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

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

Москва, 2023 г.

Цель: получить опыт работы с исключениями и файлами.

Вариант 1 – задание 9

Условие: определить класс Квадратное уравнение. Класс должен содержать несколько конструкторов. Реализовать методы для поиска корней, экстремумов, а также интервалов убывания/возрастания. Создать массив объектов и определить наибольшие и наименьшие по значению корни.

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.

Код:

```
import java.util.InputMismatchException;
import java.util.Scanner;

class QuadraticEquation {
    public final double a, b, c;
    public QuadraticEquation(double a, double b, double c) {
        if (a == 0)
            throw new IllegalArgumentException("Параметр a не
может быть равен 0");
        this.a = a;
        this.b = b;
        this.c = c;
    }
    public QuadraticEquation(double a, double b) {
        if (a == 0)
            throw new IllegalArgumentException("Параметр a не
может быть равен 0");
        this.a = a;
        this.b = b;
        this.c = 0;
    }
    public QuadraticEquation(double a) {
        if (a == 0)
            throw new IllegalArgumentException("Параметр a не
может быть равен 0");
        this.a = a;
        this.b = 0;
        this.c = 0;
    }
    public String toString() {
        return "(" + Double.toString(this.a) + ")x^2 + (" +
Double.toString(this.b) + ")x + (" +
                Double.toString(this.c) + ")";
    }
    private double getDiscriminant() {
        return b * b - 4 * a * c;
    }
    public double[] getRoots() {
        double discriminant = getDiscriminant();
        if (discriminant < 0) {
```

```
            return new double[0];
        } else if (discriminant == 0) {
            return new double[]{-b / (2 * a)};
        } else {
            double root1 = (-b + Math.sqrt(discriminant)) / (2 *
a);
            double root2 = (-b - Math.sqrt(discriminant)) / (2 *
a);
            return new double[]{root1, root2};
        }
    }

    public double getExtrema() {
        return -b / (2 * a);
    }

    public double[] getIncreasingInterval() {
        double extrema = getExtrema();
        if (a > 0) {
            return new double[]{extrema,
Double.POSITIVE_INFINITY};
        } else {
            return new double[]{Double.NEGATIVE_INFINITY,
extrema};
        }
    }

    public double[] getDecreasingInterval() {
        double extrema = getExtrema();
        if (a > 0) {
            return new double[]{Double.NEGATIVE_INFINITY,
extrema};
        } else {
            return new double[]{extrema,
Double.POSITIVE_INFINITY};
        }
    }

    public double getLargestRoot() {
        double[] roots = getRoots();
        if (roots.length == 0)
            return Double.NaN;
        else if (roots.length == 1)
            return roots[0];
        else
            return Math.max(roots[0], roots[1]);
    }

    // Method to get the smallest root of the quadratic equation
    public double getSmallestRoot() {
        double[] roots = getRoots();
        if (roots.length == 0)
            return Double.NaN;
```

```
        else if (roots.length == 1)
            return roots[0];
        else
            return Math.min(roots[0], roots[1]);
    }
}

public class var1_ex9 {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.print("Введите количество уравнений: ");
        int n;
        try {
            n = in.nextInt();
        } catch (InputMismatchException e) {
            System.out.println("Необходимо ввести число типа
int");
            return;
        }

        QuadraticEquation[] equations;
        try {
            equations = new QuadraticEquation[n];
        } catch (NegativeArraySizeException e) {
            System.out.println("Количество уравнений должно быть
положительным");
            return;
        }
        for (int i = 0; i < n; i++) {
            System.out.print("Введите параметры a, b, c для " +
(i + 1) + "-ого уравнения: ");
            double a, b, c;
            try {
                a = in.nextDouble();
                b = in.nextDouble();
                c = in.nextDouble();
            } catch (InputMismatchException e) {
                System.out.println("Необходимо ввести число типа
double");
                return;
            } catch (OutOfMemoryError e) {
                System.out.println("Нехватка памяти");
                return;
            }

            try {
                equations[i] = new QuadraticEquation(a, b, c);
            } catch (IllegalArgumentException e) {
                System.out.println(e.getMessage());
                return;
            }
        }
```

```
        for (QuadraticEquation equation : equations) {
            System.out.println("Уравнение " + equation + ":");
            System.out.println("Наименьший корень: " +
equation.getSmallestRoot());
            System.out.println("Наибольший корень: " +
equation.getLargestRoot());
            System.out.println();
        }
    }
}
```

