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, ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ 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 №5

Название: Исключения и файлы
Дисциплина: Языки программирования для работы с большими данными

Вариант: 2

Студент	ИУ6-22М		Н.А. Аскерова
	(Группа)	(Подпись, дата)	(И.О. Фамилия)
Преподаватель			П.В. Степанов
		(Подпись, дата)	(И.О. Фамилия)

Вариант 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.

2. 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 1 – Код программы

```
import java.util.ArrayList;

class Vector {
    private double[] elements;

    public Vector(double[] elements) {
        this.elements = elements;
    }

    public Vector(int dimension) {
        elements = new double[dimension];
    }

    public double modulus() {
        double sumOfSquares = 0.0;
        for (double element : elements) {
            sumOfSquares += element * element;
        }
        return Math.sqrt(sumOfSquares);
    }

    public double scalarProduct(Vector other) {
        double result = 0.0;
        for (int i = 0; i < elements.length; i++) {
            result += elements[i] * other.elements[i];
        }
        return result;
    }

    public Vector add(Vector other) {
        if (elements.length != other.elements.length) {
            throw new IllegalArgumentException("Vectors must have the same dimension");
        }
        double[] result = new double[elements.length];
        for (int i = 0; i < elements.length; i++) {
            result[i] = elements[i] + other.elements[i];
        }
        return new Vector(result);
    }
```

```
    public Vector subtract(Vector other) {
        if (elements.length != other.elements.length) {
            throw new IllegalArgumentException("Vectors must have the same dimension");
        }
        double[] result = new double[elements.length];
        for (int i = 0; i < elements.length; i++) {
            result[i] = elements[i] - other.elements[i];
        }
        return new Vector(result);
    }

    // Method for multiplying a vector by a constant
    public Vector multiply(double constant) {
        double[] result = new double[elements.length];
        for (int i = 0; i < elements.length; i++) {
            result[i] = elements[i] * constant;
        }
        return new Vector(result);
    }

    // Method for determining whether two vectors are collinear or orthogonal
    public String determineRelationship(Vector other) {
        if (elements.length != other.elements.length) {
            throw new IllegalArgumentException("Vectors must have the same dimension");
        }
        double scalarProduct = scalarProduct(other);
        if (scalarProduct == 0) {
            return "The vectors are orthogonal";
        } else if (scalarProduct == modulus() * other.modulus()) {
            return "The vectors are collinear";
        } else {
            return "The vectors are neither collinear nor orthogonal";
        }
    }
    public static void main(String[] args) {
        Vector[] vectors = new Vector[4];
        vectors[0] = new Vector(new double[] {-2.0, -2.0, 3.0});
        vectors[1] = new Vector(new double[] {4.0, 5.0, 6.0});
        vectors[2] = new Vector(new double[] {8.0, 8.0, 12.0});
        vectors[3] = new Vector(new double[] {4.0, 4.0, 6.0});

        Vector sum = vectors[0].add(vectors[1]);
        System.out.println("The sum of vectors[0] and vectors[1] is " + sum);

        Vector difference = vectors[1].subtract(vectors[2]);
        System.out.println("The difference of vectors[1] and vectors[2] is " + difference);

        Vector product = vectors[2].multiply(2.0);
        System.out.println("The product of vectors[2] and 2.0 is " + product);

        System.out.println("The modulus of vectors[0] is " + vectors[0].modulus());
        System.out.println("The scalar product of vectors[0] and vectors[1] is " + vectors[0].scalarProduct(vectors[1]));

        System.out.println("The relationship between vectors[0] and vectors[1] is " +
vectors[0].determineRelationship(vectors[1]));
        System.out.println("The relationship between vectors[1] and vectors[2] is " +
vectors[1].determineRelationship(vectors[2]));
```

```
        System.out.println("The relationship between vectors[0] and vectors[2] is " +
vectors[0].determineRelationship(vectors[2]));
        System.out.println("The relationship between vectors[0] and vectors[2] is " +
vectors[2].determineRelationship(vectors[3]));
    }

    // toString method for printing
    @Override
    public String toString() {
        StringBuilder stringBuilder = new StringBuilder("(");
        for (int i = 0; i < elements.length; i++) {
            stringBuilder.append(elements[i]);
            if (i != elements.length - 1) {
                stringBuilder.append(", ");
            }
        }
        stringBuilder.append(")");
        return stringBuilder.toString();
    }
}
```

```
The sum of vectors[0] and vectors[1] is (2.0, 3.0, 9.0)
The difference of vectors[1] and vectors[2] is (-4.0, -3.0, -6.0)
The product of vectors[2] and 2.0 is (16.0, 16.0, 24.0)
The modulus of vectors[0] is 4.123105625617661
The scalar product of vectors[0] and vectors[1] is 0.0
The relationship between vectors[0] and vectors[1] is The vectors are orthogonal
The relationship between vectors[1] and vectors[2] is The vectors are neither collinear nor orthogonal
The relationship between vectors[0] and vectors[2] is The vectors are neither collinear nor orthogonal
The relationship between vectors[0] and vectors[2] is The vectors are collinear

Process finished with exit code 0
```

