



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 № 11

Название: Исключения и файлы

Дисциплина: Языки программирования для работы с большими данными

Студент

ИУ6-23М

(Группа)

(Подпись, дата)

С.В.Мельников

(И.О. Фамилия)

Преподаватель

(Подпись, дата)

П.В. Степанов

(И.О. Фамилия)

Москва, 2023

Цель работы

Изучить работу с исключениями в языке программирования Java. Изучить работу с файлами.

Задание 1 (Вариант 1, Задание 1):

Выполнить задания на основе варианта 1 лабораторной работы 3, 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.

Определить класс Вектор размерности n . Реализовать методы сложения, вычитания, умножения, инкремента, декремента, индексирования. Определить массив из m объектов. Каждую из пар векторов передать в методы, возвращающие их скалярное произведение и длины. Вычислить и вывести углы между векторами.

Листинг программы:

Код класса Matix:

```
public class Matrix {
    private final int[][] values;
    private final int rows;
    private final int columns;

    public Matrix(int n) {
        if (n < 1) {
            throw new IllegalArgumentException("Incorrect size of matrix!");
        }
        this.rows = n;
        this.columns = n;
        this.values = new int[n][n];
        for (int i = 0; i < n; ++i)
            for (int j = 0; j < n; j++)
                this.values[i][j] = 0;
    }

    public Matrix(int n, int m) {
        if (n < 1 || m < 1) {
            throw new IllegalArgumentException("Incorrect size of matrix!");
        }
        this.rows = m;
        this.columns = n;
        this.values = new int[m][n];
    }

    public Matrix(int[][] values) {
```

```

        if (values == null) {
            throw new IllegalArgumentException("Incorrect values for matrix!");
        }
        if (values.length == 0) {
            throw new IllegalArgumentException("Incorrect size of values array!");
        }
        if (values[0].length == 0) {
            throw new IllegalArgumentException("Incorrect size of values array!");
        }
        this.values = new int[values.length][];
        this.rows = values.length;
        this.columns = values[0].length;
        for (int i = 0; i < values.length; i++)
            this.values[i] = values[i].clone();
    }

    public Matrix add(Matrix matrix) {
        if (matrix == null) {
            throw new IllegalArgumentException("Incorrect matrix argument!");
        }
        if (this.rows != matrix.rows || this.columns != matrix.columns) {
            throw new ArithmeticException("Invalid sizes of matrix!");
        }
        int[][] new_values = new int[this.rows][this.columns];
        for (int i = 0; i < this.rows; i++)
            for (int j = 0; j < this.columns; j++)
                new_values[i][j] = this.values[i][j] + matrix.values[i][j];
        return new Matrix(new_values);
    }

    public Matrix subtract(Matrix matrix) {
        if (matrix == null) {
            throw new IllegalArgumentException("Incorrect matrix argument!");
        }
        if (this.rows != matrix.rows || this.columns != matrix.columns) {
            throw new ArithmeticException("Invalid sizes of matrix!");
        }
        int[][] new_values = new int[this.rows][this.columns];
        for (int i = 0; i < this.rows; i++)
            for (int j = 0; j < this.columns; j++)
                new_values[i][j] = this.values[i][j] - matrix.values[i][j];
        return new Matrix(new_values);
    }

    public Matrix multiply(Matrix matrix) {
        if (matrix == null) {
            throw new IllegalArgumentException("Incorrect matrix argument!");
        }
        if (this.columns != matrix.rows) {
            throw new ArithmeticException("Invalid sizes of matrix!");
        }
        int[][] new_values = new int[this.rows][matrix.columns];
        for (int i = 0; i < this.rows; i++)
            for (int j = 0; j < this.columns; j++)
                for (int k = 0; k < matrix.columns; k++)
                    new_values[i][k] += this.values[i][j] * matrix.values[j][k];
        return new Matrix(new_values);
    }

    public Matrix multiply(int coefficient) {
        int[][] new_values = new int[this.rows][this.columns];
        for (int i = 0; i < this.rows; i++)
            for (int j = 0; j < this.columns; j++)
                new_values[i][j] = this.values[i][j] * coefficient;
        return new Matrix(new_values);
    }
}

