){  
 if(c >= 48 && c <= 57){  
 deque.addFirst((char) c);  
 }  
 else if(c >= 65 && c <= 90 || c >= 97 && c <= 122){  
 if(deque.isEmpty())  
 deque.addFirst((char) c);  
 int iter = 0;  
 if(!((int) deque.getFirst() >= 48 && (int) deque.getFirst() <= 57)){  
 deque.addFirst((char) c);  
 }  
 else{  
  
 while(!(deque.getLast() >= 48 && deque.getLast() <= 57) && !(deque.getLast() >= 65 && deque.getLast() <= 90 || deque.getLast() >= 97 && deque.getLast() <= 122)){  
 deque.addFirst(deque.removeLast());  
 ++iter;  
 }  
 deque.addLast((char) c);  
 for(int i = 0; i < iter; i++){  
 deque.addLast(deque.removeFirst());  
 }  
 }  
 }  
 else{  
 deque.addLast((char) c);  
 }  
  
 }  
 }  
 catch(IOException ex){  
  
 System.*out*.println(ex.getMessage());  
 }  
  
 while(!(deque.isEmpty())){  
 System.*out*.print(deque.removeFirst() + " ");  
 }  
 System.*out*.println();  
 }  
  
 static void task7(){  
 int c;  
 String line = "";  
 Deque<Integer> deque = new Deque<Integer>();  
  
 try(FileReader reader = new FileReader("task7.txt")){  
 while((c = reader.read()) > 0){  
 if(c != ' '){  
 line += (char) c;  
 }  
 else{  
 if(Integer.*parseInt*(line) >= 0){  
 deque.addLast(Integer.*parseInt*(line));  
 }  
 else{  
 if(deque.isEmpty() || deque.getFirst() >= 0)  
 deque.addFirst(Integer.*parseInt*(line));  
 else{  
 int iter = 0;  
 if(deque.getLast() >= 0){  
 while(deque.getFirst() < 0){  
 deque.addLast(deque.removeFirst());  
 ++iter;  
 }  
 deque.addFirst(Integer.*parseInt*(line));  
 for(int i = 0; i < iter; i++){  
 deque.addFirst(deque.removeLast());  
 }  
 }  
 else{  
 for(int i = 0; i < deque.size(); i++){  
 deque.addLast(deque.removeFirst());  
 }  
 deque.addFirst(Integer.*parseInt*(line));  
 for(int i = 0; i < deque.size() - 1; i++){  
 deque.addFirst(deque.removeLast());  
 }  
 }  
 }  
 }  
 line = "";  
 continue;  
  
 }  
 }  
 }  
 catch(IOException ex){  
  
 System.*out*.println(ex.getMessage());  
 }  
  
  
 System.*out*.println("Result: ");  
  
 while(!deque.isEmpty()){  
 System.*out*.print(deque.removeFirst() + " ");  
 }  
 }  
 static void task8(){  
 Stack<String> stack = new Stack<>();  
  
 try{  
 FileReader fr = new FileReader("task8.txt");  
 BufferedReader reader = new BufferedReader(fr);  
 String line;  
  
 while((line = reader.readLine()) != null){  
 stack.push(line);  
 }  
 }  
 catch(IOException ex){  
  
 System.*out*.println(ex.getMessage());  
 }  
  
 try(FileWriter writer = new FileWriter("task8\_1.txt", false))  
 {  
 // запись всей строки  
 while(!stack.isEmpty()){  
 writer.write(stack.pop() + "\n");  
 }  
 writer.flush();  
 }  
 catch(IOException ex){  
  