Результат выполнения:

```
Введите количество уравнений: a
Необходимо ввести число типа int
Введите количество уравнений: 2
Введите параметры a, b, c для 1-ого уравнения: 1 2 в
Необходимо ввести число типа double
Введите количество уравнений: 1
Введите параметры a, b, c для 1-ого уравнения: 0 0 0
Параметр a не может быть равен 0
```

Рисунок 1 – Результат выполнения 1.9

Вариант 1 – задание 10

Условие: определить класс Булева матрица (BoolMatrix) размерности (n x m). Класс должен содержать несколько конструкторов. Реализовать методы для логического сложения (дизъюнкции), умножения и инверсии матриц. Реализовать методы для подсчета числа единиц в матрице и упорядочения строк в лексикографическом порядке.

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.

Код:

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

class BoolMatrix {
    public final boolean[][] matrix;
    private final int rows, cols;

    public BoolMatrix(boolean[][] matrix) {
        this.matrix = matrix;
        this.rows = matrix.length;
        this.cols = matrix[0].length;
    }

    public BoolMatrix(int rows, int cols) {
        this.matrix = new boolean[rows][cols];
        this.rows = rows;
```

```
        this.cols = cols;
    }

    public BoolMatrix disjunction(BoolMatrix other) {
        if (this.rows != other.rows || this.cols != other.cols)
{
            throw new IllegalArgumentException("Матрицы должны
быть одного размера");
        }
        BoolMatrix result = new BoolMatrix(this.rows,
this.cols);
        for (int i = 0; i < this.rows; i++) {
            for (int j = 0; j < this.cols; j++) {
                result.matrix[i][j] = this.matrix[i][j] |
other.matrix[i][j];
            }
        }
        return result;
    }

    public BoolMatrix multiplication(BoolMatrix other) {
        if (this.cols != other.rows) {
            throw new IllegalArgumentException("Матрицы должны
быть одного размера");
        }
        BoolMatrix result = new BoolMatrix(this.rows,
other.cols);
        for (int i = 0; i < this.rows; i++) {
            for (int j = 0; j < this.cols; j++) {
                result.matrix[i][j] = this.matrix[i][j] &
other.matrix[i][j];
            }
        }
        return result;
    }

    public BoolMatrix inversion() {
        BoolMatrix result = new BoolMatrix(this.cols,
this.rows);
        for (int i = 0; i < this.rows; i++) {
            for (int j = 0; j < this.cols; j++) {
                result.matrix[i][j] = !this.matrix[i][j];
            }
        }
        return result;
    }

    public int countOnes() {
        int count = 0;
        for (int i = 0; i < this.rows; i++) {
            for (int j = 0; j < this.cols; j++) {
                if (this.matrix[i][j]) {
                    count++;
```

```
                }
            }
        }
        return count;
    }

    public void orderRows() {
        Arrays.sort(matrix, new Comparator<boolean[]>() {
            @Override
            public int compare(boolean[] row1, boolean[] row2) {
                for (int i = 0; i < row1.length; i++) {
                    if (row1[i] != row2[i]) {
                        return row1[i] ? 1 : -1;
                    }
                }
                return 0;
            }
        });
    }

    public void print() {
        for (int i = 0; i < this.rows; i++) {
            for (int j = 0; j < this.cols; j++) {
                if (this.matrix[i][j])
                    System.out.print("1\t");
                else
                    System.out.print("0\t");
            }
            System.out.println();
        }
    }
}

public class var1_ex10 {
    private static BoolMatrix inputBoolMatrix() {
        Scanner in = new Scanner(System.in);
        System.out.print("Введите размер матрицы: ");
        int n;
        try {
            n = in.nextInt();
        } catch (InputMismatchException e) {
            throw new IllegalArgumentException("Необходимо
ввести число типа int");
        }

        if (n == 0)
            throw new IllegalArgumentException("Размер матрицы
должен быть положительным");

        boolean[][] matrix;
        try {
            matrix = new boolean[n][n];
        } catch (NegativeArraySizeException e) {
```

```
            throw new IllegalArgumentException("Размер матрицы
должен быть положительным");
        }