```

```

public void increment() {
    for (int i = 0; i < this.rows; i++)
        for (int j = 0; j < this.columns; j++)
            this.values[i][j] += 1;
}

public void decrement() {
    for (int i = 0; i < this.rows; i++)
        for (int j = 0; j < this.columns; j++)
            this.values[i][j] -= 1;
}

public int get_element(int row, int column) {
    if (row < 0 || this.rows <= row || column < 0 || this.columns <= column) {
        throw new IndexOutOfBoundsException("Illegal indexes!");
    }
    return this.values[row][column];
}

@Override
public String toString() {
    StringBuilder result = new StringBuilder();
    for (int i = 0; i < this.rows; i++) {
        for (int j = 0; j < this.columns; j++)
            result.append(this.values[i][j]).append(" ");
        result.append("\n");
    }
    return result.toString();
}

public static int scalarProduct(Matrix a, Matrix b) {
    if (a == null || b == null) {
        throw new IllegalArgumentException("Incorrect matrix argument!");
    }
    if (a.columns > 1 || b.columns > 1 || a.rows != b.rows) {
        throw new ArithmeticException("Must be vectors with same size!");
    }
    int result = 0;
    for (int i = 0; i < a.rows; i++)
        result += a.values[i][0] * b.values[i][0];
    return result;
}

public static double angle(Matrix a, Matrix b) {
    if (a == null || b == null) {
        throw new IllegalArgumentException("Incorrect matrix argument!");
    }
    if (a.columns > 1 || b.columns > 1 || a.rows != b.rows) {
        throw new ArithmeticException("Must be vectors with same size!");
    }
    double scalarProduct = 0;
    int a_part = 0;
    int b_part = 0;
    for (int i = 0; i < a.rows; i++) {
        scalarProduct += a.values[i][0] * b.values[i][0];
        a_part += pow(a.values[i][0], 2);
        b_part += pow(b.values[i][0], 2);
    }
    if (a_part == 0 || b_part == 0)
        return 0;
    return Math.acos(scalarProduct / (Math.sqrt(a_part) * Math.sqrt(b_part)))
* 180 / Math.PI;
}

