



**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 09.04.01/12 Интеллектуальный анализ больших  
данных в системах поддержки принятия решений.**

**О Т Ч Е Т**

**по лабораторной работе № 6**

**Вариант № 5**

**Название:** Коллекции

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

Студент

ИУ6-23М

(Группа)

\_\_\_\_\_  
(Подпись, дата)

А.О.Крейденко

(И.О. Фамилия)

Преподаватель

\_\_\_\_\_  
(Подпись, дата)

П.В. Степанов

(И.О. Фамилия)

Москва, 2024

**Цель:** изучить работу с коллекциями в java.

**Задание 1:** умножить два многочлена заданной степени, если коэффициенты многочленов хранятся в различных списках.

Код класса PolynomialMultiplication:

```
import java.util.ArrayList;
import java.util.List;

public class PolynomialMultiplication {

    // Метод для умножения двух многочленов
    public static List<Integer>
multiplyPolynomials(List<Integer> poly1, List<Integer> poly2) {
        int degree1 = poly1.size() - 1;
        int degree2 = poly2.size() - 1;
        int resultDegree = degree1 + degree2;

        // Создаем список для хранения коэффициентов
        // результирующего многочлена
        List<Integer> result = new ArrayList<>(resultDegree +
1);
        for (int i = 0; i <= resultDegree; i++) {
            result.add(0);
        }

        // Умножаем многочлены
        for (int i = 0; i <= degree1; i++) {
            for (int j = 0; j <= degree2; j++) {
                int coefficient = result.get(i + j) +
poly1.get(i) * poly2.get(j);
                result.set(i + j, coefficient);
            }
        }

        return result;
    }

    public static void main(String[] args) {
        // Пример многочленов
        List<Integer> poly1 = List.of(1, 2, 3); // 1 + 2x + 3x^2
        List<Integer> poly2 = List.of(4, 5);    // 4 + 5x

        // Умножение многочленов
        List<Integer> result = multiplyPolynomials(poly1,
poly2);

        // Вывод результата
        System.out.println("Результат умножения многочленов:");
        for (int i = 0; i < result.size(); i++) {
            System.out.print(result.get(i));
            if (i > 0) {
```

```

        System.out.print("x^" + i);
    }
    if (i < result.size() - 1) {
        System.out.print(" + ");
    }
}
}
}

```

Работа программы показана на рисунке 1.

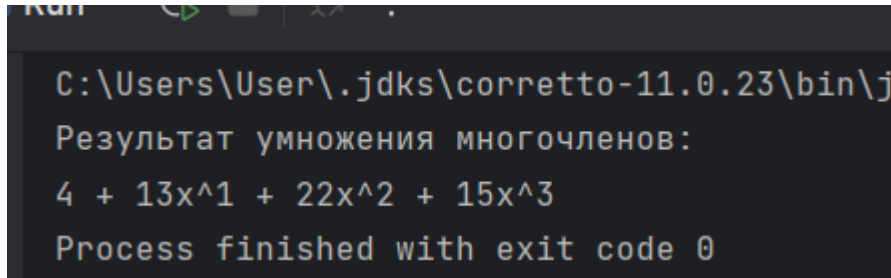


Рисунок 1 – Работа программы 1

**Задание 2:** не используя вспомогательных объектов, переставить отрицательные элементы данного списка в конец, а положительные – в начало этого списка.

Код класса RearrangeList:

```

import java.util.List;
import java.util.ArrayList;
import java.util.Collections;

public class RearrangeList {

    public static void rearrange(List<Integer> list) {
        int n = list.size();
        int left = 0;
        int right = n - 1;

        while (left < right) {
            // Найти первый отрицательный элемент слева
            while (left < right && list.get(left) >= 0) {
                left++;
            }

            // Найти первый положительный элемент справа
            while (left < right && list.get(right) < 0) {
                right--;
            }

            // Поменять местами отрицательный и положительный
элементы
            if (left < right) {