Рисунок 1 – Результат работы программы

3. Определить класс Вектор в R3. Реализовать методы для проверки векторов на ортогональность, проверки пересечения не ортогональных векторов, сравнения векторов. Создать массив из m объектов. Определить, какие из векторов компланарны.

Листинг 2 – Код программы

```
import java.util.ArrayList;
import java.util.Arrays;
class Vector {
    private double x;
    private double y;
    private double z;

    public Vector(double x, double y, double z) {
        this.x = x;
        this.y = y;
        this.z = z;
    }

    public Vector() {
        this(0, 0, 0);
    }
```

```
public double modulus() {
   return Math.sqrt(x * x + y * y + z * z);
}

public double scalarProduct(Vector v) {
   return x * v.x + y * v.y + z * v.z;
}

public Vector add(Vector v) {
   return new Vector(x + v.x, y + v.y, z + v.z);
}

public Vector subtract(Vector v) {
   return new Vector(x - v.x, y - v.y, z - v.z);
}

public Vector multiply(double k) {
   return new Vector(k * x, k * y, k * z);
}

public boolean isCollinear(Vector v) {
   return x / v.x == y / v.y && y / v.y == z / v.z;
}

public boolean isOrthogonal(Vector v) {
   return scalarProduct(v) == 0;
}


public boolean intersects(Vector v) {
   return !isOrthogonal(v);
}

public boolean equals(Vector v) {
   return x == v.x && y == v.y && z == v.z;
}

public int compareTo(Vector v) {
   double modulus1 = modulus();
   double modulus2 = v.modulus();
   if (modulus1 == 0) {
      throw new IllegalArgumentException("Vector1 has 0 module");
   }
   if (modulus2 == 0) {
      throw new IllegalArgumentException("Vector2 has 0 module");
   }
   if (modulus1 < modulus2) {
      return -1;
   } else if (modulus1 > modulus2) {
      return 1;
   } else {
      return 0;
   }
}

public String toString() {
   return "(" + x + ", " + y + ", " + z + ")";
}
```

```
    public static void main(String[] args) {
        int m = 3;
        Vector[] vectors = new Vector[m];
        for (int i = 0; i < m; i++) {
            vectors[i] = new Vector();
        }
        Vector v1 = new Vector(1, 2, 3);
        Vector v2 = new Vector(2, 4, 6);
        Vector v3 = new Vector(2, -1, 0);
        for (int i = 0; i < m; i++) {
            for (int j = i + 1; j < m; j++) {
                for (int k = j + 1; k < m; k++) {
                    if (isCoplanar(vectors[i], vectors[j], vectors[k])) {
                        System.out.println("Vectors are coplanar.");
                    }
                }
            }
        }

        // Example usage of methods


        // Check orthogonality
        System.out.println("v1 and v2 are orthogonal: " + v1.isOrthogonal(v2));
        System.out.println("v1 and v3 are orthogonal: " + v1.isOrthogonal(v3));

        System.out.println("v1 and v3 are collinear: " + v1.isCollinear(v2));

        // Check intersection
        System.out.println("v1 and v2 intersect: " + v1.intersects(v2));
        System.out.println("v1 and v3 intersect: " + v1.intersects(v3));

        // Compare vectors
        System.out.println("v1 compared to v2: " + v1.compareTo(v2));
        System.out.println("v2 compared to v1: " + v2.compareTo(v1));
        System.out.println("v1 compared to v3: " + v1.compareTo(v3));

    }

    public static boolean isCoplanar(Vector v1, Vector v2, Vector v3) {
        return v1.scalarProduct(v2.crossProduct(v3)) == 0;
    }

    public Vector crossProduct(Vector v) {
        return new Vector(y * v.z - z * v.y, z * v.x - x * v.z, x * v.y - y * v.x);
    }

}
```