```

```

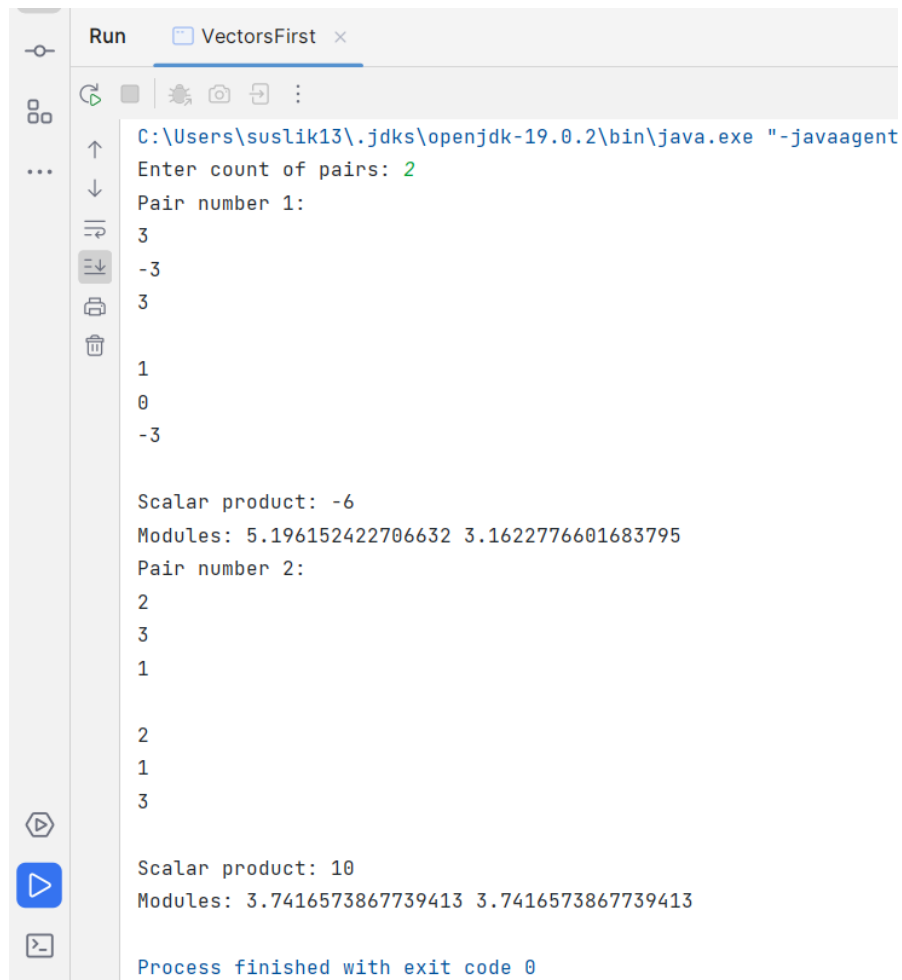
public static double module(Matrix a) {
    if (a == null) {
        throw new IllegalArgumentException("Incorrect matrix argument!");
    }
    if (a.columns > 1) {
        throw new ArithmeticException("Must be vector!");
    }
    int result = 0;
    for (int i = 0; i < a.rows; i++)
        result += pow(a.values[i][0], 2);
    return Math.sqrt(result);
}

public static boolean isCollinear(Matrix a, Matrix b) {
    if (a == null || b == null) {
        throw new IllegalArgumentException("Incorrect matrix argument!");
    }
    if (a.columns > 1 || b.columns > 1 || a.rows != b.rows) {
        throw new ArithmeticException("Must be vectors with same size!");
    }
    if (a.rows == 0 || b.rows == 0)
        return true;
    boolean result = true;
    double coefficient = a.values[0][0] * 1.0 / b.values[0][0];
    if (Double.isNaN(coefficient)) {
        return false;
    }
    for (int i = 1; i < a.rows; i++)
        result = result && (Math.abs(a.values[i][0] * 1.0 / b.values[i][0] -
coefficient)) < 0.00001;
    return result;
}

public static boolean isOrthogonal(Matrix a, Matrix b) {
    return Matrix.scalarProduct(a, b) == 0;
}
}

```

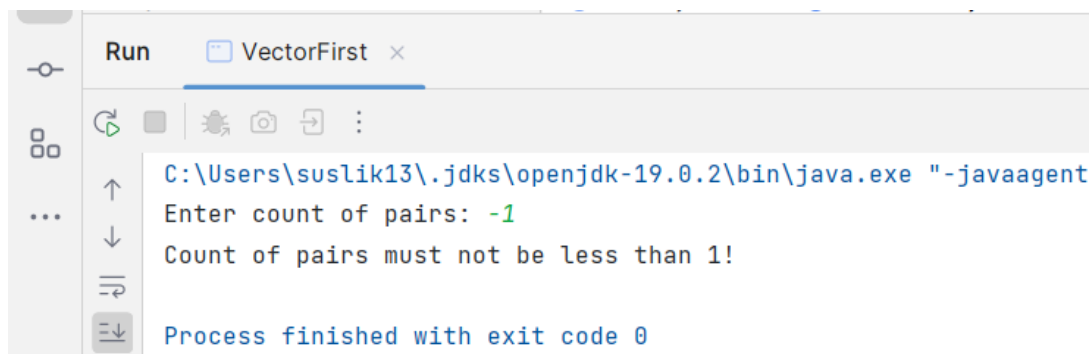
Работа программы представлена на рисунке 1.



```
Run VectorsFirst x
C:\Users\suslik13\.jdk\openjdk-19.0.2\bin\java.exe "-javaagent
Enter count of pairs: 2
Pair number 1:
3
-3
3
1
0
-3
Scalar product: -6
Modules: 5.196152422706632 3.1622776601683795
Pair number 2:
2
3
1
2
1
3
Scalar product: 10
Modules: 3.7416573867739413 3.7416573867739413
Process finished with exit code 0
```