```

```

        Collections.swap(list, left, right);
        left++;
        right--;
    }
}

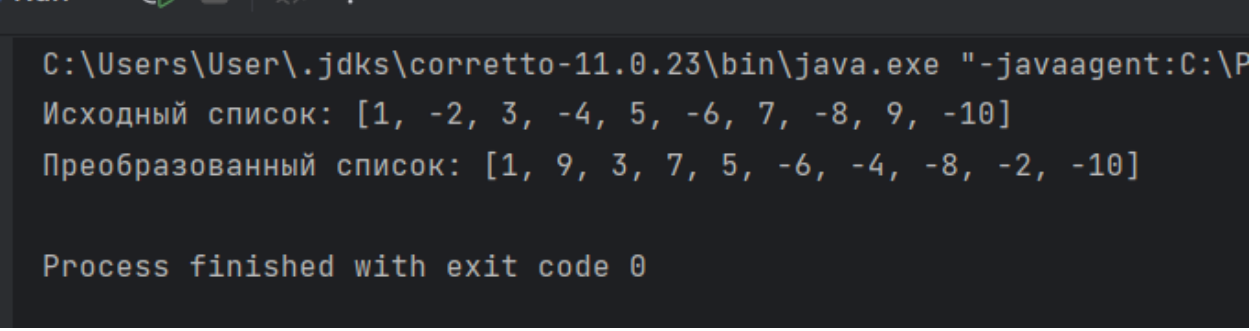
public static void main(String[] args) {
    List<Integer> list = new ArrayList<>(List.of(1, -2, 3, -4, 5, -6, 7, -8, 9, -10));
    System.out.println("Исходный список: " + list);

    rearrange(list);

    System.out.println("Преобразованный список: " + list);
}
}

```

Работа программы показана на рисунке 2.



```

C:\Users\User\.jdk\corretto-11.0.23\bin\java.exe -javaagent:C:\P
Исходный список: [1, -2, 3, -4, 5, -6, 7, -8, 9, -10]
Преобразованный список: [1, 9, 3, 7, 5, -6, -4, -8, -2, -10]

Process finished with exit code 0

```

Рисунок 2 – Работа программы 2

**Задание 3:** во входном файле расположены два набора положительных чисел; между наборами стоит отрицательное число. Построить два списка C1 и C2, элементы которых содержат соответственно числа 1-го и 2-го набора таким образом, чтобы внутри одного списка числа были упорядочены по возрастанию. Затем объединить списки C1 и C2 в один упорядоченный список, изменяя только значения полей ссылочного типа.

Код класса Main:

```

import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

public class Main {

```

```

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

    try {
        // Чтение всех строк из файла
        List<String> lines =
Files.readAllLines(Paths.get(filePath));
        if (lines.isEmpty()) {
            System.out.println("Файл пуст.");
            return;
        }

        // Парсинг чисел из первой строки файла
        String[] numbers = lines.get(0).split("\\s+");
        List<Integer> C1 = new ArrayList<>();
        List<Integer> C2 = new ArrayList<>();

        boolean firstSet = true;
        for (String numStr : numbers) {
            int num = Integer.parseInt(numStr);
            if (num < 0) {
                firstSet = false; // Встретили отрицательное
число, переключаемся на второй набор
                continue;
            }
            if (firstSet) {
                C1.add(num);
            } else {
                C2.add(num);
            }
        }

        // Сортировка списков C1 и C2
        Collections.sort(C1);
        Collections.sort(C2);

        // Объединение списков C1 и C2 в один упорядоченный
список
        List<Integer> mergedList = new ArrayList<>(C1);
        mergedList.addAll(C2);
        Collections.sort(mergedList);

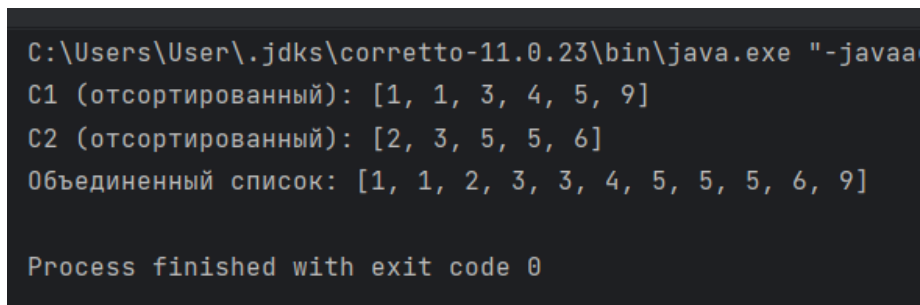
        // Вывод результата
        System.out.println("C1 (отсортированный): " + C1);
        System.out.println("C2 (отсортированный): " + C2);
        System.out.println("Объединенный список: " +
mergedList);

        } catch (IOException e) {
            System.err.println("Ошибка чтения файла: " +
e.getMessage());
        }
    }
}