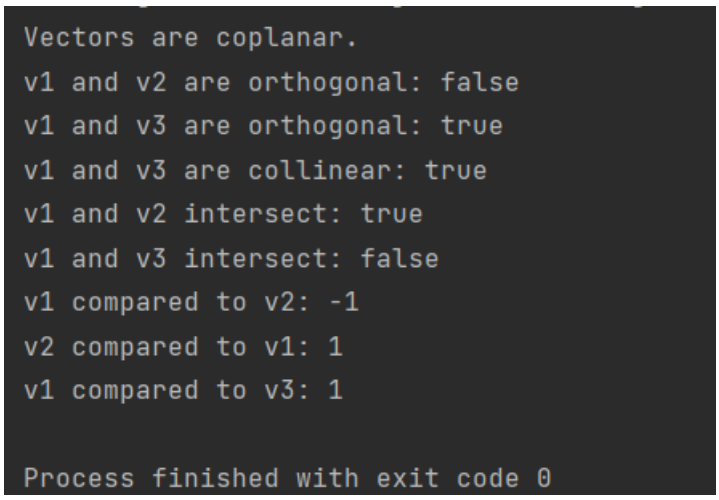


Рисунок 2 – Результат работы программы

Вариант 2

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

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

Листинг 3 – Код программы

```
import java.io.IOException;
import java.util.InputMismatchException;
import java.util.Scanner;
import java.util.Comparator;
import java.util.Arrays;

class MyException extends Exception {
   public MyException() {}
   public MyException(String msg) {
      super(msg);
   }
}
public class Main {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      Customer[] cust = new Customer[3];
      cust[0] = new Customer(1, "Anna;  ", "Petrova;  ", "Petrovna;  ", "Moscow;  ", 123, 56521);
      cust[1] = new Customer(2, "Boris;  ", "Artemyev;  ", "Alexandrovich;  ", "Zelenograd;  ", 321, 45545);
      cust[2] = new Customer(3, "Alexandr;  ", "Shirokov;  ", "Arsenyevich;  ", "Khimki;  ", 461, 12357);


      while (true) {

         System.out.println(
               "Выберете пункт меню:\n"
                     + "1. вывод покупателей в алфавитном порядке\n"
                     + "2. вывод покупателей с кредитной картой в интервале\n"
                     + ": "
         );
```

```
        try {
            int choice;
            choice = scanner.nextInt();
            if (choice < 1 || choice > 2)
                throw new InputMismatchException();


            if (choice == 0)
                break;
            if (choice < 1 || choice > 2) {
                System.out.println("выбран неправильный пункт меню, повторите ввод.");
                continue;
            }

            switch (choice) {
                case 1:
                    Arrays.sort(cust, Comparator.comparing(Customer::getSurname));
                    for (int i = 0; i <= 2; i++) {
                        System.out.println(cust[i]);
                    }
                    break;
                case 2:
                    long numto = 0, numfrom = 0;
                    System.out.println("Введите диапазон кредитных карт\n");
                    System.out.println("От: ");
                    int n1 = scanner.nextInt();
                    System.out.println("До: ");
                    int n2 = scanner.nextInt();
                    Customer.customer_check_credit_card(cust, n1, n2) ;
                    break;
                default:
                    throw new IllegalStateException("Unexpected value: " + choice);
            }
        } catch (InputMismatchException e) {
            System.out.println("Exception - " + e);
        }
    }
  }
}
```

```
Выберете пункт меню:
1. вывод покупателей в алфавитном порядке
2. вывод покупателей с кредитной картой в интервале
:
0
Exception - java.util.InputMismatchException
Выберете пункт меню:
1. вывод покупателей в алфавитном порядке
2. вывод покупателей с кредитной картой в интервале
:
```

Рисунок 3 – Результат работы программы

3. Patient: id, Фамилия, Имя, Отчество, Адрес, Телефон, Номер медицинской

карты, Диагноз. Создать массив объектов. Вывести: a) список пациентов, имеющих данный диагноз; b) список пациентов, номер медицинской карты у которых находится в заданном интервале.