 System.*out*.println(ex.getMessage());  
 }  
 }  
  
 static void task9(){  
 Stack<Integer> stack = new Stack<Integer>();  
 String line = "";  
 try(FileReader reader = new FileReader("task9.txt")){  
 int c;  
 while((c = reader.read()) > 0){  
 if(c == ' ')  
 continue;  
 if(c == 'T' || c == 'F'){  
 line += (char) c;  
 }  
 else if(c == '('){  
 stack.push(c);  
 }  
 else if(c == ')'){  
 int r;  
 while((!stack.isEmpty()) && (r = stack.pop()) != '('){  
 line += (char) r;  
 }  
 }  
 else if(c == 'N'){  
 stack.push(c);  
 }  
 else if(c == 'A' || c == 'X' || c == 'O'){  
 int r = 0;  
 while(!(stack.isEmpty()) && *priority*(r = stack.pop()) >= *priority*(c)){  
 line += (char) r;  
 }  
 if(!stack.isEmpty())  
 stack.push(r);  
 stack.push(c);  
 }  
 }  
 }  
 catch(IOException ex){  
  
 System.*out*.println(ex.getMessage());  
 }  
 while(!stack.isEmpty()){  
 int a = stack.pop();  
 line += (char) a;  
 }  
  
 System.*out*.println(line);  
  
 for(int i = 0; i < line.length(); i++){  
 if(line.charAt(i) == 'T' || line.charAt(i) == 'F'){  
 stack.push((int) line.charAt(i));  
 }  
 else if(line.charAt(i) == 'N'){  
 int a = stack.pop();  
 if(a == 'F')  
 a = (int) 'T';  
 else if(a == 'T')  
 a = (int) 'F';  
 stack.push(a);  
 }  
 else if(line.charAt(i) == 'A'){  
 int a = stack.pop();  
 int b = stack.pop();  
 if(a == 'F' || b == 'F'){  
 stack.push((int) 'F');  
 }  
 else{  
 stack.push((int) 'T');  
 }  
 }  
 else if(line.charAt(i) == 'O'){  
 int a = stack.pop();  
 int b = stack.pop();  
 if(a == 'F' && b == 'F'){  
 stack.push((int) 'F');  
 }  
 else{  
 stack.push((int) 'T');  
 }  
 }  
 else if(line.charAt(i) == 'X'){  
 int a = stack.pop();  
 int b = stack.pop();  
 if(a == b){  
 stack.push((int) 'F');  
 }  
 else{  
 stack.push((int) 'T');  
 }  
 }  
 }  
 int p = stack.pop();  
 line = "" + (char) p;  
 System.*out*.println(line);  
 }  
 static int priority(int c){  
 switch (c){  
 case '(': case ')':  
 return 0;  
 case 'X':  
 return 1;  
 case 'O':  
 return 1;  
 case 'A':  
 return 2;  
 case 'N':  
 return 3;  
 default:  
 return 0;  
 }  
 }  
  
 static void task10(){  
 Stack<Integer> stack = new Stack<Integer>();  
 String line = "";  
 try(FileReader reader = new FileReader("task10.txt")){  
 int c;  
 while((c = reader.read()) > 0){  
 if(c == ' ')  
 continue;  
 if(c >= '0' && c <= '9'){  
 line += (char) c;  
 }  
 else if(c == 'M' || c == 'N'){  
 line += (char) c;  
 }  
 }  
 }  
 catch(IOException ex){  
  
 System.*out*.println(ex.getMessage());  
 }  
 while(!stack.isEmpty()){  
 int a = stack.pop();  
 line += (char) a;  
 }  
  
 System.*out*.println("Polish notation: " + line);  
  
 for(int i = line.length() - 1; i >= 0; i--){  
 if(line.charAt(i) >= '0' && line.charAt(i) <= '9'){  
 stack.push((int) line.charAt(i));  
 }  
 else if(line.charAt(i) == 'N'){  
 int a = stack.pop();  
 int b = stack.pop();  
 if(a < b){  
 stack.push(a);  
 }  
 else{  
 stack.push(b);  
 }  
 }  
 else if(line.charAt(i) == 'M'){  
 int a = stack.pop();  
 int b = stack.pop();  
 if(a > b){  
 stack.push(a);  
 }  
 else{  
 stack.push(b);  
 }  
 }  
 }  
 int p = stack.pop();  
 line = "" + (char) p;  
 System.*out*.println("Result: " + line);  
 }  
  
