



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 № 5**

**Вариант № 7**

**Название:** исключения, файлы

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

Студент

ИУ6-23М

(Группа)

\_\_\_\_\_  
(Подпись, дата)

Ф.А. Лучкин

(И.О. Фамилия)

Преподаватель

\_\_\_\_\_  
(Подпись, дата)

П.В. Степанов

(И.О. Фамилия)

Москва, 2024

**Цель:** освоить принципы работы с классами, наследованием и полиморфизмом на языке программирования Java.

**Задание 1:** определить класс Дробь в виде пары (m,n). Класс должен содержать несколько конструкторов. Реализовать методы для сложения, вычитания, умножения и деления дробей. Объявить массив из k дробей, ввести/вывести значения для массива дробей. Создать массив объектов и передать его в метод, который изменяет каждый элемент массива с четным индексом путем добавления следующего за ним элемента массива. Сделать это, контролируя состояние потоков ввода/вывода. При возникновении ошибок, связанных с корректностью выполнения математических операций, генерировать и обрабатывать исключительные ситуации. Предусмотреть обработку исключений, возникающих при нехватке памяти, отсутствии требуемой записи (объекта) в файле, недопустимом значении поля и т.д.

Код класса Fraction:

```
package lab5_var1_7;

import static lab1_var2_7.Main.getGreatestCommonDivisor;

public class Fraction {
    private int m;
    private int n;

    public Fraction() {
        this.setM(1);
        this.setN(1);
    }

    public Fraction(int m, int n) {
        this.setM(m);
        this.setN(n);
    }

    public int getM() {
        return this.m;
    }

    public void setM(int m) {
        this.m = m;
    }

    public int getN() {
        return this.n;
    }

    public void setN(int n) {
        this.n = n;
    }

    public void print() {
        System.out.printf("%d/%d ", m, n);
    }
}
```

```

    }

    public static Fraction add(Fraction first, Fraction second) {
        try {
            int denominator = first.getN() * second.getN();
            int numerator = first.getM() * second.getN() + first.getN() *
second.getM();
            return signFix(numerator, denominator);
        } catch (ArithmeticException e) {
            System.out.println("Ошибка при выполнении операции сложения: " +
e.getMessage());
            return null;
        }
    }

    public static Fraction subtract(Fraction first, Fraction second) {
        try {
            int denominator = first.getN() * second.getN();
            int numerator = first.getM() * second.getN() - first.getN() *
second.getM();
            return signFix(numerator, denominator);
        } catch (ArithmeticException e) {
            System.out.println("Ошибка при выполнении операции вычитания: " +
e.getMessage());
            return null;
        }
    }

    public static Fraction multiply(Fraction first, Fraction second) {
        try {
            int numerator = first.getM() * second.getM();
            int denominator = first.getN() * second.getN();
            return signFix(numerator, denominator);
        } catch (ArithmeticException e) {
            System.out.println("Ошибка при выполнении операции умножения: " +
e.getMessage());
            return null;
        }
    }

    public static Fraction divide(Fraction first, Fraction second) {
        try {
            int numerator = first.getM() * second.getN();
            int denominator = first.getN() * second.getM();
            if (denominator == 0) {
                throw new ArithmeticException("Деление на ноль недопустимо");
            }
            return signFix(numerator, denominator);
        } catch (ArithmeticException e) {
            System.out.println("Ошибка при выполнении операции деления: " +
e.getMessage());
            return null;
        }
    }

    private static Fraction signFix(int numerator, int denominator) {
        try {
            int commonDivisor = getGreatestCommonDivisor(numerator, denominator);
            numerator = numerator / commonDivisor;
            denominator = denominator / commonDivisor;
            if (denominator < 0) {
                denominator = denominator * -1;
                numerator = numerator * -1;
            }
            return new Fraction(numerator, denominator);
        }
    }

```

```

    } catch (ArithmeticException e) {
        System.out.println("Ошибка при исправлении знака: " + e.getMessage());
        return null;
    }
}
}

```

### Код класса Main:

```

package lab3_var1_7;

import java.util.Random;

public class Main {
    public static void main(String[] args) {
        Random random = new Random();

        /*Fraction first = new Fraction();
        Fraction second = new Fraction(1, 2);
        operationsDemonstration(first, second);

        first.setN(4);
        second.setM(3);
        second.setN(8);
        operationsDemonstration(first, second);

        first.setM(5);
        first.setN(6);
        second.setM(6);
        second.setN(7);
        operationsDemonstration(first, second);
        System.out.println("TASK:");*/

        int arrSize = random.nextInt(9) + 2;
        Fraction[] fractions = new Fraction[arrSize];

        System.out.println("Source array:");
        for (int i = 0; i < arrSize; i++) {
            int n = random.nextInt(8) + 2;
            int m = random.nextInt(n) + 1;
            fractions[i] = new Fraction(m, n);
            fractions[i].print();
        }

        System.out.println("\nResult array:");
        for (Fraction fraction : taskMethod(fractions)) {
            fraction.print();
        }
    }

    private static Fraction[] taskMethod(Fraction[] fractions) {
        for (int i = 0; i < fractions.length; i++) {
            if ((i % 2 == 0) && (i < fractions.length - 1)) {
                fractions[i] = Fraction.add(fractions[i], fractions[i+1]);
            }
        }
        return fractions;
    }

    private static void operationsDemonstration(Fraction first, Fraction second) {
        Fraction fractionSum = Fraction.add(first, second);
        Fraction fractionDiff = Fraction.subtract(first, second);
    }
}

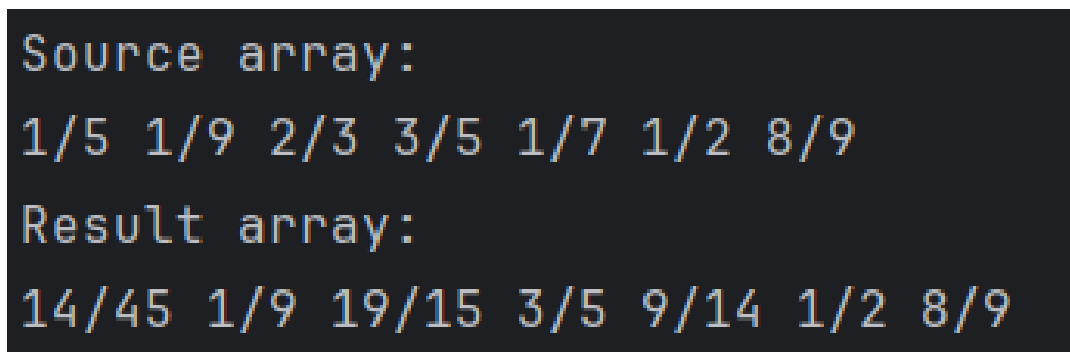
```

```

        Fraction fractionMultiple = Fraction.multiply(first, second);
        Fraction fractionQuotient = Fraction.divide(first, second);
        System.out.printf("First fraction: %d/%d\n", first.getM(), first.getN());
        System.out.printf("Second fraction: %d/%d\n", second.getM(),
second.getN());
        System.out.printf("Sum fraction: %d/%d\n", fractionSum.getM(),
fractionSum.getN());
        System.out.printf("Diff fraction: %d/%d\n", fractionDiff.getM(),
fractionDiff.getN());
        System.out.printf("Multiple fraction: %d/%d\n",
            fractionMultiple.getM(), fractionMultiple.getN());
        System.out.printf("Quotient of fractions: %d/%d\n-----\n",
            fractionQuotient.getM(), fractionQuotient.getN());
    }
}