Листинг 4 – Код программы

```
import java.io.IOException;

public class Patient {
    int patientId;
    String name;
    String surname;
    String patronymic;
    String address;
    long telephon;
    long numberMedCard;
    String diagnosis;

    public Patient() throws IOException {

    }

    public Patient(int patientId, String name, String surname, String patronymic, String address, long telephon, long
numberMedCard, String diagnosis) {
        this.patientId = patientId;
        this.name = name;
        this.surname = surname;
        this.patronymic = patronymic;
        this.address = address;
        this.telephon = telephon;
        this.numberMedCard = numberMedCard;
        this.diagnosis = diagnosis;
    }

    public int getPatientId() throws IOException{
        return patientId;
    }

    public void setPatientId(int patientId) throws IOException{
        this.patientId = patientId;
    }

    public String getName() throws IOException{
        return name;
    }

    public void setName(String name) throws IOException{
        this.name = name;
    }

    public String getSurname() throws IOException{
        return surname;
    }

    public void setSurname(String surname) throws IOException{
        this.surname = surname;
    }

    public String getPatronymic() throws IOException{
```

```
        return patronymic;
    }

    public void setPatronymic(String patronymic) throws IOException{
        this.patronymic = patronymic;
    }

    public String getAddress() throws IOException{
        return address;
    }

    public void setAddress(String address) throws IOException{
        this.address = address;
    }

    public long getTelephon() throws IOException{
        return telephon;
    }

    public void setTelephon(long telephon) throws IOException{
        this.telephon = telephon;
    }

    public long getNumberMedCard() {
        return numberMedCard;
    }

    public void setNumberMedCard(long numberMedCard) throws IOException{
        this.numberMedCard = numberMedCard;
    }

    public String getDiagnosis() {
        return diagnosis;
    }

    public void setDiagnosis(String diagnosis) throws IOException{
        this.diagnosis = diagnosis;
    }

    @Override
    public String toString() {
        return "Id пациента: " + patientId + ";  Имя: " + name + "Фамилия: " + surname +
                "Отчество: " + patronymic + "Адрес: " + address + "Телефон: " + telephon + ";  Номер мед карточки: "
+ numberMedCard +
                ";  Диагноз: " + diagnosis;
    }
    static void customer_check_med_card(Patient[] patients, int n, int b){
        for (Patient patient : patients) {
            if(patient.getNumberMedCard()>=n  &&  patient.getNumberMedCard()<=b)
                System.out.println(patient.toString());
        }
    }

    static void check_diagnosis(Patient[] patients,String diagnosis){
        for(Patient patient:patients){
            if(patient.getDiagnosis().equals(diagnosis)){
                System.out.println(patient.toString());
            }
```

```
        }
        System.out.println();
    }

}
```

