|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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 (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.01 Информатика и вычислительная техника**

МАГИСТЕРСКАЯ ПРОГРАММА **09.04.01/07 Интеллектуальные системы анализа,**

**обработки и интерпретации больших данных.**

**Отчет**

|  |  |
| --- | --- |
| **по лабораторной работе №** | 6 |
| **Вариант №** 8 |  |

**Название:**

Коллекции

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент | ИУ6-23М | |  |  | Н.М. Иванюк |
|  | | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  | |  |  |  |  |
| Преподаватель | |  |  |  | П.В. Степанов |
|  | |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2022

## Цель работы

Получение первичных навыков работы с коллекциями языка программирования Java.

## Ход работы

*Задание 1.*

8. Задана строка, состоящая из символов '(', ')', '[', ']', '{', '}'. Проверить правильность расстановки скобок. Использовать стек.

9. Задан файл с текстом на английском языке. Выделить все различные слова. Слова, отличающиеся только регистром букв, считать одинаковыми. Использовать класс HashSet.

Листинг 1 – Код программы первого задания

import java.io.File;

import java.util.ArrayList;

import java.util.HashSet;

import java.util.Scanner;

import java.util.Stack;

public class App {

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

String exp = "[()]{}{[()()]()}";

String exp\_nb = "[{}";

// Function call

if (areBracketsBalanced(exp))

System.out.println("Balanced ");

else

System.out.println("Not Balanced ");

if (areBracketsBalanced(exp\_nb))

System.out.println("Balanced ");

else

System.out.println("Not Balanced ");

System.out.println("Отрабатываем английский файл");

File poem = new File("data\\text.txt");

Scanner reader = new Scanner(poem, "UTF-8");

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

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

while (reader.hasNextLine()) {

System.out.print(".");

rows.add(reader.nextLine());

}

System.out.println("\n");

for (String r:rows) {

String[] splitted = r.split(" ");

for (String s:splitted) {

words.add(s.toLowerCase());

}

}

for (String w:words) {

System.out.println(w);

}

}

static boolean areBracketsBalanced(String expr)

{

Stack<Character> stack

= new Stack<Character>();

// Traversing the Expression

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

{

char x = expr.charAt(i);

if (x == '(' || x == '[' || x == '{')

{

// Push the element in the stack

stack.push(x);

continue;

}

// If current character is not opening

// bracket, then it must be closing. So stack

// cannot be empty at this point.

if (stack.isEmpty())

return false;

char check;

switch (x) {

case ')':

check = stack.pop();

if (check == '{' || check == '[')

return false;

break;

case '}':

check = stack.pop();

if (check == '(' || check == '[')

return false;

break;

case ']':

check = stack.pop();

if (check == '(' || check == '{')

return false;

break;

}

}

// Check Empty Stack

return (stack.isEmpty());

}

}

Приведем результаты выполнения данного кода.

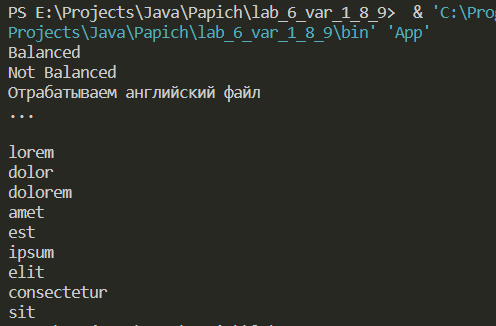


Рисунок 1 – Результат выполнения варианта задания 1

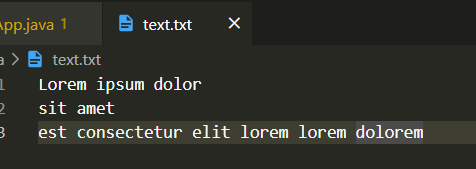


Рисунок 2 – Входной файл

*Задание 2.*

8. 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.

9. Дана матрица из целых чисел. Найти в ней прямоугольную подматрицу, состоящую из максимального количества одинаковых элементов. Использовать класс Stack.

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.

