|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ \_\_\_\_ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА \_\_\_\_\_\_КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.01 Информатика и вычислительная техника**

МАГИСТЕРСКАЯ ПРОГРАММА **09.04.01/12 Интеллектуальный анализ больших**

**данных в системах поддержки принятия решений.**

**Отчет**

**по лабораторной работе № 6**

**Название:** Коллекции

**Дисциплина:** Языки программирования для работы с большими данными

Студент \_\_\_ИУ6-23М\_\_\_\_ **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_**А.Д. Зеленский**\_\_\_**

(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_**П. В. Степанов**\_\_\_\_**

(Подпись, дата) (И.О.Фамилия)

**Цель:** ознакомиться с базовыми принципами языка Java для работы с большими данными.

**Вариант 1:**

1. Задана строка, состоящая из символов '(', ')', '[', ']', '{', '}'. Проверить правильность расстановки скобок. Использовать стек.
2. Задан файл с текстом на английском языке. Выделить все различные слова. Слова, отличающиеся только регистром букв, считать одинаковыми. Использовать класс HashSet.

**Решение:**

Ниже представлен код класса первой программы, а на рисунке 1 – результат работы программы.

import java.util.Scanner;

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

System.out.print("А) Введите строку: ");

StringBuilder s = new StringBuilder(input.nextLine());

Stack<Character> stac = new Stack<>();

for (int i = 0; i < s.length(); i++) {

if ((s.charAt(i) == '{') || (s.charAt(i) == '[') || (s.charAt(i) == '(')) {

stac.add(s.charAt(i));

}

if ((s.charAt(i) == '}') || (s.charAt(i) == ']') || (s.charAt(i) == ')')) {

if ((s.charAt(i) == '}') && (stac.peek() == '{')) stac.pop();

if ((s.charAt(i) == ']') && (stac.peek() == '[')) stac.pop();

if ((s.charAt(i) == ')') && (stac.peek() == '(')) stac.pop();

}

}

if (stac.isEmpty()) System.out.print("Все скобки расставленный верно");

else System.out.print("Скобки расставленный неверно");

}

}

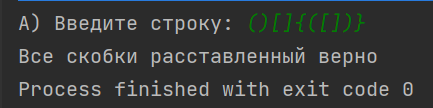


Рисунок 1 – Результат работы программы

Ниже представлен код класса второй программы, а на рисунке 2 – результат работы программы.

import java.io.\*;

import java.util.\*;

public class Main {

public static void main(String[] args) throws Exception {

FileReader fr= new FileReader("C:\\Users\\artem.zelenskiy\\IdeaProjects\\Lab\_6\\Lab\_6\_2\\src\\test.txt");

Scanner scanner = new Scanner(fr);

ArrayList<String> words = new ArrayList<String>();

StringBuilder oneLongString =new StringBuilder();

HashSet<String> hasChar =new HashSet<>();

while (scanner.hasNextLine()) {

String s = scanner.nextLine().replaceAll("[.,?!:-;']", "").toUpperCase();

words.addAll(Arrays.asList(s.split(" ")));

oneLongString.append(s);

}

for ( String str : words )

if (!hasChar.contains(str)) hasChar.add(str);

System.out.println("--------------------------------------------");

System.out.println("Все различные слова: ");

System.out.println(hasChar);

}

}

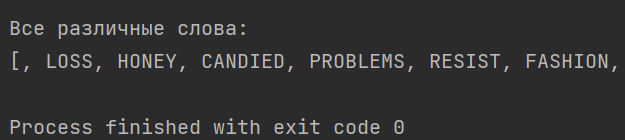


Рисунок 2 – Результат работы программы

**Вариант 2:**

1. На клетчатом листе бумаги закрашена часть клеток. Выделить все различные фигуры, которые образовались при этом. Фигурой считается набор закрашенных клеток, достижимых друг из друга при движении в четырёх направлениях. Две фигуры являются различными, если их нельзя совместить поворотом на угол, кратный 90 градусам, и параллельным переносом. Используйте класс HashSet.
2. Дана матрица из целых чисел. Найти в ней прямоугольную подматрицу, состоящую из максимального количества одинаковых элементов. Использовать класс Stack.