 static void task11(){  
 Stack<Integer> stack = new Stack<Integer>();  
 String line = "";  
 try(FileReader reader = new FileReader("task11.txt")){  
 int c;  
 while((c = reader.read()) > 0){  
 if(c == ' ')  
 continue;  
 if(c == 'x' || c == 'y' || c == 'z'){  
 line += (char) c;  
 }  
 else if(c == '('){  
 stack.push(c);  
 }  
 else if(c == ')'){  
 int r = -1;  
 while((!stack.isEmpty()) && (r = stack.pop()) != '('){  
 line += (char) r;  
 }  
 if(stack.isEmpty() && r == -1){  
 System.*out*.println("Ops");  
 return;  
 }  
 }  
 else if(c == '+' || c == '-'){  
 int r = 0;  
 while(!(stack.isEmpty()) && *priority1*(r = stack.pop()) >= *priority1*(c)){  
 line += (char) r;  
 }  
 if(!stack.isEmpty())  
 stack.push(r);  
 stack.push(c);  
 }  
 }  
 }  
 catch(IOException ex){  
  
 System.*out*.println(ex.getMessage());  
 }  
 while(!stack.isEmpty()){  
 int a = stack.pop();  
 if(a != '+' && a != '-'){  
 System.*out*.println("Ops");  
 return;  
 }  
 line += (char) a;  
 }  
 int l1 = 0;  
 int l2 = 0;  
 for(int i = 0; i < line.length(); i++){  
 if(line.charAt(i) == 'x' || line.charAt(i) == 'y' || line.charAt(i) == 'z'){  
 l1++;  
 }  
 else if(line.charAt(i) == '+' || line.charAt(i) == '-'){  
 l2++;  
 }  
 }  
 if(l1 - l2 != 1){  
 System.*out*.println("Ops");  
 return;  
 }  
 System.*out*.println(line);  
 }  
 static int priority1(int c){  
 switch (c){  
 case '(': case ')':  
 return 0;  
 case '+': case '-':  
 return 1;  
 default:  
 return 0;  
 }  
 }  
}