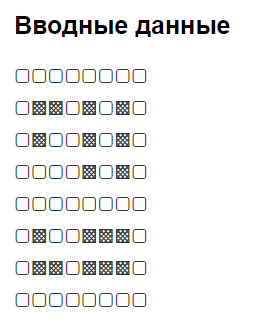


Рисунок 4 – Матрица клеток, подаваемая на ввод

Листинг 2 – Код класса «Point2D»

public class Point2D {

int x, y;

public Point2D(int x, int y) {

this.x = x;

this.y = y;

}

@Override

public boolean equals(Object obj) {

// Если сравниваем с самим собой

if (obj == this) {

return true;

}

// Если это не тот класс

if (obj == null || getClass() != obj.getClass()) return false;

// Теперь можем сравнить поля

Point2D p = (Point2D) obj;

return this.x == p.x && this.y == p.y;

}

@Override

public int hashCode() {

return 2\*x + 3\*y;

}

}

Листинг 3 – Код класса «BoundaryBox»

public class BoundaryBox {

Point2D min;

Point2D max;

public BoundaryBox(Point2D min, Point2D max) {

this.min = min;

this.max = max;

}

}

Листинг 4 – Код класса «Figure»

import java.util.HashSet;

public class Figure {

HashSet<Point2D> points = new HashSet<>();

// Добавление точки

public void AddPoint(int x, int y) {

points.add(new Point2D(x, y));

}

// Сравнение фигур

@Override

public boolean equals(Object obj) {

// Если сравниваем с самим собой

if (obj == this) {

return true;

}

// Если это не фигура

if (obj == null || getClass() != obj.getClass()) return false;

// Теперь можем сравнить поля

Figure fig = (Figure) obj;

// Сравнить со всеми возможными поворотами фигуры

boolean equal = false;

Figure normalized = this.Normalized();

// Поворот на 90

Figure fig90 = fig.Rotated90().Normalized();

equal |= fig90.points.equals(normalized.points);

// Поворот на 180

Figure fig180 = fig90.Rotated90().Normalized();

equal |= fig180.points.equals(normalized.points);

// Поворот на 270

Figure fig270 = fig180.Rotated90().Normalized();

equal |= fig270.points.equals(normalized.points);

return equal;

}

@Override

public int hashCode() {

return points.size();

}

// Красивый вывод фигуры

@Override

public String toString() {

String result = "";

BoundaryBox boundary = getBoundaryBox();

for (int j = boundary.min.y - 1; j <= boundary.max.y + 1; j++) {

for (int i = boundary.min.x - 1; i <= boundary.max.x + 1; i++) {

if (points.contains(new Point2D(i, j))) {

result += "\u25FC";

} else {

result += "\u25FB";

}

}

result += "\n";

}

return result;

}

// Нормализация координат для сравнения

public Figure Normalized() {

// Создаем новую фигуру, которую потом вернем

Figure normalizedFigure = new Figure();

// Найдем boundary box для точек

Point2D min = getBoundaryBox().min;

// Сдвинем очерченную фигуру в начало координат

for (Point2D p : points) {

normalizedFigure.AddPoint(p.x - min.x, p.y - min.y);

}

// Вернем новую нормализованную фигуру

return normalizedFigure;

}

// Повернем фигуру на 90 градусов вокруг (0,0)

public Figure Rotated90() {

// Создаем новую фигуру, которую потом вернем

Figure rotatedFigure = new Figure();

// Проходим по всем точкам и поворачиваем их как (-y, x)

for (Point2D p : points) {

rotatedFigure.AddPoint(-p.y, p.x);

}

// Вернем повернутую фигуру

return rotatedFigure;

}

// Определение boundary box для фигуры

private BoundaryBox getBoundaryBox() {

int min\_x = Integer.MAX\_VALUE;

int min\_y = Integer.MAX\_VALUE;

int max\_x = Integer.MIN\_VALUE;

int max\_y = Integer.MIN\_VALUE;

for (Point2D p : points) {

if (p.x < min\_x) min\_x = p.x;

if (p.x > max\_x) max\_x = p.x;

if (p.y < min\_y) min\_y = p.y;

if (p.y > max\_y) max\_y = p.y;