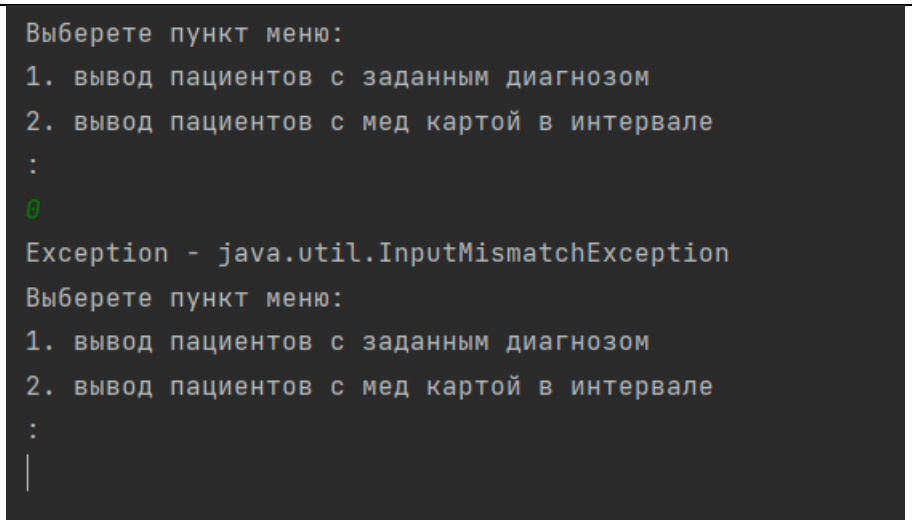


Рисунок 4 – Результат работы программы

Вариант 3

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

Листинг 5 – Код программы

```
import java.io.*;
import java.nio.charset.StandardCharsets;
import java.util.Scanner;

public class SubstringReplacer {
    public static void main(String[] args) {
        // Check if the correct number of command-line arguments are provided
        if (args.length != 3) {
            System.out.println("Usage: java SubstringReplacer input_file output_file substring_replacement");
            System.exit(1);
        }

        // Parse the command-line arguments
        String inputFile = args[0];
        String outputFile = args[1];
        String replacement = args[2];
        String str;
        try {
            str = new String("улица".getBytes(), "UTF-8");
        } catch (UnsupportedEncodingException e) {
            throw new RuntimeException(e);
        }


        // Open the input file
        Scanner scanner = null;
        try {
            scanner = new Scanner(new File(inputFile), StandardCharsets.UTF_8);
        } catch (FileNotFoundException e) {
            System.out.println("Error: Input file does not exist.");
```

```
            System.exit(1);
        } catch (IOException e) {
            throw new RuntimeException(e);
        }

        // Open the output file
        PrintWriter writer = null;
        try {
            writer = new PrintWriter(outputFile, StandardCharsets.UTF_8);
        } catch (FileNotFoundException e) {
            System.out.println("Error: Output file cannot be created.");
            System.exit(1);
        } catch (IOException e) {
            throw new RuntimeException(e);
        }

        // Read the input file line by line
        while (scanner.hasNextLine()) {
            String line = scanner.nextLine();

            // Replace the specified substring with the replacement string
            line = line.replace(str, replacement);

            // Write the modified line to the output file
            writer.println(line);
        }

        // Close the input and output files
        scanner.close();
        writer.close();
    }
}

// javac SubstringReplacer.java
// java SubstringReplacer input output_file ЗАМЕНА
```

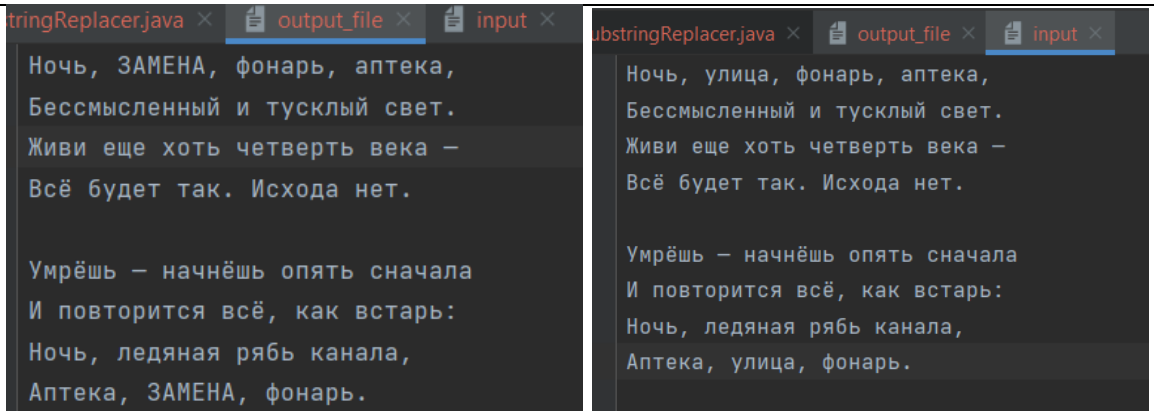


Рисунок 5 – Результат работы команды java SubstringReplacer input output_file ЗАМЕНА

3. В каждой строке найти слова, начинающиеся с гласной буквы.

