



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

Вариант № 17

Название: Коллекции

Дисциплина: Языки программирования для работы с большими данными

Студент

ИУ6-23М

(Группа)

(Подпись, дата)

М.О. Усманов

(И.О. Фамилия)

Преподаватель

(Подпись, дата)

П.В. Степанов

(И.О. Фамилия)

Москва, 2022

Цель работы

Получение первичных навыков работы с коллекциями языка программирования Java.

Ход работы

Задание 1.

– Не используя вспомогательных объектов, переставить отрицательные элементы данного списка в конец, а положительные – в начало этого списка.

– Ввести строки из файла, записать в список ArrayList. Выполнить сортировку строк, используя метод sort() из класса Collections.

Листинг 1 – Код программы первого задания

```
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;

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

        /**
         * Задание 6 из варианта 1
         */
        ArrayList<Integer> list = new ArrayList<>(
            Arrays.asList(
                1, -2, 3, -8, 2, 4, -4, 0
            )
        );

        System.out.println("Before sort: " + list);

        list.sort(Collections.reverseOrder());

        System.out.println("After sort: " + list);

        /**
         * Задание 7 из варианта 1
         */
    }
}
```

```

// Получаем путь файла
Path inputPath = Paths.get("data\\in.txt");

// Читаем все строки файла с кодом программы
List<String> strings = Files.readAllLines(inputPath,
StandardCharsets.UTF_8);
ArrayList<String> outList = new ArrayList<String>(strings);

// Сортируем
outList.sort(Comparator.naturalOrder());

System.out.println("After sort: " + outList);
}
}

```

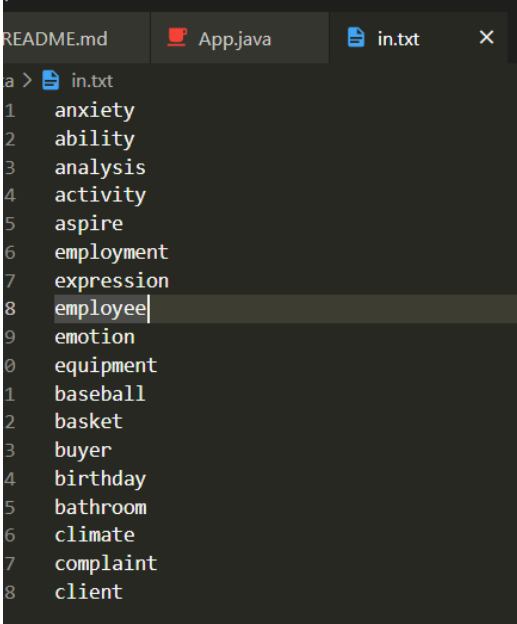
Приведем результаты выполнения данного кода.

```

PS E:\Projects\Java\Repo\lab_6_var_1_6_7> & 'C:\Program Files\Amazon Corretto\jdk17.0.2_8\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'E:\Projects\Java\Repo\lab_6_var_1_6_7\bin' 'App'
Before sort: [1, -2, 3, -8, 2, 4, -4, 0]
After sort: [4, 3, 2, 1, 0, -2, -4, -8]
After sort: [ability, activity, analysis, anxiety, aspire, baseball, basket, bathroom, birthday, buyer, client, climate, complaint, emotion, employee, employment, equipment, expression]
PS E:\Projects\Java\Repo\lab_6_var_1_6_7>

```

Рисунок 1 – Результат выполнения варианта задания 1



```

README.md App.java in.txt
a > in.txt
1 anxiety
2 ability
3 analysis
4 activity
5 aspire
6 employment
7 expression
8 employee
9 emotion
10 equipment
11 baseball
12 basket
13 buyer
14 birthday
15 bathroom
16 climate
17 complaint
18 client

```

Рисунок 2 – Входной файл

Задание 2.

– На плоскости задано N отрезков. Найти точку пересечения двух отрезков, имеющую минимальную абсциссу. Использовать класс TreeMap.

Листинг 2 – Код класса «Point»

```
public class Point
{
    float x;
    float y;

    public Point(int x, int y)
    {
        this.x = x;
        this.y = y;
    }

    public Point(float x, float y)
    {
        this.x = x;
        this.y = y;
    }

    public float getX() {
        return x;
    }

    public float getY() {
        return y;
    }
};
```

Листинг 3 – Код класса «LineSegment»

```
public class LineSegment {

    Point start;
    Point end;

    public LineSegment(Point start, Point end) {
        this.start = start;
        this.end = end;
    }

    public Point getStart() {
        return start;
    }

    public Point getEnd() {
        return end;
    }

    public Point FindIntersection(LineSegment b) {
        // L1 = [x_1 y_1] + t * [(x_2 - x_1) (y_2 - y_1)]
        // L2 = [x_3 y_3] + t * [(x_4 - x_3) (y_4 - y_3)]
        // L1 - это this

        LineSegment a = new LineSegment(start, end);
```

```

        float t = ((a.start.getX() - b.start.getX())*(b.start.getY() -
b.end.getY()) - (a.start.getY() - b.start.getY())*(b.start.getX() -
b.end.getX())) / ((a.start.getX() - a.end.getX())*(b.start.getY() -
b.end.getY()) - (a.start.getY() - a.end.getY())*(b.start.getX() -
b.end.getX()));

        float u = ((a.start.getX() - a.start.getX())*(a.start.getY() -
a.end.getY()) - (a.start.getY() - b.start.getY())*(a.start.getX() -
a.end.getX())) / ((a.start.getX() - a.end.getX())*(b.start.getY() -
b.end.getY()) - (a.start.getY() - a.end.getY())*(b.start.getX() -
b.end.getX()));

        // Если 0 <= t <= 1 и 0 <= u <= 1, то пересечение есть

        if ((t >= 0 && t <= 1.0) && (u >= 0 && u <= 1.0)) {

            Point p = new Point(a.start.getX() + t * (a.end.getX() -
a.start.getX()), a.start.getY() + t * (a.end.getY() - a.start.getY()));

            return p;
        }

        return null;
    }
}