```

Работа программы показана на рисунке 1.



```

Source array:
1/5 1/9 2/3 3/5 1/7 1/2 8/9
Result array:
14/45 1/9 19/15 3/5 9/14 1/2 8/9

```

Рисунок 1 – Работа программы

**Задание 2:** определить класс Комплекс. Класс должен содержать несколько конструкторов. Реализовать методы для сложения, вычитания, умножения, деления, присваивания комплексных чисел. Создать два вектора размерности n из комплексных координат. Передать их в метод, который выполнит их сложение. Сделать это, контролируя состояние потоков ввода/вывода. При возникновении ошибок, связанных с корректностью выполнения математических операций, генерировать и обрабатывать исключительные ситуации. Предусмотреть обработку исключений, возникающих при нехватке памяти, отсутствии требуемой записи (объекта) в файле, недопустимом значении поля и т.д.

Код класса ComplexNumber:

```

package lab5_var1_8;

public class ComplexNumber {
    private double re;
    private double im;

    public ComplexNumber() {
        this.setRe(1);
        this.setIm(0);
    }
}

```

```

public ComplexNumber(double re, double im) {
    this.setRe(re);
    this.setIm(im);
}

public double getRe() {
    return this.re;
}

public void setRe(double re) {
    this.re = re;
}

public double getIm() {
    return this.im;
}

public void setIm(double im) {
    this.im = im;
}

public void setComplexNumber(double re, double im) {
    this.re = re;
    this.im = im;
}

public void print() {
    System.out.printf("%.3f ; %.3f", re, im);
}

public static ComplexNumber add(ComplexNumber first, ComplexNumber second) {
    return new ComplexNumber(
        first.getRe() + second.getRe(),
        first.getIm() + second.getIm()
    );
}

public static ComplexNumber subtract(ComplexNumber first, ComplexNumber
second) {
    return new ComplexNumber(
        first.getRe() - second.getRe(),
        first.getIm() - second.getIm()
    );
}

public static ComplexNumber multiply(ComplexNumber first, ComplexNumber
second) {
    try {
        double operationRe = first.getRe() * second.getRe() - first.getIm() *
second.getIm();
        double operationIm = first.getRe() * second.getIm() + first.getIm() *
second.getRe();
        return new ComplexNumber(operationRe, operationIm);
    } catch (Exception e) {
        System.out.println("Error during multiplication: " + e.getMessage());
        return null;
    }
}

public static ComplexNumber divide(ComplexNumber first, ComplexNumber second)
{
    try {
        double divisor = second.getRe() * second.getRe() + second.getIm() *
second.getIm();

```

```

        if (divisor == 0) {
            throw new IllegalArgumentException("Division by zero is not
allowed");
        }

        double operationRe = (first.getRe() * second.getRe() + first.getIm() *
second.getIm()) / divisor;
        double operationIm = (first.getIm() * second.getRe() - first.getRe() *
second.getIm()) / divisor;

        return new ComplexNumber(operationRe, operationIm);
    } catch (IllegalArgumentException e) {
        System.out.println("Error during division: " + e.getMessage());
        return null;
    }
}
}

```

### Код класса Main:

```

package lab3_var1_8;

import java.util.Random;

public class Main {
    public static void main(String[] args) {
        Random random = new Random();

        /*ComplexNumber first = new ComplexNumber();
        ComplexNumber second = new ComplexNumber(1, 2);
        operationsDemonstration(first, second);

        first.setIm(4);
        second.setRe(3);
        second.setIm(8);
        operationsDemonstration(first, second);

        first.setRe(5);
        first.setIm(6);
        second.setRe(6);
        second.setIm(7);
        operationsDemonstration(first, second);*/
        //System.out.println("TASK:");

        int n = random.nextInt(9) + 2;
        ComplexNumber[] firstVector = new ComplexNumber[n];
        ComplexNumber[] secondVector = new ComplexNumber[n];

        System.out.println("Source vectors:");
        for (int i = 0; i < n; i++) {
            int re = random.nextInt(10) - random.nextInt(10);
            int im = random.nextInt(10) - random.nextInt(10);
            firstVector[i] = new ComplexNumber(re, im);

            re = random.nextInt(10) - random.nextInt(10);
            im = random.nextInt(10) - random.nextInt(10);
            secondVector[i] = new ComplexNumber(re, im);
        }
        for (ComplexNumber numbers : firstVector) {
            numbers.print();
            System.out.print(' ');
        }
        System.out.print('\n');
    }
}