**Решение:**

Ниже представлен кусок кода первой программы, а на рисунке 3 – результат работы программы.

import java.util.\*;

class FiguresOnGrid {

public static void main(String[] args) {

boolean[][] grid = {

{true, true, false, false},

{true, false, true, false},

{false, true, false, false},

{false, false, false, true}

};

Set<Set<String>> figures = findFigures(grid);

System.out.println(figures);

}

public static Set<Set<String>> findFigures(boolean[][] grid) {

int n = grid.length;

int m = grid[0].length;

boolean[][] visited = new boolean[n][m];

Set<Set<String>> figures = new HashSet<>();

for (int i = 0; i < n; i++) {

for (int j = 0; j < m; j++) {

if (grid[i][j] && !visited[i][j]) {

Set<String> figure = new HashSet<>();

dfs(grid, visited, figure, i, j,i,j);

figures.add(figure);

}

}

}

return figures;

}

public static void dfs(boolean[][] grid, boolean[][] visited, Set<String> figure, int i, int j, int i\_min, int j\_min) {

int n = grid.length;

int m = grid[0].length;

visited[i][j] = true;

figure.add("{"+(i-i\_min) +", "+ (j-j\_min)+"}");

if (i > 0 && grid[i - 1][j] && !visited[i - 1][j]) {

dfs(grid, visited, figure, i - 1, j,i\_min,j\_min);

}

if (i < n - 1 && grid[i + 1][j] && !visited[i + 1][j]) {

dfs(grid, visited, figure, i + 1, j,i\_min,j\_min);

}

if (j > 0 && grid[i][j - 1] && !visited[i][j - 1]) {

dfs(grid, visited, figure, i, j - 1,i\_min,j\_min);

}

if (j < m - 1 && grid[i][j + 1] && !visited[i][j + 1]) {

dfs(grid, visited, figure, i, j + 1,i\_min,j\_min);

}

}

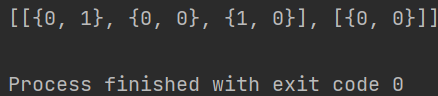
}

Рисунок 3 – Результат работы программы

Ниже представлен часть кода второй программы, а на рисунке 4 – результат работы программы.

import java.util.HashSet;

import java.util.\*;

class MaxRectangle {

public static void main(String[] args) {

int[][] matrix =

{{0, 0, 0, 1, 1, 5, 6, 0, 8, 9},

{0, 0, 0, 1, 1, 5, 6, 8, 8, 9},

{0, 1, 1, 3, 3, 5, 6, 8, 8, 9},

{0, 1, 1, 1, 1, 5, 6, 8, 8, 9},

{1, 1, 1, 3, 3, 5, 6, 8, 8, 9}};

String rez = getMaxRectangleArea(matrix);

System.out.println("Ответ: " + rez);

}

private static String getMaxRectangleArea(int[][] matrix) {

Stack<Integer> stack = new Stack<>();

Set<Integer> numbers = new HashSet<>();

int rows = matrix.length;

int cols = matrix[0].length;

int maxArea = 0;

int numb\_ax=0;

// Заношу в стек все неповторяющиеся цифры

for (int i = 0; i < rows; i++) {

for (int j = 0; j < cols; j++) {

if ( !numbers.contains(matrix[i][j])) stack.push(matrix[i][j]);

}

}

// Прохожусь по каждому объекту матрицы

for (int st=0;st<stack.size();st++) {

int sta = stack.pop();

int[] heights = new int[cols];

for (int i = 0; i < rows; i++) {

for (int j = 0; j < cols; j++) {

if (matrix[i][j] != sta) {

heights[j] = 0;

} else {

heights[j]++;

}

}

int area = getMaxHistogramArea(heights);

if (area > maxArea) {

maxArea = area;

numb\_ax=sta;

}

}

}

return " Максимальная площадь равна " +maxArea +"; из цифры " +numb\_ax;

}

private static int getMaxHistogramArea(int[] heights) {

Stack<Integer> stack = new Stack<>();

int maxArea = 0;

int i = 0;

while (i < heights.length) {

if (stack.isEmpty() || heights[i] >= heights[stack.peek()]) {

stack.push(i++);

} else {

int topIndex = stack.pop();

int area = heights[topIndex] \* (stack.isEmpty() ? i : i - stack.peek() - 1);

if (area > maxArea) {

maxArea = area;

}

}

}

while (!stack.isEmpty()) {

int topIndex = stack.pop();

int area = heights[topIndex] \* (stack.isEmpty() ? i : i - stack.peek() - 1);

if (area > maxArea) {

maxArea = area;

}

}

return maxArea;

}

}



Рисунок 4 – Результат работы программы

**Вывод:** были получены базовые представления о работе языка java с коллекциями. Были написаны программы согласно вариантам.