Листинг 6 – Код программы

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
```

```
import java.io.IOException;

public class VowelWords {
   public static void main(String[] args) {
      String inputFilename = args[0];
      String outputFilename = args[1];

      try (BufferedReader reader = new BufferedReader(new FileReader(inputFilename));
           BufferedWriter writer = new BufferedWriter(new FileWriter(outputFilename))) {
         String line;
         while ((line = reader.readLine()) != null) {
            String[] words = line.split("\\s+"); // split the line into words
            for (String word : words) {
               if (word.matches("[AEIOUaeiou].*")) { // check if the word starts with a vowel
                  writer.write(word);
                  writer.write(" ");
               }
            }
            writer.newLine();
         }
      } catch (IOException e) {
         System.err.println("Error: " + e.getMessage());
      }
   }
}
```

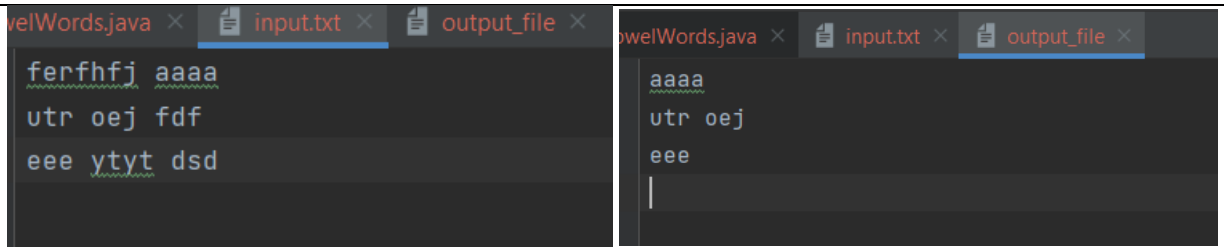


Рисунок 6 – Результат работы программы

Вариант 4

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

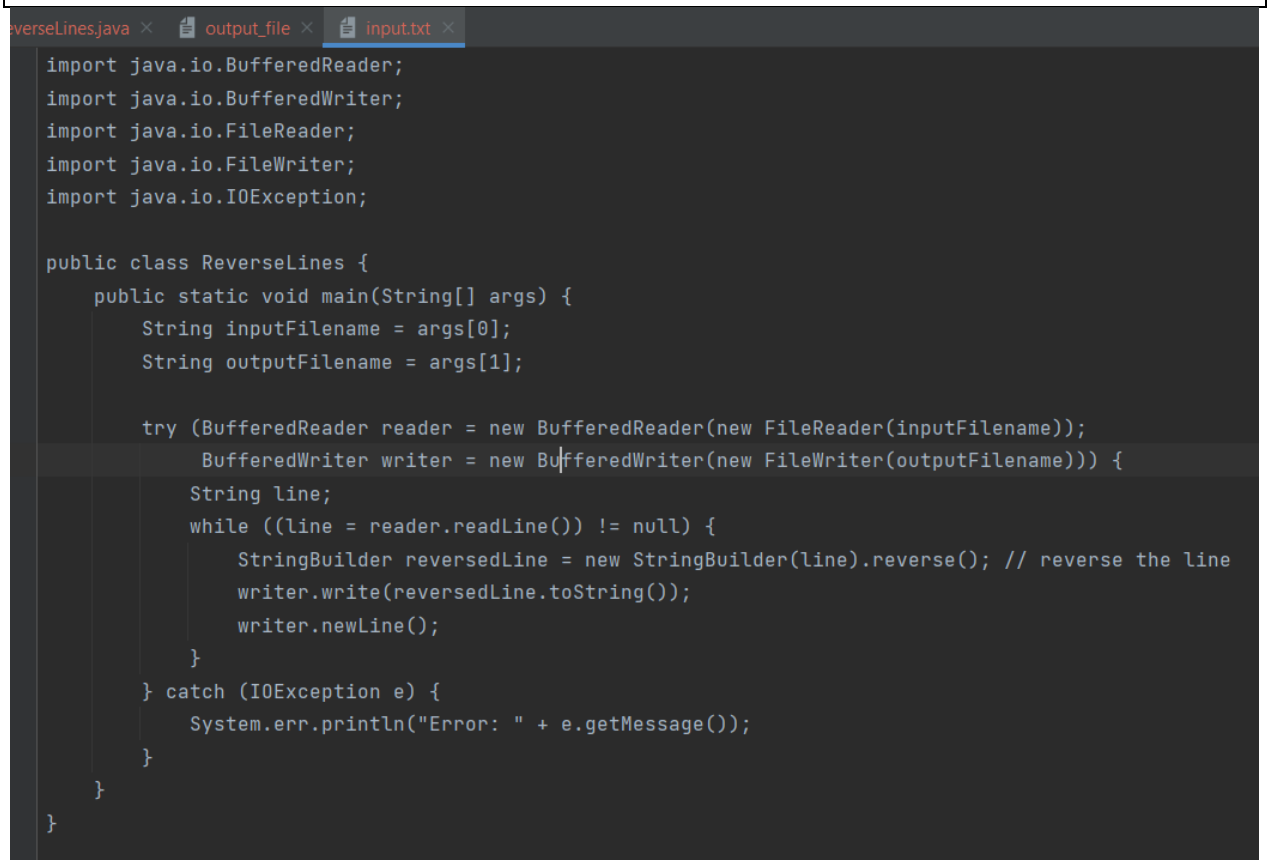
Листинг 7 – Код программы

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

public class ReverseLines {
   public static void main(String[] args) {
      String inputFilename = args[0];
      String outputFilename = args[1];

      try (BufferedReader reader = new BufferedReader(new FileReader(inputFilename));
           BufferedWriter writer = new BufferedWriter(new FileWriter(outputFilename))) {
         String line;
         while ((line = reader.readLine()) != null) {
            StringBuilder reversedLine = new StringBuilder(line).reverse(); // reverse the line
            writer.write(reversedLine.toString());
            writer.newLine();
         }
```

```
        } catch (IOException e) {
            System.err.println("Error: " + e.getMessage());
        }
    }
}
```



