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**

О Т Ч Е Т

по лабораторной работе № 6

Название: Коллекции

Дисциплина: Языки программирования для работы с большими данными

Студент ИУ6-22М _____ Т.И. Кадыров
(Группа) (Подпись, дата) (И.О. Фамилия)

Преподаватель _____ П.В. Степанов
(Подпись, дата) (И.О. Фамилия)

Москва, 2024

Цель работы: освоить базовые принципы работы с коллекциями на языке Java.

Вариант: 8.

Задание 1: Задана строка, состоящая из символов '(', ')', '[', ']', '{', '}'. Проверить правильность расстановки скобок. Использовать стек.

Код решения приведен в листинге 1.

Листинг 1 — реализация решения

```
import java.util.HashMap;
import java.util.Scanner;

public class Main {
   public static void main(String[] args) {
      boolean isCorrect = true;
      HashMap<String, Integer> bracketMap = new HashMap<>(3);
      bracketMap.put("(", 0);
      bracketMap.put("[", 0);
      bracketMap.put("{", 0);

      Scanner in = new Scanner(System.in);
      String source = in.nextLine();
      for (char ch : source.toCharArray()) {
         switch (ch) {
            case '(':
               bracketMap.put("(", bracketMap.get("(") + 1);
               break;
            case ')':
               bracketMap.put("(", bracketMap.get("(") - 1);
               if (bracketMap.get("(") < 0) {
                  isCorrect = false;
               }
               break;
            case '[':
               bracketMap.put("[", bracketMap.get("[") + 1);
               break;
            case ']':
               bracketMap.put("[", bracketMap.get("[") - 1);
               if (bracketMap.get("[") < 0) {
                  isCorrect = false;
               }
               break;
            case '{':
               bracketMap.put("{", bracketMap.get("{") + 1);
               break;
            case '}':
               bracketMap.put("{", bracketMap.get("{") - 1);
               if (bracketMap.get("{") < 0) {
                  isCorrect = false;
               }
               break;
```

```
        }
      }
      if (isCorrect) {
        System.out.println("String is correct");
        return;
      }
      System.out.println("String is not correct");
    }
}
```

Задание 2: Задан файл с текстом на английском языке. Выделить все различные слова. Слова, отличающиеся только регистром букв, считать одинаковыми. Использовать класс HashSet.

Код решения приведен в листинге 2.

Листинг 2 — реализация решения

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.HashSet;

public class Main {
    public static void main(String[] args) {
        String filePath = "test.txt";

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

        try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {
            String line;
            while ((line = reader.readLine()) != null) {
                String[] words = line.split("\\s+|\\p{Punct}");

                for (String word : words) {
                    if (!word.isEmpty()) {
                        uniqueWords.add(word.toLowerCase());
                    }
                }
            }
        } catch (IOException e) {
            System.err.println("Ошибка чтения файла: " + e.getMessage());
        }

        System.out.println("Уникальные слова в файле:");
        for (String word : uniqueWords) {
            System.out.println(word);
        }
    }
}
```

Задание 3: 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.

Код решения приведен в листинге 3.

Листинг 3 — реализация решения

```
import java.util.HashSet;

public class Main {
  private static final int[][] DIRECTIONS = {{-1, 0}, {1, 0}, {0, -1}, {0, 1}};

  public static void main(String[] args) {
    int[][] grid = {
        {0, 1, 0, 0, 1},
        {1, 1, 0, 1, 1},
        {0, 0, 0, 1, 0},
        {0, 1, 1, 1, 0},
        {0, 1, 1, 1, 0}
    };

    System.out.println("Различные фигуры:");
    HashSet<String> figures = findFigures(grid);
    for (String figure : figures) {
      System.out.println(figure);
    }
  }

  private static HashSet<String> findFigures(int[][] grid) {
    HashSet<String> figures = new HashSet<>();
    int rows = grid.length;
    int cols = grid[0].length;
    boolean[][] visited = new boolean[rows][cols];

    for (int i = 0; i < rows; i++) {
      for (int j = 0; j < cols; j++) {
        if (grid[i][j] == 1 && !visited[i][j]) {
          StringBuilder figure = new StringBuilder();
          dfs(grid, visited, i, j, i, j, figure);
          figures.add(figure.toString());
        }
      }
    }

    return figures;
  }
```

```
    private static void dfs(int[][] grid, boolean[][] visited, int startRow, int startCol, int currentRow, int currentCol, StringBuilder
figure) {
        visited[currentRow][currentCol] = true;
        figure.append("(").append(currentRow).append(",").append(currentCol).append(")");

        for (int[] dir : DIRECTIONS) {
            int newRow = currentRow + dir[0];
            int newCol = currentCol + dir[1];

            if (isValid(grid, newRow, newCol) && grid[newRow][newCol] == 1 && !visited[newRow][newCol]) {
                dfs(grid, visited, startRow, startCol, newRow, newCol, figure);
            }
        }
    }

    private static boolean isValid(int[][] grid, int row, int col) {
        return row >= 0 && row < grid.length && col >= 0 && col < grid[0].length;
    }
}
```

Задание 4: Дана матрица из целых чисел. Найти в ней прямоугольную подматрицу, состоящую из максимального количества одинаковых элементов. Использовать класс Stack.

Код решения приведен в листинге 4.

Листинг 4 — реализация решения

```
import java.util.Stack;

public class Main {
    public static void main(String[] args) {
        int[][] matrix = {
                {1, 0, 1, 0, 0},
                {1, 0, 1, 1, 1},
                {1, 1, 1, 1, 1},
                {1, 0, 0, 1, 0}
        };

        int maxArea = maxRectangle(matrix);
        System.out.println("Максимальная площадь прямоугольной подматрицы: " + maxArea);
    }

    public static int maxRectangle(int[][] matrix) {
        if (matrix == null || matrix.length == 0 || matrix[0].length == 0) {
            return 0;
        }

        int maxArea = 0;
        int[] heights = new int[matrix[0].length];

        for (int[] row : matrix) {
            for (int j = 0; j < matrix[0].length; j++) {
```

```
            heights[j] = (row[j] == 0) ? 0 : heights[j] + 1;
        }
        maxArea = Math.max(maxArea, largestRectangleArea(heights));
    }

    return maxArea;
  }

  public static int largestRectangleArea(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);
        i++;
      } else {
        int top = stack.pop();
        int area = heights[top] * (stack.isEmpty() ? i : i - stack.peek() - 1);
        maxArea = Math.max(maxArea, area);
      }
    }

    while (!stack.isEmpty()) {
      int top = stack.pop();
      int area = heights[top] * (stack.isEmpty() ? i : i - stack.peek() - 1);
      maxArea = Math.max(maxArea, area);
    }

    return maxArea;
  }
}
```

Вывод: в ходе лабораторной работы были освоены базовые принципы работы с коллекциями на языке Java.