```

```

        for (ComplexNumber numbers : secondVector) {
            numbers.print();
            System.out.print(' ');
        }

        System.out.println("\nResult vector:");
        for (ComplexNumber numbers : taskMethod(firstVector, secondVector)) {
            numbers.print();
            System.out.print(' ');
        }
    }

    private static ComplexNumber[] taskMethod(ComplexNumber[] firstVector,
ComplexNumber[] secondVector) {
        ComplexNumber[] resultVector = new ComplexNumber[firstVector.length];
        for (int i = 0; i < firstVector.length; i++) {
            resultVector[i] = ComplexNumber.add(firstVector[i], secondVector[i]);
        }
        return resultVector;
    }

    private static void operationsDemonstration(ComplexNumber first, ComplexNumber
second) {
        ComplexNumber fractionSum = ComplexNumber.add(first, second);
        ComplexNumber fractionDiff = ComplexNumber.subtract(first, second);
        ComplexNumber fractionMultiple = ComplexNumber.multiply(first, second);
        ComplexNumber fractionQuotient = ComplexNumber.divide(first, second);
        System.out.printf("First number: (%.3f ; %.3f)\n", first.getRe(),
first.getIm());
        System.out.printf("Second number: (%.3f ; %.3f)\n", second.getRe(),
second.getIm());
        System.out.printf("Sum number: (%.3f ; %.3f)\n", fractionSum.getRe(),
fractionSum.getIm());
        System.out.printf("Diff number: (%.3f ; %.3f)\n", fractionDiff.getRe(),
fractionDiff.getIm());
        System.out.printf("Multiple number: (%.3f ; %.3f)\n",
            fractionMultiple.getRe(), fractionMultiple.getIm());
        System.out.printf("Quotient of numbers: (%.3f ; %.3f)\n-----
-\n",
            fractionQuotient.getRe(), fractionQuotient.getIm());
    }
}

```

Работа программы показана на рисунке 2.

```

Source vectors:
(-4,000 ; 2,000) (-2,000 ; -1,000) (2,000 ; 5,000) (2,000 ; -8,000) (-4,000 ; -3,000)
(7,000 ; -3,000) (-5,000 ; -2,000) (2,000 ; 0,000) (2,000 ; -4,000) (-8,000 ; -1,000)
Result vector:
(3,000 ; -1,000) (-7,000 ; -3,000) (4,000 ; 5,000) (4,000 ; -12,000) (-12,000 ; -4,000)

```

Рисунок 2 – Работа программы

**Задание 3:** создать классы, спецификации которых приведены ниже. Определить конструкторы и методы setТип(), getТип(), toString(). Определить дополнительно методы в классе, создающем массив объектов. Задать критерий выбора данных и вывести эти данные на консоль. Phone: id, Фамилия, Имя, Отчество, Адрес, Номер кредитной карточки, Дебет, Кредит, Время городских и



междугородных разговоров. Создать массив объектов. Вывести: а) сведения об абонентах, у которых время внутригородских разговоров превышает заданное; б) сведения об абонентах, которые пользовались междугородной связью; с) сведения об абонентах в алфавитном порядке. Сделать это, реализовав собственные обработчики исключений и исключения ввода/вывода.

Код класса InvalidInputException:

```
package lab5_var2_7;

public class InvalidInputException extends Exception {
    public InvalidInputException(String message) {
        super(message);
    }
}
```

Код класса Phone:

```
package lab5_var2_7;

public class Phone implements Comparable<Phone> {
    private int id;
    private String lastname;
    private String firstname;
    private String surname;
    private String address;
    private int creditCard;
    private double debit;
    private double credit;

    private int cityCallsMinutes;
    private int longDistanceCallsMinutes;

    public Phone(int id, String lastname, String firstname, String surname,
        String address, int creditCard, double debit, double credit,
        int cityCallsMinutes, int longDistanceCallsMinutes) {
        try {
            this.setId(id);
            this.setLastname(lastname);
            this.setFirstname(firstname);
            this.setSurname(surname);
            this.setAddress(address);
            this.setCreditCard(creditCard);
            this.setDebit(debit);
            this.setCredit(credit);
            this.setCityCallsMinutes(cityCallsMinutes);
            this.setLongDistanceCallsMinutes(longDistanceCallsMinutes);
        } catch (InvalidInputException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }

    @Override
    public String toString() {
        return String.format("
        Phone = {
            \tid: %d; lastname: %s; firstname: %s; surname: %s;
            \taddress: %s; creditCard: %d;
            \tdebit: %.3f; credit: %.3f;
            \tcityCallsMinutes: %d; longDistanceCallsMinutes: %d
        }",
            id, lastname, firstname, surname, address, creditCard, debit,
            credit, cityCallsMinutes, longDistanceCallsMinutes);
    }
}
```

```

credit, cityCallsMinutes, longDistanceCallsMinutes);
    }

    @Override
    public int compareTo(Phone p) {
        try {
            int result = this.getLastname().compareTo(p.getLastname());
            if (result != 0) {
                return result;
            } else {
                result = this.getFirstname().compareTo(p.getFirstname());
            }
            if (result != 0) {
                return result;
            } else {
                return this.getSurname().compareTo(p.getSurname());
            }
        } catch (NullPointerException e) {
            System.out.println("Error: " + e.getMessage());
            return 0;
        }
    }

    public int getId() {
        return id;
    }

    public void setId(int id) throws InvalidInputException {
        if (id <= 0) {
            throw new InvalidInputException("ID must be a positive integer.");
        }
        this.id = id;
    }

    public String getLastname() {
        return lastname;
    }

    public void setLastname(String lastname) throws InvalidInputException {
        if (lastname == null || lastname.isEmpty()) {
            throw new InvalidInputException("Lastname cannot be null or empty.");
        }
        this.lastname = lastname;
    }

    public String getFirstname() {
        return firstname;
    }

    public void setFirstname(String firstname) throws InvalidInputException {
        if (firstname == null || firstname.isEmpty()) {
            throw new InvalidInputException("Firstname cannot be null or empty.");
        }
        this.firstname = firstname;
    }

    public String getSurname() {
        return surname;
    }

    public void setSurname(String surname) throws InvalidInputException {
        if (surname == null || surname.isEmpty()) {
            throw new InvalidInputException("Surname cannot be null or empty.");
        }
        this.surname = surname;
    }