Рисунок 1 – Работа программы VectorsFirst.java

Если будут введены не корректные данные, программа обработает ошибку, как, например представлено на рисунке 2:



```
Run VectorFirst x
C:\Users\suslik13\.jdk\openjdk-19.0.2\bin\java.exe "-javaagent
Enter count of pairs: -1
Count of pairs must not be less than 1!
Process finished with exit code 0
```

Рисунок 2 – Работа программы VectorsFirst.java с ошибкой


Задание 2 (Вариант 1, Задание 2):

Выполнить задания на основе варианта 1 лабораторной работы 3, 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.

Определить класс Вектор размерности n . Определить несколько конструкторов. Реализовать методы для вычисления модуля вектора, скалярного произведения, сложения, вычитания, умножения на константу. Объявить массив объектов. Написать метод, который для заданной пары векторов будет определять, являются ли они коллинеарными или ортогональными.

Листинг программы общий с первым заданием.

Работа программы представлена на рисунке 3.



```
Run VectorsSecond x
C:\Users\suslik13\.jdk\openjdk-19.0.2\bin\java.exe "-javaagent:
Enter count of pairs: 2
Pair number 1:
3
3
1
Is collinear? : false
Is orthogonal?: false
Pair number 2:
3
-2
3
Is collinear? : false
Is orthogonal?: false
Process finished with exit code 0
```

Рисунок 3 – Работа программы VectorsSecond.java

Если будут введены не корректные данные, программа обработает ошибку, как, например представлено на рисунке 4:

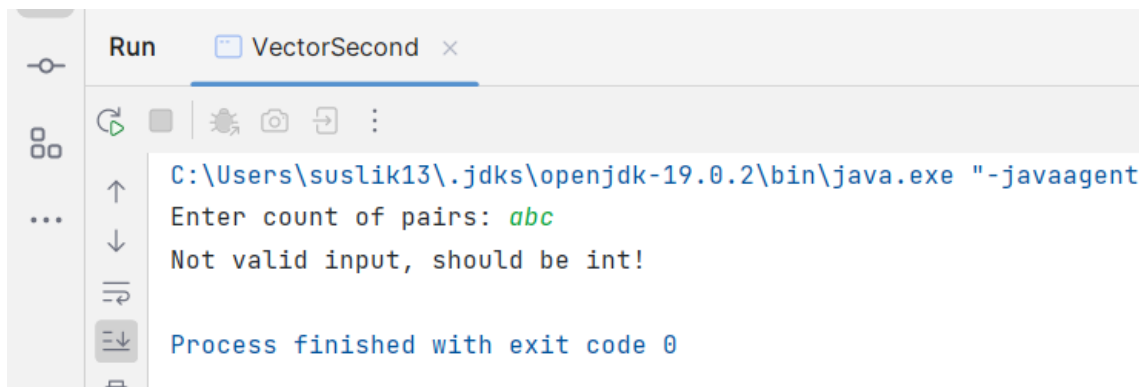


Рисунок 4 – Работа программы VectorsSecond.java с ошибкой

Задание 3 (Вариант 2, Задание 1):

Выполнить задания из варианта 2 лабораторной работы 3, реализуя собственные обработчики исключений и исключения ввода/вывода.