}

Point2D min = new Point2D(min\_x, min\_y);

Point2D max = new Point2D(max\_x, max\_y);

return new BoundaryBox(min, max);

}

}

Листинг 5 – Код выполнения основной программы

import java.util.HashSet;

import java.util.LinkedList;

import java.util.Queue;

public class App {

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

// Задаем поле из NxN клеток

int[][] field = {

{0, 0, 0, 0, 0, 0, 0, 0}, // ▢▢▢▢▢▢▢▢

{0, 1, 1, 0, 1, 0, 1, 0}, // ▢▩▩▢▩▢▩▢

{0, 1, 0, 0, 1, 0, 1, 0}, // ▢▩▢▢▩▢▩▢

{0, 0, 0, 0, 1, 0, 1, 0}, // ▢▢▢▢▩▢▩▢

{0, 0, 0, 0, 0, 0, 0, 0}, // ▢▢▢▢▢▢▢▢

{0, 1, 0, 0, 1, 1, 1, 0}, // ▢▩▢▢▩▩▩▢

{0, 1, 1, 0, 1, 1, 1, 0}, // ▢▩▩▢▩▩▩▢

{0, 0, 0, 0, 0, 0, 0, 0} // ▢▢▢▢▢▢▢▢

};

// Задаем множество фигур

HashSet<Figure> figures = new HashSet<>();

// Проходим по каждой клетке, ищем смежные

for (int y = 0; y < field.length; y++) {

for (int x = 0; x < field[y].length; x++) {

// Если натыкаемся на закрашенную клетку, начинаем Flood Fill

if (field[x][y] == 1) {

// Создаем новый объект `Фигура`, в который записываем точки

Figure figure = new Figure();

// Заводим очередь точек

Queue<Point2D> points = new LinkedList<>();

points.add(new Point2D(x, y));

// Пока очередь не опустеет, ищем смежные

while (!points.isEmpty()) {

Point2D p = points.remove();

figure.AddPoint(p.x, p.y);

Point2D n1 = new Point2D(p.x - 1, p.y);

if (field[n1.x][n1.y] == 1 && !figure.points.contains(n1)) points.add(new Point2D(n1.x, n1.y));

Point2D n2 = new Point2D(p.x + 1, p.y);

if (field[n2.x][n2.y] == 1 && !figure.points.contains(n2)) points.add(new Point2D(n2.x, n2.y));

Point2D n3 = new Point2D(p.x, p.y - 1);

if (field[n3.x][n3.y] == 1 && !figure.points.contains(n3)) points.add(new Point2D(n3.x, n3.y));

Point2D n4 = new Point2D(p.x, p.y + 1);

if (field[n4.x][n4.y] == 1 && !figure.points.contains(n4)) points.add(new Point2D(n4.x, n4.y));

}

figures.add(figure);

// Удаляем с поля найденную фигуру (нужно предусмотреть какой-то буфер клеток)

for (Point2D p : figure.points) {

field[p.x][p.y] = 0;

}

}

}

}

System.out.println("Total figures found: " + figures.size() + "\n");

// Выводим найденные фигуры

for (Figure f : figures) {

System.out.println(f.toString());

}

}

}

Приведем результаты выполнения работы программы.

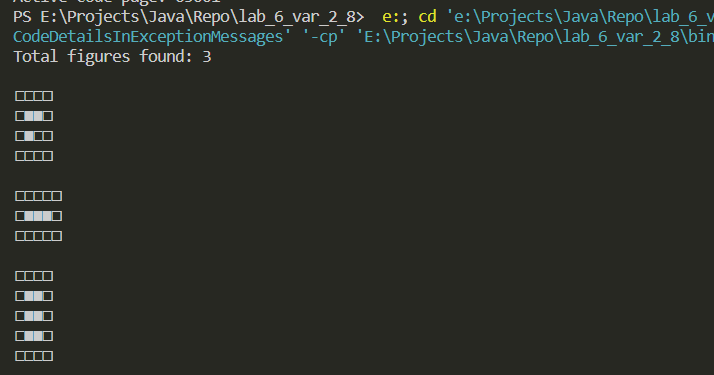


Рисунок 4 – Результат выполнения программы. Найденные фигуры

Листинг 6 – Код основной программы

// Java implementation of the approach

import java.util.\*;

class App

{

static int row = 6;

static int col = 8;

// Function to find the maximum rectangular

// area under given histogram with n bars

static int cal(int hist[], int n)

{

// Create an empty stack. The stack holds indexes

// of hist[] array. The bars stored in stack are

// always in increasing order of their heights.