```

```

    }

    public String getAddress() {
        return address;
    }

    public void setAddress(String address) throws InvalidInputException {
        if (address == null || address.isEmpty()) {
            throw new InvalidInputException("Address cannot be null or empty.");
        }
        this.address = address;
    }

    public int getCreditCard() {
        return creditCard;
    }

    public void setCreditCard(int creditCard) throws InvalidInputException {
        if (creditCard <= 0) {
            throw new InvalidInputException("Credit card number must be a positive integer.");
        }
        this.creditCard = creditCard;
    }

    public double getDebit() {
        return debit;
    }

    public void setDebit(double debit) throws InvalidInputException {
        if (debit < 0) {
            throw new InvalidInputException("Debit amount cannot be negative.");
        }
        this.debit = debit;
    }

    public double getCredit() {
        return credit;
    }

    public void setCredit(double credit) throws InvalidInputException {
        if (credit < 0) {
            throw new InvalidInputException("Credit amount cannot be negative.");
        }
        this.credit = credit;
    }

    public int getCityCallsMinutes() {
        return cityCallsMinutes;
    }

    public void setCityCallsMinutes(int cityCallsMinutes) throws InvalidInputException {
        if (cityCallsMinutes < 0) {
            throw new InvalidInputException("City calls minutes cannot be negative.");
        }
        this.cityCallsMinutes = cityCallsMinutes;
    }

    public int getLongDistanceCallsMinutes() {
        return longDistanceCallsMinutes;
    }

    public void setLongDistanceCallsMinutes(int longDistanceCallsMinutes) throws

```

```

InvalidInputException {
    if (longDistanceCallsMinutes < 0) {
        throw new InvalidInputException("Long distance calls minutes cannot be
negative.");
    }
    this.longDistanceCallsMinutes = longDistanceCallsMinutes;
}
}

```

### Код класса Main:

```

package lab3_var2_7;

import java.util.Arrays;
import java.util.Random;
import java.util.Scanner;
import static java.lang.System.*;

public class Main {
    public static void main(String[] args) {
        Random random = new Random();
        Scanner scanner = new Scanner(in);

        int n = random.nextInt(5) + 2;
        out.printf("Input info about %d users\n", n);
        Phone[] users = new Phone[n];

        for (int i = 0; i < n; i++) {
            int id = i + 1;
            out.printf("Input lastname for user №%d:\n", id);
            String lastname = scanner.nextLine();
            out.printf("Input firstname for user №%d:\n", id);
            String firstname = scanner.nextLine();
            out.printf("Input surname for user №%d:\n", id);
            String surname = scanner.nextLine();
            out.printf("Input address for user №%d:\n", id);
            String address = scanner.nextLine();

            int creditCard = id * 1000;
            int debit = random.nextInt(1000000);
            int credit = random.nextInt(1000000);
            int cityCallsMinutes = random.nextInt(30);
            int longDistanceCallsMinutes = random.nextInt(30) * (i % 3);

            users[i] = new Phone(id, lastname, firstname, surname,
                                address, creditCard, debit, credit,
                                cityCallsMinutes, longDistanceCallsMinutes);
        }
        out.println("Source array:");
        printAll(users);
        out.println("-----\nSet limit for city calls:");
        int limit = scanner.nextInt();
        out.println("-----\nExceeded the time limit:");
        exceedingCityCallsTime(limit, users);
        out.println("-----\nLong distance calls usage:");
        longDistanceCallsUsage(users);
        out.println("-----\nSorted by full name:");
        Arrays.sort(users);
        printAll(users);
    }

    private static void printAll(Phone[] users) {
        for (Phone user : users) {

```

```

        out.println(user);
    }
}
private static void exceedingCityCallsTime(int limit, Phone[] users) {
    int flag = 0;
    for (Phone user : users) {
        if (user.getCityCallsMinutes() > limit) {
            out.println(user);
            flag = 1;
        }
    }
    if (flag == 0) out.println("No one");
}

private static void longDistanceCallsUsage(Phone[] users) {
    int flag = 0;
    for (Phone user : users) {
        if (user.getLongDistanceCallsMinutes() > 0) {
            out.println(user);
            flag = 1;
        }
    }
    if (flag == 0) out.println("No one");
}
}

```

Работа программы показана на рисунках 3-4.

```

Input info about 2 users
Input lastname for user №1:
Luchkin
Input firstname for user №1:
Fedor
Input surname for user №1:
Antonovich
Input address for user №1:
Borovaya 8
Input lastname for user №2:
Kochikina
Input firstname for user №2:
Lada
Input surname for user №2:
Viacheslavovna
Input address for user №2:
Borovaya 8
Source array:
Phone = {
    id: 1; lastname: Luchkin; firstname: Fedor; surname: Antonovich;
    address: Borovaya 8; creditCard: 1000;
    debit: 751494,000; credit: 309032,000;
    cityCallsMinutes: 22; longDistanceCallsMinutes: 0
}
Phone = {
    id: 2; lastname: Kochikina; firstname: Lada; surname: Viacheslavovna;
    address: Borovaya 8; creditCard: 2000;
    debit: 17262,000; credit: 899737,000;
    cityCallsMinutes: 4; longDistanceCallsMinutes: 23
}
-----