Создать классы, спецификации которых приведены ниже. Определить конструкторы и методы setType(), getType(), toString(). Определить дополнительно методы в классе, создающем массив объектов. Задать критерий выбора данных и вывести эти данные на консоль.

Student: id, Фамилия, Имя, Отчество, Дата рождения, Адрес, Телефон, Факультет, Курс, Группа. Создать массив объектов. Вывести: а) список студентов заданного факультета; б) списки студентов для каждого факультета и курса; с) список студентов, родившихся после заданного года; д) список учебной группы.

Листинг программы:

Код класса StudentMain:

```
public class StudentMain {  
    public static void printStudentsFromFaculty(Student[] students, String  
faculty) {  
        boolean success = false;  
        for (Student student: students) {  
            if (student.getFaculty().equals(faculty)) {  
                success = true;  
                System.out.println(student);  
            }  
        }  
    }  
}
```



```

        if (!success) {
            System.out.println("There are no students in faculty");
        }
    }

    public static void printStudentsByFacultyAndCourse(Student[] students) {
        HashMap<String, HashMap<Integer, ArrayList<Student>>> grouped_students =
new HashMap<>();
        for (Student student: students) {
            String faculty = student.getFaculty();
            int course = student.getCourse();
            if (grouped_students.containsKey(faculty)) {
                if (grouped_students.get(faculty).containsKey(course)) {
                    grouped_students.get(faculty).get(course).add(student);
                } else {
                    ArrayList<Student> students_in_course = new ArrayList<>();
                    students_in_course.add(student);
                    grouped_students.get(faculty).put(course, students_in_course);
                }
            } else {
                HashMap<Integer, ArrayList<Student>> students_on_faculty = new
HashMap<>();
                ArrayList<Student> students_in_course = new ArrayList<>();
                students_in_course.add(student);
                students_on_faculty.put(course, students_in_course);
                grouped_students.put(faculty, students_on_faculty);
            }
        }
        for (Map.Entry<String, HashMap<Integer, ArrayList<Student>>> entry:
grouped_students.entrySet()) {
            String f_key = entry.getKey();
            HashMap<Integer, ArrayList<Student>> value = entry.getValue();
            for (Map.Entry<Integer, ArrayList<Student>> inner_entry:
value.entrySet()) {
                int c_key = inner_entry.getKey();
                ArrayList<Student> s_value = inner_entry.getValue();
                System.out.println("Faculty: " + f_key + ", course: " + c_key +
":");

                for (Student student: s_value) {
                    System.out.println(student);
                }
            }
        }

        public static void printStudentsAfterDate(Student[] students, Date date) {
            for (Student student: students) {
                if (student.getBirthDay().after(date)) {
                    System.out.println(student);
                }
            }
        }

        public static void printStudentsFromGroup(Student[] students, String faculty,
int course, int group) {
            for (Student student : students) {
                if (student.getGroup() == group &&
student.getFaculty().equals(faculty) && student.getCourse() == course) {
                    System.out.println(student);
                }
            }
        }

        public static void main(String[] args) throws ParseException {
            Scanner scanner = new Scanner(System.in);