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

// Initialize max area

int max\_area = 0;

// To store top of the stack

int tp;

// To store area with top bar

int area\_with\_top;

// as the smallest bar

// Run through all bars of given histogram

int i = 0;

while (i < n)

{

// If this bar is higher than the bar on top

// stack, push it to stack

if (s.empty() || hist[s.peek()] <= hist[i])

s.push(i++);

// If this bar is lower than top of stack,

// then calculate area of rectangle with stack

// top as the smallest (or minimum height) bar.

// 'i' is 'right index' for the top and element

// before top in stack is 'left index'

else

{

// Store the top index

tp = s.peek();

// Pop the top

s.pop();

// Calculate the area with hist[tp] stack

// as smallest bar

area\_with\_top = hist[tp] \* (s.empty() ? i :

i - s.peek() - 1);

// Update max area, if needed

if (max\_area < area\_with\_top)

max\_area = area\_with\_top;

}

}

// Now pop the remaining bars from stack and calculate

// area with every popped bar as the smallest bar

while (s.empty() == false)

{

tp = s.peek();

s.pop();

area\_with\_top = hist[tp] \* (s.empty() ? i :

i - s.peek() - 1);

if (max\_area < area\_with\_top)

max\_area = area\_with\_top;

}

return max\_area;

}

// Function to find largest sub matrix

// with all equal elements

static int largestMatrix(int a[][])

{

// To find largest sub matrix

// with all elements 1

int [][]dp = new int[row][col];

// Fill dp[][] by traversing each

// column from bottom to up

for (int i = 0; i < col; i++)

{

int cnt = 0;

for (int j = row - 1; j >= 0; j--)

{

dp[j][i] = 0;

if (a[j][i] == 1)

{

cnt++;

dp[j][i] = cnt;

}

else

{

cnt = 0;

}

}

}

int ans = -1;

for (int i = 0; i < row; i++)

{

// Maintain the histogram array

int []hist = new int[col];

for (int j = 0; j < col; j++)

{

hist[j] = dp[i][j];

}

// Find maximum area rectangle in Histogram

ans = Math.max(ans, cal(hist, col));

}

// To fill dp[][] for finding largest

// sub matrix with all elements 0

for (int i = 0; i < col; i++)

{

int cnt = 0;

for (int j = row - 1; j >= 0; j--)

{

dp[j][i] = 0;

if (a[j][i] == 0)

{

cnt++;

dp[j][i] = cnt;

}

else

{

cnt = 0;

}

}

}

for (int i = 0; i < row; i++)

{

// Maintain the histogram array

int []hist = new int[col];

for (int j = 0; j < col; j++)

{

hist[j] = dp[i][j];

}

// Find maximum area rectangle in Histogram

ans = Math.max(ans, cal(hist, col));

}

return ans;

}

// Driver code

public static void main(String[] args)

{

int a[][] = {{ 1, 1, 0, 1, 0, 0, 0, 0 },

{ 0, 1, 1, 1, 1, 0, 0, 1 },

{ 1, 0, 0, 1, 1, 1, 0, 0 },

{ 0, 1, 1, 0, 1, 1, 0, 0 },

{ 1, 0, 1, 1, 1, 1, 1, 0 },

{ 0, 0, 1, 1, 1, 1, 1, 1 }};

System.out.println(largestMatrix(a));

}

}

Результаты выполнения задания приведем на рисунке далее.

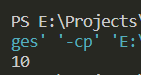


Рисунок 3 – Выполненный код задания 2

## Вывод

По итогам выполнения лабораторной работы были получены навыки программирования с использованием коллекций на языке Java. Были изучены различные варианты их использования.