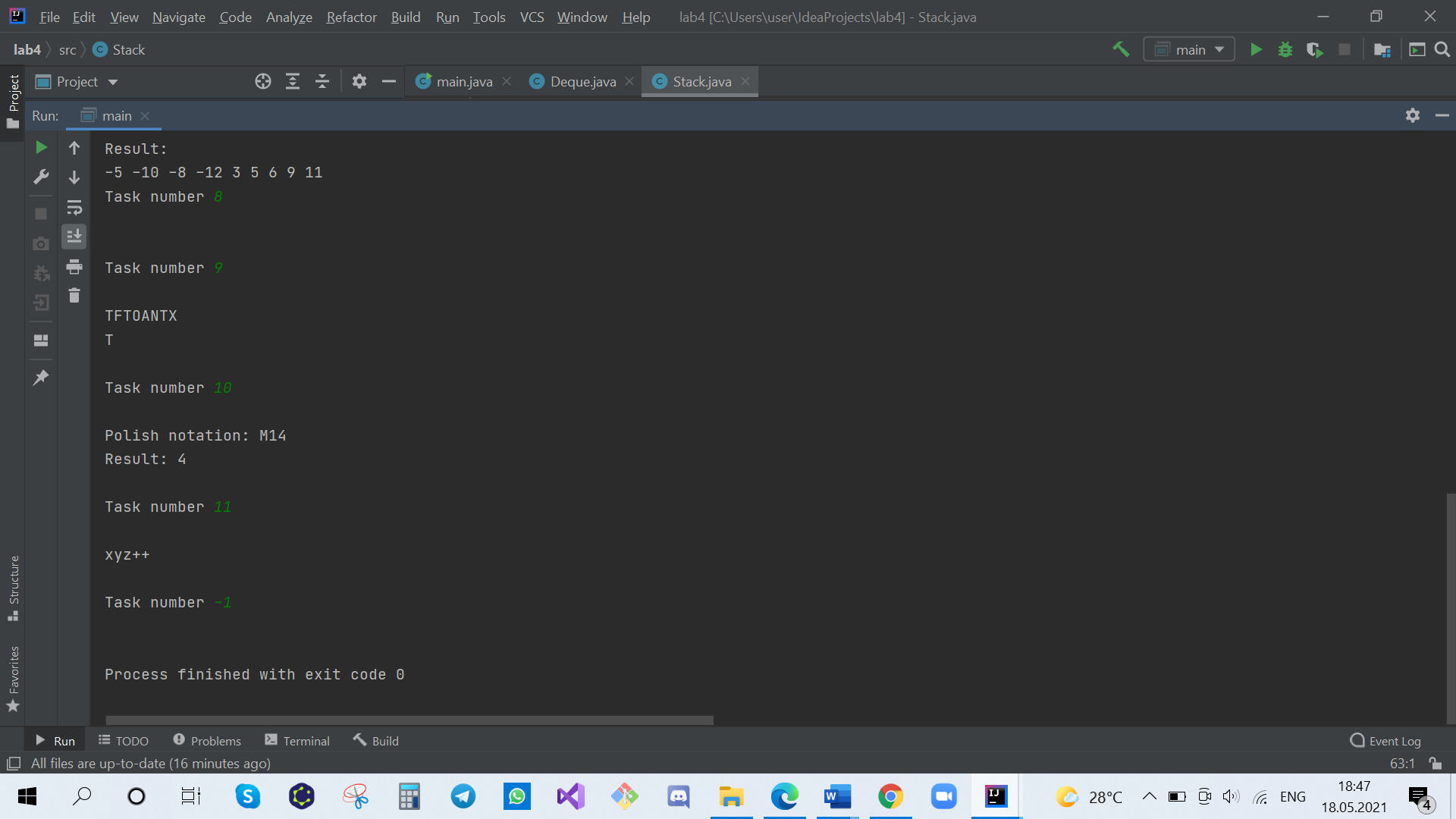
import java.io. \*;  
import java.util.\*;  
import java.lang. \*;  
  
public class Deque<T>{  
 private LinkedList<T> deque = new LinkedList<T>();  
  
 public void addFirst(T e){  
 deque.addFirst(e);  
 }  
  
 public void addLast(T e){  
 deque.addLast(e);  
 }  
  
 public T getFirst(){  
 return deque.getFirst();  
 }  
  
 public T getLast(){  
 return deque.getLast();  
 }  
  
 public T removeFirst(){  
 return deque.removeFirst();  
 }  
  
 public T removeLast(){  
 return deque.removeLast();  
 }  
  
 public int size(){  
 return deque.size();  
 }  
  
 public String toString(){  
 return deque.toString();  
 }  
  
 public boolean isEmpty(){  
 if(deque.size() == 0)  
 return true;  
 return false;  
 }  
}