```

Рисунок 3 – Работа программы

```

Set limit for city calls:
10
-----
Exceeded the time limit:
Phone = {
    id: 1; lastname: Luchkin; firstname: Fedor; surname: Antonovich;
    address: Borovaya 8; creditCard: 1000;
    debit: 751494,000; credit: 309032,000;
    cityCallsMinutes: 22; longDistanceCallsMinutes: 0
}
-----
Long distance calls usage:
Phone = {
    id: 2; lastname: Kochikina; firstname: Lada; surname: Viacheslavovna;
    address: Borovaya 8; creditCard: 2000;
    debit: 17262,000; credit: 899737,000;
    cityCallsMinutes: 4; longDistanceCallsMinutes: 23
}
-----
Sorted by full name:
Phone = {
    id: 2; lastname: Kochikina; firstname: Lada; surname: Viacheslavovna;
    address: Borovaya 8; creditCard: 2000;
    debit: 17262,000; credit: 899737,000;
    cityCallsMinutes: 4; longDistanceCallsMinutes: 23
}
Phone = {
    id: 1; lastname: Luchkin; firstname: Fedor; surname: Antonovich;
    address: Borovaya 8; creditCard: 1000;
    debit: 751494,000; credit: 309032,000;
    cityCallsMinutes: 22; longDistanceCallsMinutes: 0
}

```

Рисунок 4 – Работа программы

**Задание 4:** создать классы, спецификации которых приведены ниже. Определить конструкторы и методы `setТип()`, `getТип()`, `toString()`. Определить дополнительно методы в классе, создающем массив объектов. Задать критерий выбора данных и вывести эти данные на консоль. Car: id, Марка, Модель, Год выпуска, Цвет, Цена, Регистрационный номер. Создать массив объектов. Вывести: а) список автомобилей заданной марки; б) список автомобилей заданной модели, которые эксплуатируются больше n лет; с) список автомобилей заданного года выпуска, цена которых больше указанной. Сделать это, реализуя собственные обработчики исключений и исключения ввода/вывода.

Вычислить определитель матрицы.

Код класса `InvalidInputException`:

```

package lab5_var2_7;

public class InvalidInputException extends Exception {
    public InvalidInputException(String message) {
        super(message);
    }
}

```

## Код модуля Car:

```
package lab5_var2_8;

import java.text.DateFormat;
import java.text.SimpleDateFormat;

public class Car {
    private int id;
    private String brand;
    private String model;
    private int yearOfManufacture;
    private String color;
    private double price;
    private String regNum;

    public Car(int id, String brand, String model, int yearOfManufacture,
               String color, double price, String regNum) {
        try {
            this.setId(id);
            this.setBrand(brand);
            this.setModel(model);
            this.setYearOfManufacture(yearOfManufacture);
            this.setColor(color);
            this.setPrice(price);
            this.setRegNum(regNum);
        } catch (InvalidInputException e) {
            // Handle exception related to mathematical operations or memory
            issues
                System.err.println("An error occurred while setting car properties: "
+ e.getMessage());
        }
    }

    @Override
    public String toString() {
        DateFormat dateFormat = new SimpleDateFormat("yyyy.MM.dd");
        return String.format("
        Phone = {
            \tid: %d;
            \tbrand: %s;
            \tmodel: %s;
            \tyearOfManufacture: %s;
            \tcolor: %s;
            \tprice: %.3f;
            \tregNum: %s;
        }",
            id, brand, model, yearOfManufacture, color, price, regNum);
    }

    public int getId() {
        return id;
    }

    public void setId(int id) throws InvalidInputException {
        if (id <= 0) {
            throw new InvalidInputException("ID must be a positive integer.");
        }
        this.id = id;
    }

    public String getBrand() {
        return brand;
    }
}
```

```

    public void setBrand(String brand) throws InvalidInputException {
        if (brand == null || brand.isEmpty()) {
            throw new InvalidInputException("brand cannot be null or empty.");
        }
        this.brand = brand;
    }

    public String getModel() {
        return model;
    }

    public void setModel(String model) throws InvalidInputException {
        if (model == null || model.isEmpty()) {
            throw new InvalidInputException("model cannot be null or empty.");
        }
        this.model = model;
    }

    public int getYearOfManufacture() {
        return yearOfManufacture;
    }

    public void setYearOfManufacture(int yearOfManufacture) throws
InvalidInputException {
        if (yearOfManufacture <= 0) {
            throw new InvalidInputException("yearOfManufacture must be a positive
integer.");
        }
        this.yearOfManufacture = yearOfManufacture;
    }

    public String getColor() {
        return color;
    }

    public void setColor(String color) throws InvalidInputException {
        if (color == null || color.isEmpty()) {
            throw new InvalidInputException("Color cannot be null or empty.");
        }
        this.color = color;
    }

    public double getPrice() {
        return price;
    }

    public void setPrice(Double price) throws InvalidInputException {
        if (price <= 0) {
            throw new InvalidInputException("price must be a positive integer.");
        }
        this.price = price;
    }

    public String getRegNum() {
        return regNum;
    }

    public void setRegNum(String regNum) throws InvalidInputException {
        if (regNum == null || regNum.isEmpty()) {
            throw new InvalidInputException("regNum cannot be null or empty.");
        }
        this.regNum = regNum;
    }
}