```

```
}  
}
```

Работа программы показана на рисунке 3. Содержимое входного файла:  
«3 1 4 1 5 9 -1 2 6 5 3 5»



```
C:\Users\User\jdk\corretto-11.0.23\bin\java.exe "-javaa  
C1 (отсортированный): [1, 1, 3, 4, 5, 9]  
C2 (отсортированный): [2, 3, 5, 5, 6]  
Объединенный список: [1, 1, 2, 3, 3, 4, 5, 5, 5, 6, 9]  
  
Process finished with exit code 0
```

Рисунок 3 – Работа программы 3

**Задание 4:** на плоскости задано N точек. Вывести в файл описания всех прямых, которые проходят более чем через одну точку из заданных. Для каждой прямой указать, через сколько точек она проходит. Использовать класс HashMap.

Код класса Main:

```
import java.io.BufferedWriter;  
import java.io.FileWriter;  
import java.io.IOException;  
import java.util.*;  
  
public class Main {  
  
    public static void main(String[] args) {  
        // Пример набора точек  
        List<Point> points = List.of(  
            new Point(1, 1),  
            new Point(2, 2),  
            new Point(3, 3),  
            new Point(4, 4),  
            new Point(1, 2),  
            new Point(2, 4),  
            new Point(3, 6)  
        );  
  
        // Вызываем метод для нахождения всех прямых  
        findLines(points, "output.txt");  
    }  
  
    public static void findLines(List<Point> points, String  
outputFilePath) {  
        // Карта для хранения описания прямых и количества точек  
на каждой прямой
```

```

        HashMap<Line, Integer> linesMap = new HashMap<>();

        // Находим все прямые, проходящие через более чем одну
точку
        for (int i = 0; i < points.size(); i++) {
            for (int j = i + 1; j < points.size(); j++) {
                Line line = new Line(points.get(i),
points.get(j));
                linesMap.put(line, linesMap.getOrDefault(line,
0) + 1);
            }
        }

        // Фильтрация прямых, проходящих через более чем одну
точку
        linesMap.entrySet().removeIf(entry -> entry.getValue()
<= 1);

        // Запись результатов в файл
        try (BufferedWriter writer = new BufferedWriter(new
FileWriter(outputFilePath))) {
            for (Map.Entry<Line, Integer> entry :
linesMap.entrySet()) {
                Line line = entry.getKey();
                int count = entry.getValue();
                writer.write(String.format("Прямая: %s,
Количество точек: %d\n", line, count));
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

```

### Код класса Line:

```

import java.util.Objects;

// Класс для представления прямой в виде  $y = mx + b$ 
public class Line {
    double slope, intercept;

    Line(Point p1, Point p2) {
        if (p1.x == p2.x) { // вертикальная прямая
            this.slope = Double.POSITIVE_INFINITY;
            this.intercept = p1.x;
        } else {
            this.slope = (double) (p2.y - p1.y) / (p2.x - p1.x);
            this.intercept = p1.y - this.slope * p1.x;
        }
    }
}

```

```

@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    Line line = (Line) o;
    return Double.compare(line.slope, slope) == 0 &&
        Double.compare(line.intercept, intercept) == 0;
}

@Override
public int hashCode() {
    return Objects.hash(slope, intercept);
}

@Override
public String toString() {
    return slope == Double.POSITIVE_INFINITY ?
String.format("x = %.2f", intercept)
        : String.format("y = %.2fx + %.2f", slope,
intercept);
}
}

```

**Код класса Point:**

```

// Класс для представления точки
public class Point {
    int x, y;

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

```

Работа программы показана на рисунке 4.

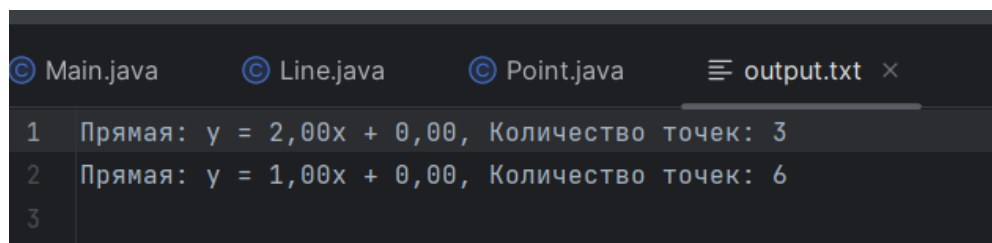


Рисунок 4 – Работа программы 4

**Вывод:** были изучена работа с коллекциями в java.