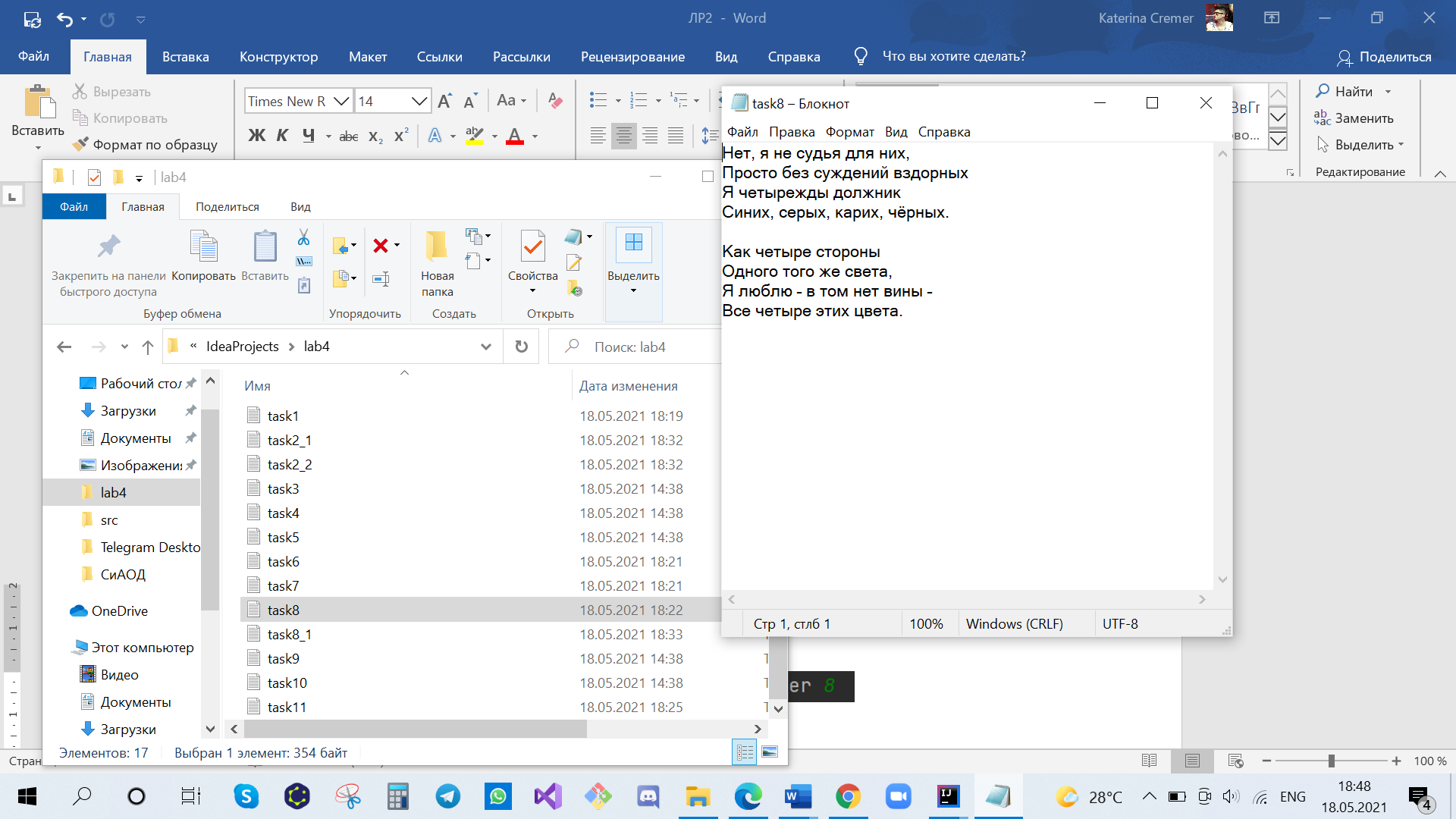
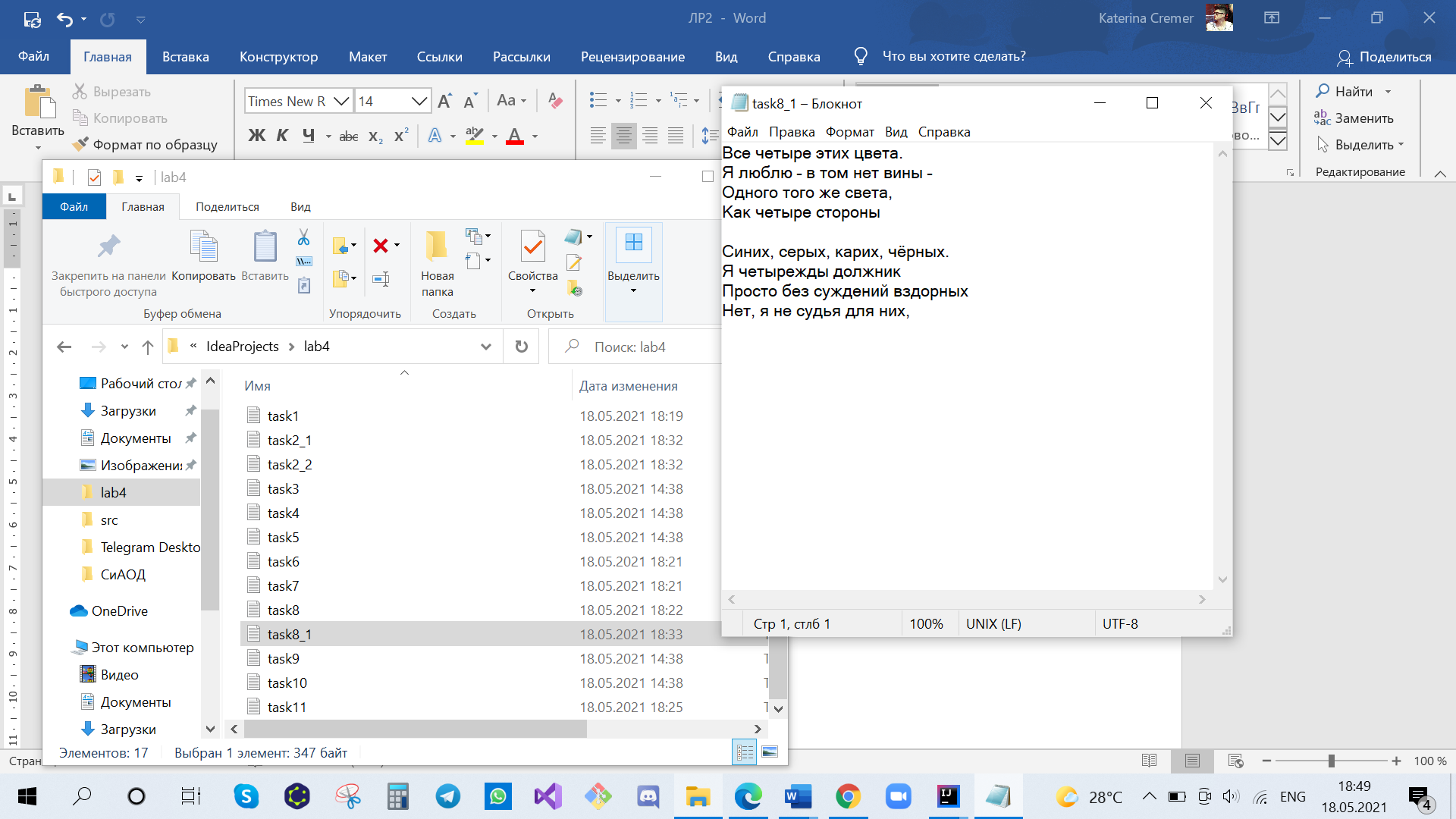
import java.io. \*;  
import java.util.\*;  
import java.lang. \*;  
  