```

Листинг 4 – Код класса основной программы

```

import java.util.TreeMap;

public class App {
    public static void main(String[] args) throws Exception {
        LineSegment a = new LineSegment(
            new Point(1, 1),
            new Point(3, 2)
        );

        LineSegment b = new LineSegment(
            new Point(1, 2),
            new Point(3, 1)
        );

        LineSegment c = new LineSegment(
            new Point(4, 1),
            new Point(5, 3)
        );

        LineSegment d = new LineSegment(
            new Point(4, 5),
            new Point(5, 1)
        );

        TreeMap<Float, Float> tree = new TreeMap<>();

        AddToTree(tree, c.FindIntersection(d));
        AddToTree(tree, a.FindIntersection(b));
    }
}

```

```

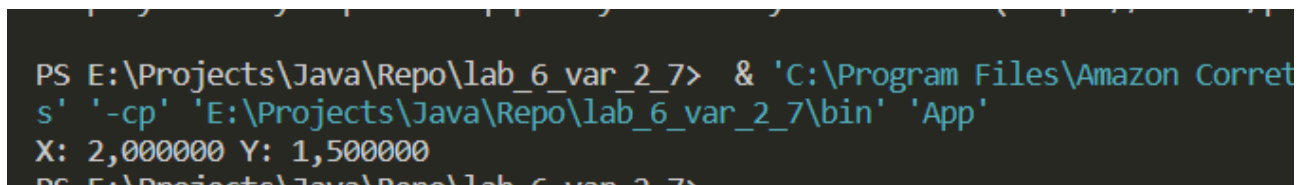
        AddToTree(tree, a.FindIntersection(d));
        AddToTree(tree, b.FindIntersection(c));

        System.out.format("X: %f Y: %f", tree.firstEntry().getKey(),
tree.firstEntry().getValue());
    }

    public static void AddToTree(TreeMap<Float, Float> tree, Point p) {
        if (p != null) {
            tree.put(p.getX(), p.getY());
        }
    }
}

```

Результаты выполнения задания приведем на рисунке далее.



```

PS E:\Projects\Java\Repo\lab_6_var_2_7> & 'C:\Program Files\Amazon Corret
s' '-cp' 'E:\Projects\Java\Repo\lab_6_var_2_7\bin' 'App'
X: 2,000000 Y: 1,500000
PS E:\Projects\Java\Repo\lab_6_var_2_7>

```

Рисунок 3 – Выполненный код п. 1 задания 2

Подзадача 2.

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.

Вводные данные

```

□□□□□□□□
□■□■□■□■
□■□□■□■□
□□□□■□■□
□□□□□□□□
□■□□■□■□
□■□■□■□■
□□□□□□□□

```

Рисунок 4 – Матрица клеток, подаваемая на ввод

Листинг 5 – 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 6 – Код класса «BoundaryBox»

```
public class BoundaryBox {
    Point2D min;
    Point2D max;

    public BoundaryBox(Point2D min, Point2D max) {
        this.min = min;
        this.max = max;
    }
}
```

Листинг 7 – 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 8 – 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}, // 00000000
            {0, 1, 1, 0, 1, 0, 1, 0}, // 01101010
            {0, 1, 0, 0, 1, 0, 1, 0}, // 01001010
            {0, 0, 0, 0, 1, 0, 1, 0}, // 00001010
            {0, 0, 0, 0, 0, 0, 0, 0}, // 00000000
            {0, 1, 0, 0, 1, 1, 1, 0}, // 01001110
            {0, 1, 1, 0, 1, 1, 1, 0}, // 01101110
            {0, 0, 0, 0, 0, 0, 0, 0} // 00000000
        };

        // 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 наткнемся на закрашенную клетку, начинаем Flood
                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.

```

PS E:\Projects\Java\Repo\lab_6_var_2_8> e;; cd 'e:\Projects\Java\Repo\lab_6_v
CodeDetailsInExceptionMessages' '-cp' 'E:\Projects\Java\Repo\lab_6_var_2_8\bin
Total figures found: 3

  □□□□
  □■□□
  □□□□
  □□□□

  □□□□
  □□□■
  □□□□
  □□□□

  □□□□
  □□■□
  □□■□
  □□■□
  □□□□

```

Рисунок 5 – Результат выполнения программы. Найденные фигуры

Местоположение репозитория с файлами проекта

Файлы проекта расположены в репозитории веб-платформы для совместной разработки Github. Местоположение в репозитории:

https://github.com/s314/big-data-studies/tree/main/lab_6_var_1_6_7

https://github.com/s314/big-data-studies/tree/main/lab_6_var_2_7

https://github.com/s314/big-data-studies/tree/main/lab_6_var_2_8

Вывод

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