        System.out.println("Введите элементы матрицы (0/1):");
        for (int i = 0; i < n; i++)
            for (int j = 0; j < n; j++) {
                int a;
                try {
                    a = in.nextInt();
                } catch (InputMismatchException e) {
                    throw new
IllegalArgumentException("Необходимо ввести 0 или 1");
                }
                if (a == 0)
                    matrix[i][j] = false;
                else if (a == 1)
                    matrix[i][j] = true;
                else
                    throw new
IllegalArgumentException("Необходимо ввести 0 или 1");
            }

        return new BoolMatrix(matrix);
    }
    public static void main(String[] args) {
        BoolMatrix[] matrix = new BoolMatrix[2];
        for (int i = 0; i < 2; i++) {
            System.out.println((i + 1) + "-ая матрица:");
            try {
                matrix[i] = inputBoolMatrix();
            } catch (IllegalArgumentException e) {
                System.out.println(e.getMessage());
                return;
            }
        }

        BoolMatrix res_disjunction;
        try {
            res_disjunction = matrix[0].disjunction(matrix[1]);
        } catch (IllegalArgumentException e) {
            System.out.println(e.getMessage());
            return;
        }
        System.out.println("\nРезультат дизъюнкции:");
        res_disjunction.print();

        BoolMatrix res_multiplication;
        try {
            res_multiplication =
matrix[0].multiplication(matrix[1]);
        } catch (IllegalArgumentException e) {
            System.out.println(e.getMessage());
```

```
            return;
        }
        System.out.println("\nРезультат конъюнкции:");
        res_multiplication.print();

        BoolMatrix res_inversion = matrix[0].inversion();
        System.out.println("\nРезультат инверсии первой
матрицы:");
        res_inversion.print();

        System.out.println("\nКоличество единиц во второй
матрице = " + matrix[1].countOnes());

        matrix[0].orderRows();
        System.out.println("\nРезультат сортировки первой
матрицы:");
        matrix[0].print();
    }
}
```

Результат выполнения:

```
1-ая матрица:
Введите размер матрицы: a
Необходимо ввести число типа int
 1-ая матрица:
 Введите размер матрицы: 2
 Введите элементы матрицы (0/1):
 1 2
 Необходимо ввести 0 или 1
```

Рисунок 2 – Результат выполнения 1.10

Вариант 2 – задание 9

Условие: Product: id, Наименование, UPC, Производитель, Цена, Срок хранения, Количество. 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. Создать массив объектов. Вывести: a) список товаров для заданного наименования; b) список товаров для заданного наименования, цена которых не превосходит заданную; c) список товаров, срок хранения которых больше заданного.

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

Код:

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

class Product {
    private int id;
    private String name;
```

```
    private String upc;
    private String manufacturer;
    private double price;
    private int shelfLife;
    private int quantity;

    public Product(int id, String name, String upc, String
manufacturer, double price, int shelfLife, int quantity) {
        this.id = id;
        this.name = name;
        this.upc = upc;
        this.manufacturer = manufacturer;
        this.price = price;
        this.shelfLife = shelfLife;
        this.quantity = quantity;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

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

    public String getUpc() {
        return upc;
    }

    public void setUpc(String upc) {
        this.upc = upc;
    }

    public String getManufacturer() {
        return manufacturer;
    }

    public void setManufacturer(String manufacturer) {
        this.manufacturer = manufacturer;
    }

    public double getPrice() {
        return price;
    }
```

```
    public void setPrice(double price) {
        this.price = price;
    }

    public int getShelfLife() {
        return shelfLife;
    }

    public void setShelfLife(int shelfLife) {
        this.shelfLife = shelfLife;
    }

    public int getQuantity() {
        return quantity;
    }

    public void setQuantity(int quantity) {
        this.quantity = quantity;
    }

    @Override
    public String toString() {
        return "Product{" +
                "id=" + id +
                ", name='" + name + '\'' +
                ", upc='" + upc + '\'' +
                ", manufacturer='" + manufacturer + '\'' +
                ", price=" + price +
                ", shelfLife=" + shelfLife +
                ", quantity=" + quantity +
                '}';
    }
}
class ProductNotFoundException extends Exception {
    private final String message;
    public String getMessage() {
        return message;
    }
    public ProductNotFoundException(String[] param_names,
String[] param_values){
        super();
        StringBuilder sb = new StringBuilder("Продукты с ");
        for (int i = 0; i < param_names.length; i++)
            sb.append(param_names[i]).append("
'").append(param_values[i]).append("' и ");
        sb.delete(sb.length() - 3, sb.length()).append(" не
найдены");
        message = sb.toString();
    }
}
public class var2_ex9 {
    private static List<Product> getProductsByName(Product[]
products, String name) throws ProductNotFoundException {
```

```
        List<Product> res = new ArrayList<>();
        for (Product product : products)
            if (product.getName().equals(name))
                res.add(product);
        if (res.size() == 0)
            throw new ProductNotFoundException(new
String[]{"именем"}, new String[]{name});
        return res;
    }
    private static List<Product>
getProductsByNameAndPrice(Product[] products, String name,
double price) throws ProductNotFoundException {
        List<Product> res = new ArrayList<>();
        for (Product product : products)
            if (product.getName().equals(name) &&
product.getPrice() <= price)
                res.add(product);
        if (res.size() == 0)
            throw new ProductNotFoundException(new
String[]{"именем", "ценой <="}, new String[]{name,
Double.toString(price)});
        return res;
    }
    private static List<Product>
getProductsByShelfLife(Product[] products, int shelfLife) throws
ProductNotFoundException {
        List<Product> res = new ArrayList<>();
        for (Product product : products)
            if (product.getShelfLife() <= shelfLife)
                res.add(product);
        if (res.size() == 0)
            throw new ProductNotFoundException(new
String[]{"сроком годности <="}, new
String[]{Integer.toString(shelfLife)});
        return res;
    }
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        Product[] products = {
                new Product(1, "Продукт 1", "123456789",
"Производитель 1", 10.99, 30, 50),
                new Product(2, "Продукт 2", "987654321",
"Производитель 2", 5.99, 60, 100),
                new Product(3, "Продукт 3", "02468",
"Производитель 3", 20.99, 90, 20),
                new Product(2, "Продукт 2", "13579",
"Производитель 4", 6.99, 60, 2000)
        };
        System.out.print("Введите имя продукта: ");
        String name = in.nextLine();
        try {
            List<Product> productsByName =
getProductsByName(products, name);
```

```
            System.out.println("\nСписок продкутов с именем '" +
name + "':");
            for (Product product : productsByName)
                System.out.println(product);
        } catch (ProductNotFoundException e) {
            System.out.println(e.getMessage());
        }

        System.out.print("\nВведите цену продукта: ");
        double price;
        try {
            price = in.nextDouble();
        } catch (InputMismatchException e) {
            System.out.println("Необходимо ввести число типа
double");
            return;
        }
        try {
            List<Product> productsByNameAndPrice =
getProductsByNameAndPrice(products, name, price);
            System.out.println("\nСписок продкутов с именем '" +
name + "' и ценой <= " + price + ":");
            for (Product product : productsByNameAndPrice)
                System.out.println(product);
        } catch (ProductNotFoundException e) {
            System.out.println(e.getMessage());
        }

        System.out.print("\nВведите срок годности продукта: ");
        int shelfLife;
        try {
            shelfLife = in.nextInt();
        } catch (InputMismatchException e) {
            System.out.println("Необходимо ввести число типа
int");
            return;
        }
        try {
            List<Product> productsByShelfLife =
getProductsByShelfLife(products, shelfLife);
            System.out.println("\nСписок продкутов со сроком
годности <= " + shelfLife + ":");
            for (Product product : productsByShelfLife)
                System.out.println(product);
        } catch (ProductNotFoundException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

Результат выполнения:

```
Введите имя продукта: abc
Продукты с именем 'abc' не найдены

Введите цену продукта: d
Необходимо ввести число типа double
```

Рисунок 3 – Результат выполнения 2.9

Вариант 2 – задание 10

Условие: Train: Пункт назначения, Номер поезда, Время отправления, Число мест (общих, купе, плацкарт, люкс). 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. Создать массив объектов. Вывести: a) список поездов, следующих до заданного пункта назначения; b) список поездов, следующих до заданного пункта назначения и отправляющихся после заданного часа; c) список поездов, отправляющихся до заданного пункта назначения и имеющих общие места.

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

Код:

```
import java.sql.Time;
import java.util.ArrayList;
import java.util.InputMismatchException;
import java.util.List;
import java.util.Scanner;

class Train {
    private String destination;
    private int trainNumber;
    private Time departureTime;
    private int generalSeats;
    private int compartmentSeats;
    private int reservedSeats;
    private int suiteSeats;

    public Train(String destination, int trainNumber, Time
departureTime,
                 int generalSeats, int compartmentSeats, int
reservedSeats, int suiteSeats) {
        this.destination = destination;
        this.trainNumber = trainNumber;
        this.departureTime = departureTime;
        this.generalSeats = generalSeats;
        this.compartmentSeats = compartmentSeats;
        this.reservedSeats = reservedSeats;
        this.suiteSeats = suiteSeats;
    }