public class Stack<T>{  
 private static class Node<U>{  
 U item;  
 Node<U> next;  
  
 Node(){  
 item = null;  
 next = null;  
 }  
  
 Node(U item, Node<U> next){  
 this.item = item;  
 this.next = next;  
 }  
  
 boolean end(){  
 return (item == null && next == null);  
 }  
 }  
  
 private Node<T> top = new Node<T>();  
  
 public void push(T item){  
 top = new Node<T>(item, top);  
 }  
  
 public T pop(){  
 T result = top.item;  
 if(!top.end())  
 top = top.next;  
 return result;  
 }  
  
 public boolean isEmpty(){  
 return top.end();  
 }  
}

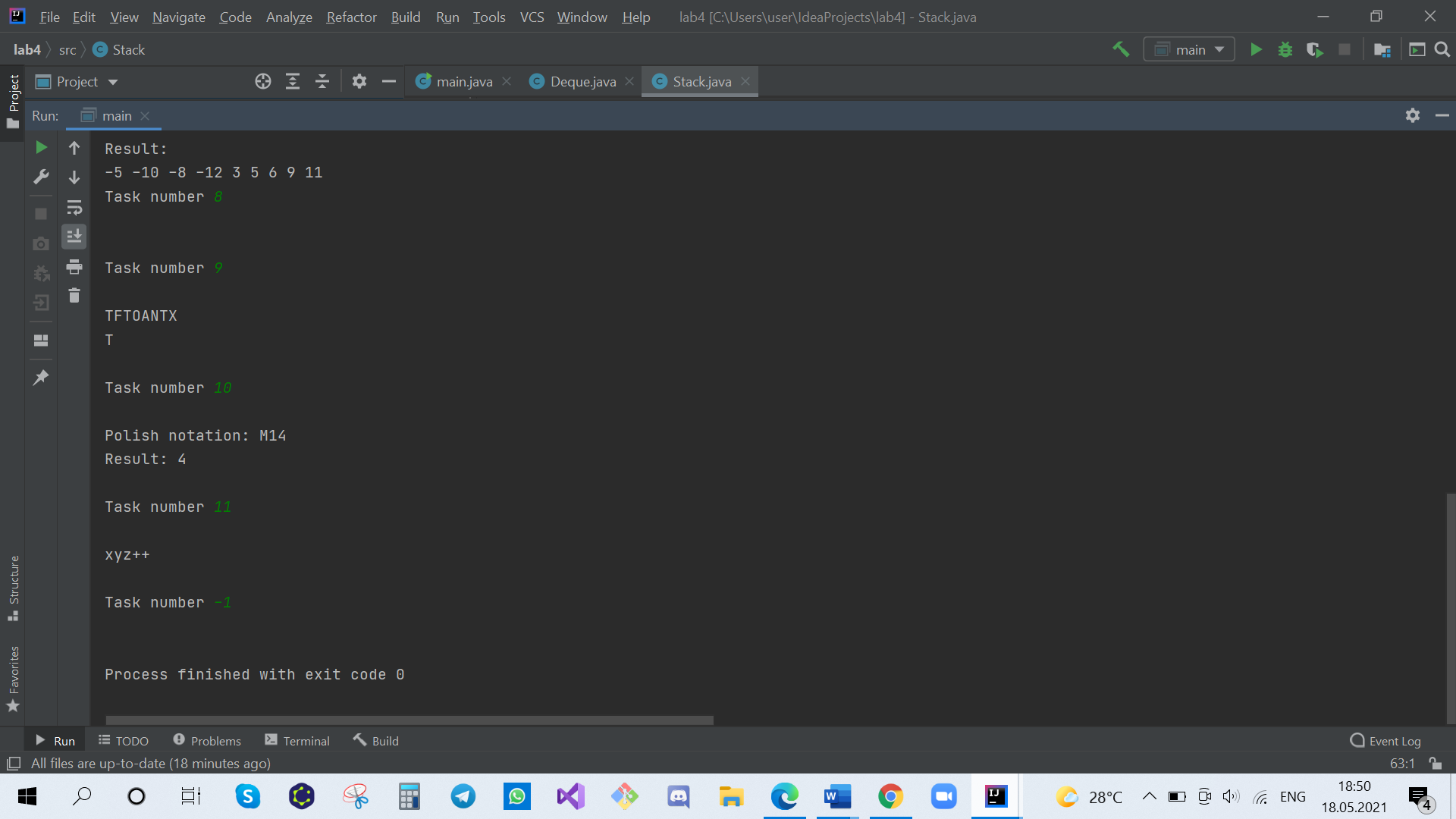
**Результат работы программы**









**Вывод**

В ходе данной лабораторной работы я реализовала структуры данных (стек и дек) и с помощью них выполнила поставленные задачи.