```

```

System.out.print("Enter count of students: ");
String buffer = scanner.nextLine();
int n;
try {
    n = Integer.parseInt(buffer);
} catch (NumberFormatException e) {
    System.out.println("Not valid input, should be int!");
    return;
}
Student[] students;
try {
    students = new Student[n];
} catch (NegativeArraySizeException e) {
    System.out.println("Count of students must be positive!");
    return;
} catch (OutOfMemoryError e) {
    System.out.println("Too big count of students!");
    return;
}
for (int i = 0; i < n; i++) {
    students[i] = new Student(
        i,
        DataGenerator.randomSurname(),
        DataGenerator.randomName(),
        DataGenerator.randomPatronymic(),
        DataGenerator.randomDate(),
        DataGenerator.randomAddress(),
        DataGenerator.randomPhoneNumber(),
        DataGenerator.randomFaculty(),
        DataGenerator.randomCourse(),
        DataGenerator.randomGroup()
    );
}
System.out.println("All students:");
for (int i = 0; i < n; i++) {
    System.out.println(students[i]);
}
System.out.print("Enter faculty name: ");
String f = scanner.nextLine();
printStudentsFromFaculty(students, f);
printStudentsByFacultyAndCourse(students);
System.out.print("Enter year: ");
buffer = scanner.nextLine();
int year;
try {
    year = Integer.parseInt(buffer);
} catch (NumberFormatException e) {
    System.out.println("Not valid input, should be int!");
    return;
}
Date date = new Date(year - 1900, Calendar.DECEMBER, 31);
printStudentsAfterDate(students, date);
System.out.print("Enter groups faculty: ");
String faculty = scanner.nextLine();
System.out.print("Enter groups course: ");
buffer = scanner.nextLine();
int course;
try {
    course = Integer.parseInt(buffer);
} catch (NumberFormatException e) {
    System.out.println("Not valid input, should be int!");
    return;
}
System.out.print("Enter group: ");
buffer = scanner.nextLine();

```

```

    int group;
    try {
        group = Integer.parseInt(buffer);
    } catch (NumberFormatException e) {
        System.out.println("Not valid input, should be int!");
        return;
    }
    printStudentsFromGroup(students, faculty, course, group);
}
}

```

Работа программы представлена на рисунке 5.

```

Run lab5.var2.StudentMain x
C:\Users\suslik13\.jids\openjdk-19.0.2\bin\java.exe "-javaagent:D:\JetBrains\IntelliJ IDEA 2022.3
Enter count of students: 2
All students:
Ivanov Grigory Artemyevich 03.01.2005 Voronezh, ul. Pervomayskaya, 22 +7(6483)581-48-86 IU 4 2
Dmitriev Oleg Danilevich 12.01.1994 Krasnoyarsk, ul. Rabochaya, 17 +7(601)784-17-56 FN 1 6
Enter faculty name: 12
There are no students in faculty
Faculty: FN, course: 1:
Dmitriev Oleg Danilevich 12.01.1994 Krasnoyarsk, ul. Rabochaya, 17 +7(601)784-17-56 FN 1 6
Faculty: IU, course: 4:
Ivanov Grigory Artemyevich 03.01.2005 Voronezh, ul. Pervomayskaya, 22 +7(6483)581-48-86 IU 4 2
Enter year: 0518
Not valid input, should be int!
Process finished with exit code 0

```

Рисунок 5 – Работа программы StudentMain.java с обработкой ошибок

Задание 4 (Вариант 2, Задание 2):

Выполнить задания из варианта 2 лабораторной работы 3, реализуя собственные обработчики исключений и исключения ввода/вывода.

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.

Customer: id, Фамилия, Имя, Отчество, Адрес, Номер кредитной карточки, Номер банковского счета. Создать массив объектов. Вывести: а) список покупателей в алфавитном порядке; б) список покупателей, у которых номер кредитной карточки находится в заданном интервале.

Листинг программы:

Код класса CustomerMain:

```
public class CustomerMain {
    public static void printCustomersSorted(Customer[] customers) {
        Arrays.sort(customers);
        for (Customer customer : customers) {
            System.out.println(customer);
        }
    }

    public static void printCustomersWithCardInRange(Customer[] customers, String
lower, String upper) {
        if (customers == null) {
            throw new IllegalArgumentException("Customer is empty!");
        }
        if (lower.length() != 16 || upper.length() != 16) {
            throw new IllegalArgumentException("Card number must be with 16
digits!");
        }
        for (int i = 0; i < lower.length(); i++) {
            if (!Character.isDigit(lower.charAt(i)) ||
!Character.isDigit(upper.charAt(i))) {
                throw new IllegalArgumentException("Card number must be only with
digits!");
            }
        }
        for (Customer customer : customers) {
            if (lower.compareTo(customer.getCard_number()) <= 0 &&
upper.compareTo(customer.getCard_number()) >= 0)
                System.out.println(customer);
        }
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter count of customers: ");
        String buffer = scanner.nextLine();
        int n;
        try {
            n = Integer.parseInt(buffer);
        } catch (NumberFormatException e) {
            System.out.println("Not valid input, should be int!");
            return;
        }
        Customer[] customers;
        try {
            customers = new Customer[n];
        } catch (NegativeArraySizeException e) {
            System.out.println("Count of customers must be positive!");
            return;
        } catch (OutOfMemoryError e) {
            System.out.println("Too big count of customers!");
            return;
        }
        System.out.println("Generated customers:");
        for (int i = 0; i < n; i++) {
            customers[i] = new Customer(
                i,
                DataGenerator.randomName(),
                DataGenerator.randomSurname(),
                DataGenerator.randomPatronymic(),
                DataGenerator.randomAddress(),
                DataGenerator.randomCardNumber(),
            );
        }
    }
}
```

```

        DataGenerator.randomAccountNumber()
    );
    System.out.println(customers[i]);
}
System.out.println("Sorted customers:");
printCustomersSorted(customers.clone());
System.out.print("Enter lower card number: ");
String card_lower = scanner.nextLine();
System.out.print("Enter upper card number: ");
String card_upper = scanner.nextLine();
try {
    printCustomersWithCardInRange(customers, card_lower, card_upper);
} catch (IllegalArgumentException e) {
    System.out.println("Card numbers must be only with 16 digits!");
}
}
}

```

Работа программы представлена на рисунке 6.

```

Run CustomerMain x
C:\Users\suslik13\.jdk\openjdk-19.0.2\bin\java.exe "-javaagent:D:\JetBrains\IntelliJ IDEA 2022.3.2\lib\idea_rt.jar=62770:D:\JetBra
Enter count of customers: 2
Generated customers:
Customer 0: Maksimov Stanislav Borisovich, card: 7155196087537841, account: 25343203154846264418, Izhevsk, Zelenaya str., 39
Customer 1: Anisimov Demid Antonovich, card: 3054391010262590, account: 44187516714883026785, Yekaterinburg, Berezovaya str., 24
Sorted customers:
Customer 1: Anisimov Demid Antonovich, card: 3054391010262590, account: 44187516714883026785, Yekaterinburg, Berezovaya str., 24
Customer 0: Maksimov Stanislav Borisovich, card: 7155196087537841, account: 25343203154846264418, Izhevsk, Zelenaya str., 39
Enter lower card number: 12
Enter upper card number: 9
Card numbers must be only with 16 digits!
Process finished with exit code 0

```

Рисунок 6 – Работа программы CustomerMain.java с обработкой ошибок

Задание 5 (Вариант 3, Задание 1):

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 каждой строке найти и удалить заданную подстроку.

Листинг программы:

Код класса SubstringDeleting:

```
public class SubstringDeleting {
    public static String input = "./src/lab5/var3/input.txt";
    public static String output = "./src/lab5/var3/output2.txt";

    public static void replace_str(String old_str, String new_str) throws
IOException {
        BufferedReader reader = new BufferedReader(new InputStreamReader(new
FileInputStream(input), StandardCharsets.UTF_8));
        FileWriter writer = new FileWriter(output);
        String tmp;
        String ost;
        int size = old_str.length();
        while ((tmp = reader.readLine()) != null) {
            int i = tmp.indexOf(old_str);
            if (i != -1) {
                ost = tmp;
                while (i != -1) {
                    writer.write(ost.substring(0, i));
                    writer.write(new_str);
                    ost = ost.substring(i + size);
                    i = ost.indexOf(old_str);
                    if (i == -1) {
                        writer.write(ost + "\n");
                    }
                }
            } else {
                writer.write(tmp + "\n");
            }
        }
        reader.close();
        writer.close();
    }

    public static void main(String[] args) throws IOException {
        replace_str("БЫЛО", "");
    }
}
```

```
}  
}
```

Работа программы:

Исходный текст:

Всё это было, было, было,
Свершился дней круговорот.
Какая ложь, какая сила
Тебя, прошедшее, вернет?