```



## Код модуля Main:

```
package lab3_var2_8;

import java.util.Calendar;
import java.util.Objects;
import java.util.Random;
import java.util.Scanner;

import static java.lang.System.*;

public class Main {
    public static void main(String[] args) {
        Random random = new Random();
        Scanner scanner = new Scanner(in);

        int n = random.nextInt(5) + 2;
        out.printf("Input info about %d cars\n", n);
        Car[] cars = new Car[n];

        for (int i = 0; i < n; i++) {
            int id = i + 1;
            out.printf("Input brand of car №%d:\n", id);
            String brand = scanner.nextLine();
            out.printf("Input model of car №%d:\n", id);
            String model = scanner.nextLine();
            out.printf("Input color of car №%d:\n", id);
            String color = scanner.nextLine();
            out.printf("Input regNum of car №%d:\n", id);
            String regNum = scanner.nextLine();

            int price = random.nextInt(3000000) + 1000000;
            int yearOfManufacture = random.nextInt(24) + 2000;

            cars[i] = new Car(id, brand, model, yearOfManufacture, color, price,
regNum);
        }

        printAll(cars);
        out.println("-----\nSet brand for filter №1:");
        String brand = scanner.nextLine();
        out.printf("-----\nFiltered by brand '%s':\n", brand);
        brandFilter(brand, cars);
        out.println("-----\nSet model for filter №2:");
        String model = scanner.nextLine();
        out.println("-----\nSet count of years for filter №2:");
        int yearCount = scanner.nextInt();
        out.printf("-----\nCars with model '%s' older than %d:\n",
model, yearCount);
        oldModelFilter(model, yearCount, cars);
        out.println("-----\nSet year of manufacture for filter №3:");
        int year = scanner.nextInt();
        out.println("-----\nSet car price for filter №3:");
        double price = scanner.nextDouble();
        out.printf("-----\nCars released in %d that more expensive
than %.3f:\n", year, price);
        yearPriceFilter(year, price, cars);

    }

    private static void printAll(Car[] cars) {
        for (Car car : cars) {
            out.println(car);
        }
    }
}
```

```

    }
}

private static void brandFilter(String brand, Car[] cars) {
    int flag = 0;
    for (Car car : cars) {
        if (Objects.equals(car.getBrand(), brand)) {
            out.println(car);
            flag = 1;
        }
    }
    if (flag == 0) out.println("No one");
}

private static void oldModelFilter(String model, int yearCount, Car[] cars) {
    int flag = 0;
    for (Car car : cars) {
        if ((Objects.equals(car.getModel(), model)) &&
            ((Calendar.getInstance().get(Calendar.YEAR) -
car.getYearOfManufacture()) > yearCount)) {
            out.println(car);
            flag = 1;
        }
    }
    if (flag == 0) out.println("No one");
}

private static void yearPriceFilter(int year, double price, Car[] cars) {
    int flag = 0;
    for (Car car : cars) {
        if (Objects.equals(car.getYearOfManufacture(), year) &&
            (car.getPrice() > price)) {
            out.println(car);
            flag = 1;
        }
    }
    if (flag == 0) out.println("No one");
}
}

```

Работа программы показана на рисунках 5-6.

```

Input info about 3 cars      Phone = {      Set brand for filter №1:
Input brand of car №1:      id: 1;      1
1                            brand: 1;      -----
Input model of car №1:     model: m1;      Filtered by brand '1':
m1                          yearOfManufacture: 2001;      Phone = {
Input color of car №1:     color: red;      id: 1;
red                        price: 1039124,000;      brand: 1;
Input regNum of car №1:    regNum: 111;      model: m1;
111                       }      yearOfManufacture: 2001;
Input brand of car №2:     }      color: red;
2                          Phone = {      price: 1039124,000;
Input model of car №2:     id: 2;      regNum: 111;
m1                          brand: 2;      }
Input color of car №2:     model: m1;      Phone = {
blue                       yearOfManufacture: 2013;      id: 3;
Input regNum of car №2:    color: blue;      brand: 1;
222                       price: 1600003,000;      model: m2;
Input brand of car №3:     regNum: 222;      yearOfManufacture: 2020;
1                          }      color: pink;
Input model of car №3:     Phone = {      price: 2539414,000;
m2                          id: 3;      regNum: 333;
Input color of car №3:     brand: 1;      }
pink                       model: m2;      -----
Input regNum of car №3:    yearOfManufacture: 2020;      Set model for filter №2:
333                       color: pink;      m1
                          price: 2539414,000;      -----
                          regNum: 333;      Set count of years for filter №2:
                          }      12
                          -----
                          Cars with model 'm1' older than 12:
                          Phone = {
                          id: 1;
                          brand: 1;
                          model: m1;
                          yearOfManufacture: 2001;
                          color: red;
                          price: 1039124,000;
                          regNum: 111;
                          }
                          -----

```

Рисунок 5 – Работа программы

```

Set year of manufacture for filter №3:
2020
-----
Set car price for filter №3:
100
-----
Cars released in 2020 that more expensive than 100,000:
Phone = {
  id: 3;
  brand: 1;
  model: m2;
  yearOfManufacture: 2020;
  color: pink;
  price: 2539414,000;
  regNum: 333;
}

```

Рисунок 6 – Работа программы

**Задание 5:** В следующих заданиях требуется ввести последовательность строк из текстового потока и выполнить указанные действия. При этом могут рассматриваться два варианта:

- каждая строка состоит из одного слова;
- каждая строка состоит из нескольких слов.

Имена входного и выходного файлов, а также абсолютный путь к ним могут быть введены как параметры командной строки или храниться в файле.

В каждом слове стихотворения Николая Заболоцкого заменить первую букву слова на прописную.