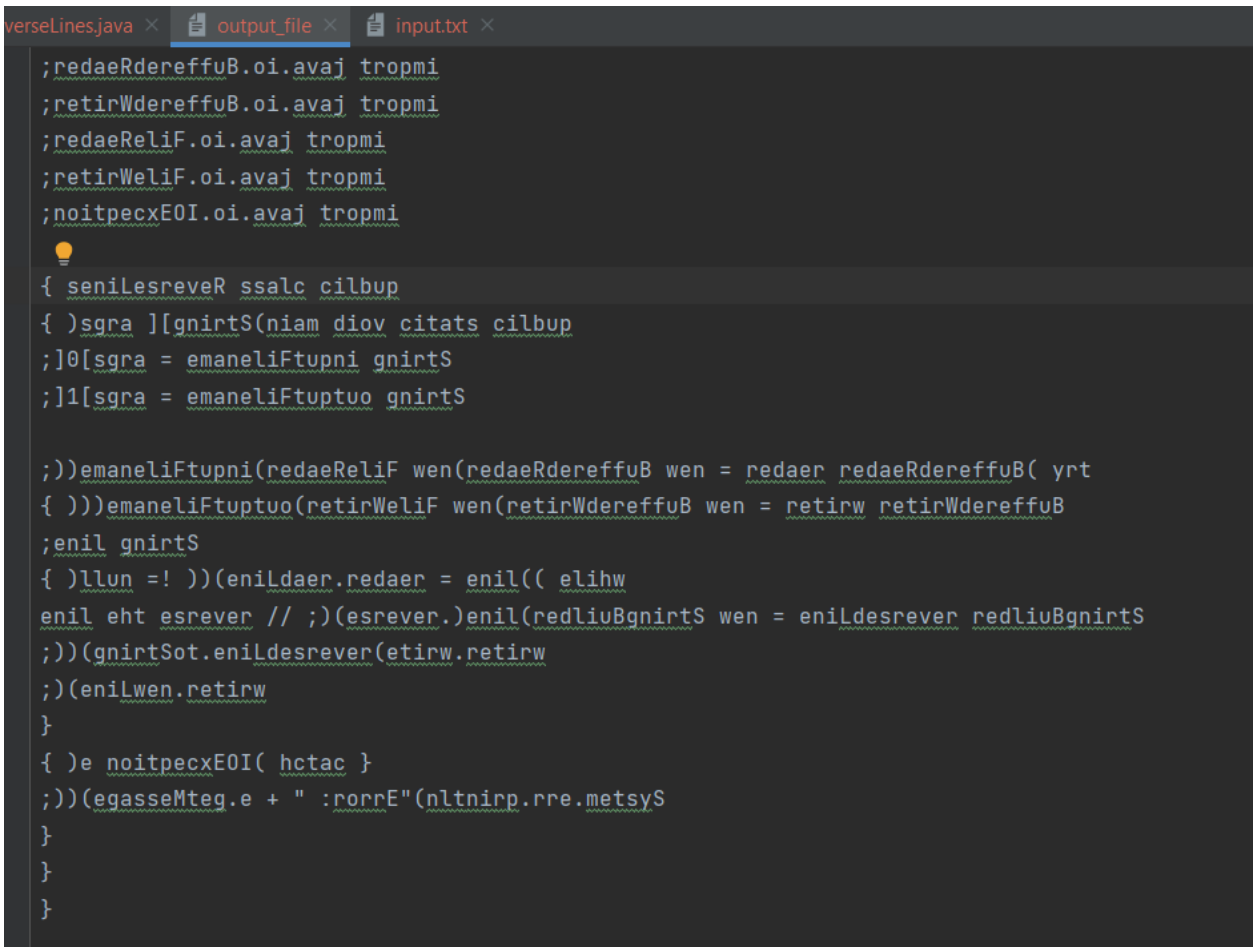


Рисунок 7 – Результат работы программы

3. Прочитать текст Java-программы и в каждом слове длиннее двух символов все строчные символы заменить прописными.

Листинг 8 – Код программы

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

public class UppercaseWords {
    public static void main(String[] args) {
        String inputFilename = args[0];
        String outputFilename = args[1];

        try (BufferedReader reader = new BufferedReader(new FileReader(inputFilename));
             BufferedWriter writer = new BufferedWriter(new FileWriter(outputFilename))) {
            String line;
            while ((line = reader.readLine()) != null) {
                String[] words = line.split("\\s+"); // split the line into words
                for (String word : words) {
                    if (word.length() > 2) { // check if the word is longer than two characters
                        String uppercaseWord = word.toUpperCase(); // convert the word to uppercase
                        writer.write(uppercaseWord);
                    } else {
                        writer.write(word);
                    }
```

```
                writer.write(" ");
            }
            writer.newLine();
        }
    } catch (IOException e) {
        System.err.println("Error: " + e.getMessage());
    }
  }
}
```

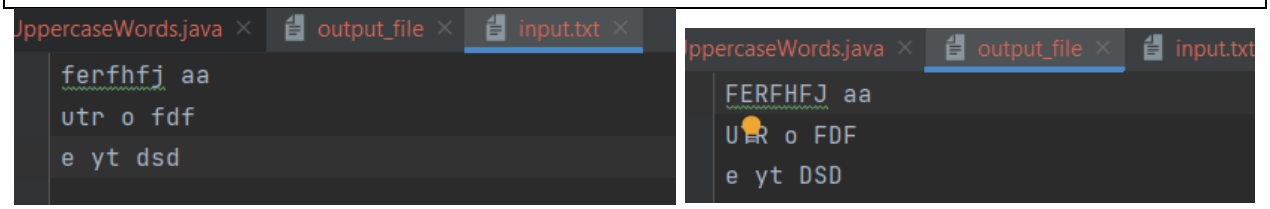


Рисунок 8 – Результат работы программы

Вывод: приобретен навык работы с исключениями и файлами.
Ссылка на репозиторий с программами: https://github.com/nargi3/BigData