Текст после выполнения программы:

Всё это , , ,
Свершился дней круговорот.
Какая ложь, какая сила
Тебя, прошедшее, вернет?

Задание 6 (Вариант 3, Задание 2):

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 каждой строке стихотворения Александра Блока найти и заменить заданную подстроку на подстроку иной длины

Листинг программы такой же как в прошлой программе, только задается не пустая строка для замены, например замена “в” на “XYZ”.

Работа программы:

Исходный текст:

Всё это было, было, было,
Свершился дней круговорот.
Какая ложь, какая сила
Тебя, прошедшее, вернет?

Текст после выполнения программы:

Всё это было, было, было,
СХYZершился дней кругоXYZорот.
Какая ложь, какая сила
Тебя, прошедшее, XYZернет?

Задание 7 (Вариант 4, Задание 1):

При выполнении следующих заданий для вывода результатов создавать новую директорию и файл средствами класса File.

Прочитать текст Java-программы и все слова `public` в объявлении атрибутов и методов класса заменить на слово `private`.

Листинг программы:

Код класса PubToPriv:

```
public class PubToPriv {
    public static final File input = new File("./src/lab5/var4/input.txt");
    public static final File output = new File("./src/lab5/var4/output.txt");
    public static void main(String[] args) throws IOException {
        try (BufferedReader bufferedReader = new BufferedReader(new
FileReader(input));
            BufferedWriter bufferedWriter = new BufferedWriter(new
FileWriter(output))) {
            String line;
            while ((line=bufferedReader.readLine())!=null) {
                bufferedWriter.append(line.replace("public",
"private")).append(System.lineSeparator());
            }
        }
    }
}
```

Фрагмент исходного текста файла:

```
public int getId() {
    return id;
}

public String getSurname() {
    return surname;
}

public String getName() {
    return name;
}
```


Результат работы программы:

```
private int getId() {
    return id;
}

private String getSurname() {
    return surname;
}

private String getName() {
    return name;
}
```

Задание 8 (Вариант 4, Задание 2):

При выполнении следующих заданий для вывода результатов создавать новую директорию и файл средствами класса File.

Прочитать текст Java-программы и записать в другой файл в обратном порядке символы каждой строки.

Листинг программы:

Код класса ReverseOrder:

```
public class ReverseOrder {
    public static final File input = new File("./src/lab5/var4/input.txt");
    public static final File output = new File("./src/lab5/var4/output2.txt");

    public static void main(String[] args) throws IOException {
        try (BufferedReader bufferedReader = new BufferedReader(new
        FileReader(input));
            BufferedWriter bufferedWriter = new BufferedWriter(new
        FileWriter(output))) {
            String line;
            while ((line=bufferedReader.readLine())!=null) {
                bufferedWriter.append(new
        StringBuilder(line).reverse().toString()).append(System.lineSeparator());
            }
        }
    }
}
```

Фрагмент исходного текста файла:

```
public int getId() {
    return id;
}

public String getSurname() {
    return surname;
}

public String getName() {
    return name;
}
```

Результат работы программы:

```
{ }(dIteg tni cilbup
;di nruter
}

{ }(emanruSteg gnirtS cilbup
;emanrus nruter
}

{ }(emaNteg gnirtS cilbup
;eman nruter
}
```

Вывод

В ходе выполнения лабораторной работы были получены навыки работы с исключениями и с файлами в языке программирования Java