Код модуля Main:

```
package lab5_var3_7;

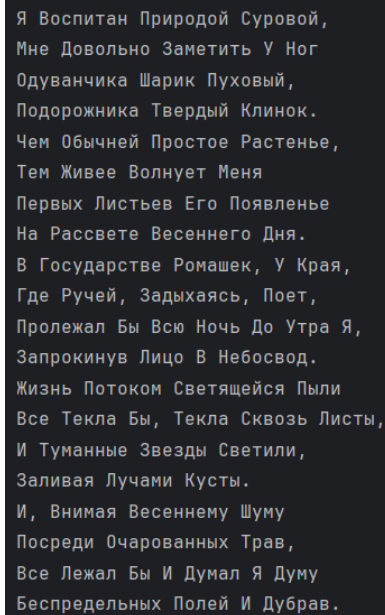
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

public class Main {
    public static void main(String[] args) {
        try {
            BufferedReader reader = new BufferedReader(new
            FileReader("src/lab5_var3_7/input.txt"));
            String line;
            while ((line = reader.readLine()) != null) {
                String[] words = line.split(" ");
                for (String word : words) {
                    if (!word.isEmpty()) {
                        char firstChar = Character.toUpperCase(word.charAt(0));
                        String capitalizedWord = firstChar + word.substring(1);
                        System.out.print(capitalizedWord + " ");
                    }
                }
                System.out.println();
            }
            reader.close();
        } catch (IOException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

Содержимое input.txt:

```
Я воспитан природой суровой,
Мне довольно заметить у ног
Одуванчика шарик пуховый,
Подорожника твердый клинок.
Чем обычной простое растение,
Тем живее волнует меня
Первых листьев его появление
На рассвете весеннего дня.
В государстве ромашек, у края,
Где ручей, задыхаясь, поет,
Пролежал бы всю ночь до утра я,
Запрокинув лицо в небосвод.
Жизнь потоком светящейся пыли
Все текла бы, текла сквозь листья,
И туманные звезды светили,
Заливая лучами кусты.
И, внимая весеннему шуму
Посреди очарованных трав,
Все лежал бы и думал я думу
Беспредельных полей и дубрав.
```

Результат работы программы показан на рисунке 7.



Я Воспитан Природой Суровой,  
Мне Довольно Заметить У Ног  
Одуванчика Шарик Пуховый,  
Подорожника Твердый Клинок.  
Чем Обычной Простое Растенье,  
Тем Живее Волнует Меня  
Первых Листьев Его Появление  
На Рассвете Весеннего Дня.  
В Государстве Ромашек, У Края,  
Где Ручей, Задыхаясь, Поет,  
Пролежал Бы Всю Ночь До Утра Я,  
Запрокинув Лицо В Небосвод.  
Жизнь Потоком Светящейся Пыли  
Все Текла Бы, Текла Сквозь Листы,  
И Туманные Звезды Светили,  
Заливая Лучами Кусты.  
И, Внимая Весеннему Шуму  
Посреди Очарованных Трав,  
Все Лежал Бы И Думал Я Думу  
Беспредельных Полей И Дубрав.

Рисунок 7 – Работа программы

**Задание 6:** В следующих заданиях требуется ввести последовательность строк из текстового потока и выполнить указанные действия. При этом могут рассматриваться два варианта:

- каждая строка состоит из одного слова;
- каждая строка состоит из нескольких слов.

Имена входного и выходного файлов, а также абсолютный путь к ним могут быть введены как параметры командной строки или храниться в файле.

Определить частоту повторяемости букв и слов в стихотворении Александра Пушкина.

Код модуля Main:

```
package lab5_var3_8;
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;

public class Main {
    public static void main(String[] args) {
        try {
            BufferedReader reader = new BufferedReader(new
            FileReader("src/lab5_var3_8/input.txt"));
            String line;
            Map<Character, Integer> charFrequency = new HashMap<>();
            Map<String, Integer> wordFrequency = new HashMap<>();

            while ((line = reader.readLine()) != null) {
                String[] words = line.split("\\\\,|\\\\\\\\.\\\\\\\\:");
```

```

        for (String word : words) {
            // Count word frequency
            wordFrequency.put(word, wordFrequency.getOrDefault(word, 0) +
1);

            // Count character frequency
            for (char c : word.toCharArray()) {
                charFrequency.put(c, charFrequency.getOrDefault(c, 0) +
1);
            }
        }
        System.out.println("Character Frequency:");
        for (Map.Entry<Character, Integer> entry : charFrequency.entrySet()) {
            System.out.println(entry.getKey() + ": " + entry.getValue());
        }
        System.out.println("\nWord Frequency:");
        for (Map.Entry<String, Integer> entry : wordFrequency.entrySet()) {
            System.out.println(entry.getKey() + ": " + entry.getValue());
        }
        reader.close();
    } catch (IOException e) {
        System.out.println("Error: " + e.getMessage());
    }
}
}

```

### Содержимое input.txt:

```

Духовной жаждою томим,
В пустыне мрачной я влачился, —
И шестикрылый серафим
На перепутье мне явился.
Перстами легкими как сон
Моих зениц коснулся он.
Отверзлись вещие зеницы,
Как у испуганной орлицы.
Моих ушей коснулся он, —
И их наполнил шум и звон:
И внял я неба содроганье,
И горний ангелов полет,
И гад морских подводный ход,
И дольней лозы прозябанье.
И он к устам моим приник,
И вырвал грешный мой язык,
И празднословный и лукавый,
И жало мудрыя змеи
В уста замершие мои
Вложил десницею кровавой.
И он мне грудь рассек мечом,
И сердце трепетное вынул,
И угль, пылающий огнем,
Во грудь отверстую водвинул.
Как труп в пустыне я лежал,
И бога глас ко мне воззвал:
«Восстань, пророк, и виждь, и внемли,
Исполнись волею моей,
И, обходя моря и земли,
Глаголом жги сердца людей».

```

Часть вывода после программы показана на рисунках 8-9.

```
Character Frequency:
р: 29
с: 30
т: 16
у: 21
ф: 1
х: 7
ц: 6
ч: 3
ш: 5
щ: 2
ы: 16
ь: 11
ю: 6
я: 14
В: 5
Г: 1
Д: 1
```

Рисунок 8 – Работа программы

```
Word Frequency:
: 7
жги: 1
уста: 1
их: 1
шестикрылый: 1
огнем: 1
морских: 1
Вложил: 1
В: 2
язык: 1
-: 2
И: 15
ангелов: 1
звон: 1
сердце: 1
сердца: 1
легкими: 1
пылающий: 1
замершие: 1
десницею: 1
рассек: 1
зениц: 1
в: 1
Перстами: 1
```

Рисунок 9 – Работа программы

**Задание 7:** прочитать текст Java-программы и удалить из него все “лишние” пробелы и табуляции, оставив только необходимые для разделения операторов. Для вывода результатов создавать новую директорию и файл средствами класса File.

Код модуля Main:

```
package lab5_var4_7;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

public class Main {
    public static void main(String[] args) {
        try {
            File inputFile = new File("src/lab5_var4_7/input.txt");
            File outputDir = new File("output");
            outputDir.mkdir();
            File outputFile = new File(outputDir, "output.txt");

            BufferedReader reader = new BufferedReader(new FileReader(inputFile));
            BufferedWriter writer = new BufferedWriter(new
FileWriter(outputFile));

            String line;
            while ((line = reader.readLine()) != null) {
                String trimmedLine = line.replaceAll("\\s+", " "); // Remove extra
spaces and tabs
                writer.write(trimmedLine);
                writer.newLine();
            }

            reader.close();
            writer.close();

            System.out.println("Output written to: " +
outputFile.getAbsolutePath());
        } catch (IOException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

Содержимое input.txt:

```
public class Example {
    public static void main(String[] args) {
        System.out.println("Пример текста с лишними пробелами и табуляциями");
        if (true) {
            System.out.println("Условие выполнено");
        } else {
            System.out.println("Условие не выполнено");
        }
    }
}
```



Результат работы программы показан на рисунке 10.

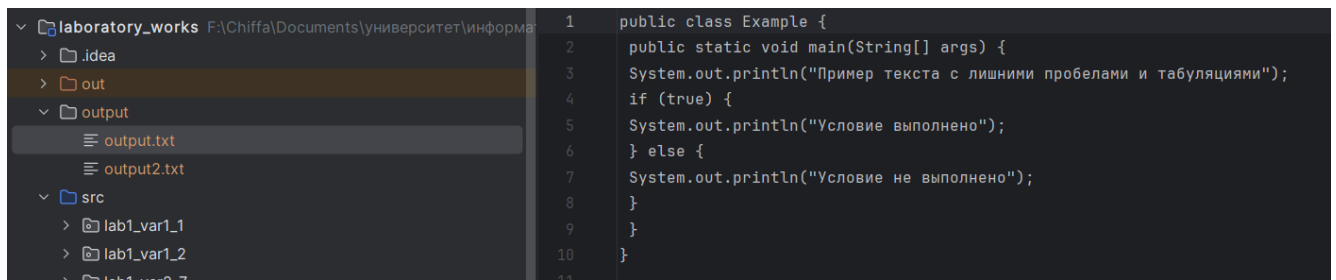


Рисунок 10 – Работа программы

**Задание 8:** из текста Java-программы удалить все виды комментариев. Для вывода результатов создавать новую директорию и файл средствами класса File.

Код модуля Main:

```
package lab5_var4_8;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

public class Main {
    public static void main(String[] args) {
        try {
            File inputFile = new File("src/lab5_var4_8/input.txt");
            File outputDir = new File("output");
            outputDir.mkdir();
            File outputFile = new File(outputDir, "output2.txt");

            BufferedReader reader = new BufferedReader(new FileReader(inputFile));
            BufferedWriter writer = new BufferedWriter(new
FileWriter(outputFile));

            String line;
            boolean inComment = false;
            while ((line = reader.readLine()) != null) {
                if (!inComment) {
                    int index = line.indexOf("//");
                    if (index != -1) {
                        line = line.substring(0, index);
                    }

                    index = line.indexOf("/*");
                    if (index != -1) {
                        inComment = true;
                        line = line.substring(0, index);
                    }
                } else {
                    int index = line.indexOf("*/");
                    if (index != -1) {
                        inComment = false;
                        line = line.substring(index + 2);
                    } else {
                        continue;
                    }
                }
            }
        }
    }
}
```

```

        if (!line.trim().isEmpty()) {
            writer.write(line);
            writer.newLine();
        }

        reader.close();
        writer.close();

        System.out.println("Output written to: " +
outputFile.getAbsolutePath());
    } catch (IOException e) {
        System.out.println("Error: " + e.getMessage());
    }
}
}

```

Содержимое input.txt:

```

public class Example {
    public static void main(String[] args) {
        // Это однострочный комментарий
        System.out.println("Пример текста с комментариями");

        /*
            Это
            многострочный
            комментарий
        */

        // TODO: Добавить обработку исключений

        // FIXME: Исправить этот код
    }
}

```

Работа программы показана на рисунке 11.

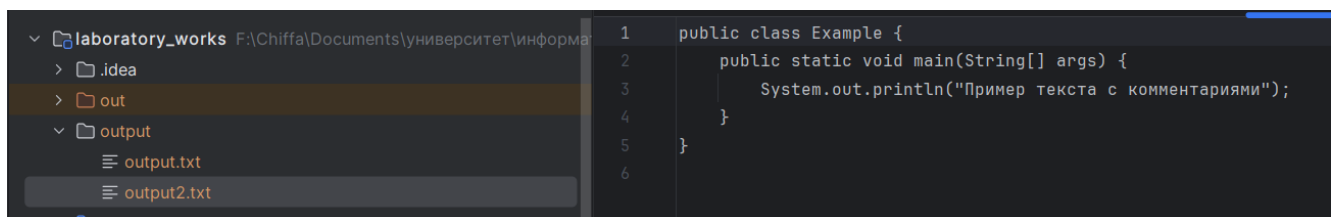


Рисунок 11 – Работа программы

**Ссылка на git-репозиторий:** [https://github.com/FedorLuchkin/Java\\_bmstu](https://github.com/FedorLuchkin/Java_bmstu)

**Вывод:** были освоены принципы работы с исключениями и файлами на языке программирования Java.