    public String getDestination() {
        return destination;
```

```
}

public void setDestination(String destination) {
    this.destination = destination;
}

public int getTrainNumber() {
    return trainNumber;
}

public void setTrainNumber(int trainNumber) {
    this.trainNumber = trainNumber;
}

public Time getDepartureTime() {
    return departureTime;
}

public void setDepartureTime(Time departureTime) {
    this.departureTime = departureTime;
}

public int getGeneralSeats() {
    return generalSeats;
}

public void setGeneralSeats(int generalSeats) {
    this.generalSeats = generalSeats;
}

public int getCompartmentSeats() {
    return compartmentSeats;
}

public void setCompartmentSeats(int compartmentSeats) {
    this.compartmentSeats = compartmentSeats;
}

public int getReservedSeats() {
    return reservedSeats;
}

public void setReservedSeats(int reservedSeats) {
    this.reservedSeats = reservedSeats;
}

public int getSuiteSeats() {
    return suiteSeats;
}

public void setSuiteSeats(int suiteSeats) {
    this.suiteSeats = suiteSeats;
}
```

```
    public String toString() {
        return "Train{" +
                "destination='" + destination + '\'' +
                ", trainNumber=" + trainNumber +
                ", departureTime='" + departureTime + '\'' +
                ", generalSeats=" + generalSeats +
                ", compartmentSeats=" + compartmentSeats +
                ", reservedSeats=" + reservedSeats +
                ", suiteSeats=" + suiteSeats +
                '}';
    }
}

class TrainNotFoundException extends Exception {
    private final String message;
    public String getMessage() {
        return message;
    }
    public TrainNotFoundException(String[] param_names, String[]
param_values){
        super();
        StringBuilder sb = new StringBuilder("Продукты с ");
        for (int i = 0; i < param_names.length; i++)
            sb.append(param_names[i]).append("
\"").append(param_values[i]).append("\" и ");
        sb.delete(sb.length() - 3, sb.length()).append(" не
найдены");
        message = sb.toString();
    }
}
public class var2_ex10 {
    private static List<Train> getTrainsByDestination(Train[]
trains, String destination) throws TrainNotFoundException {
        List<Train> res = new ArrayList<>();
        for (Train train : trains)
            if (train.getDestination().equals(destination))
                res.add(train);
        if (res.size() == 0)
            throw new TrainNotFoundException(new
String[]{"пунктом назначения"}, new String[]{destination});
        return res;
    }
    private static List<Train>
getTrainsByDestinationAndDepTime(Train[] trains, String
destination, Time depTime) throws TrainNotFoundException {
        List<Train> res = new ArrayList<>();
        for (Train train : trains)
            if (train.getDestination().equals(destination) &&
train.getDepartureTime().after(depTime))
                res.add(train);
        if (res.size() == 0)
            throw new TrainNotFoundException(new
```

```
String[]{"пунктом назначения", "временем отправления >"},
                    new String[]{destination,
depTime.toString()});
        return res;
    }
    private static List<Train>
getTrainsWithGenSeatsByDestination(Train[] trains, String
destination) throws TrainNotFoundException {
        List<Train> res = new ArrayList<>();
        for (Train train : trains)
            if (train.getDestination().equals(destination) &&
train.getGeneralSeats() > 0)
                res.add(train);
        if (res.size() == 0)
            throw new TrainNotFoundException(new
String[]{"пунктом назначения", "количеством общих мест >"},
                    new String[]{destination, "0"});
        return res;
    }
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        Train[] trains = {
                new Train("Париж", 1, new Time(10, 0, 0), 200,
100, 50, 20),
                new Train("Бердин", 2, new Time(12, 30, 0), 150,
80, 40, 10),
                new Train("Лондон", 3, new Time(14, 45, 0), 250,
120, 60, 30),
                new Train("Париж", 4, new Time(15, 0, 0), 0, 60,
30, 10)
        };
        System.out.print("Введите пункт назначения: ");
        String destination = in.nextLine();
        try {
            List<Train> trainsByDestination =
getTrainsByDestination(trains, destination);
            System.out.println("\nСписок поездов с пунктом
назначения '" + destination + "':");
            for (Train train : trainsByDestination)
                System.out.println(train);
        } catch (TrainNotFoundException e) {
            System.out.println(e.getMessage());
        }

        System.out.print("\nВведите час отправления: ");
        int hour;
        try {
            hour = in.nextInt();
        } catch (InputMismatchException e) {
            System.out.println("Необходимо ввести число типа
int");
            return;
        }
```

```
        System.out.print("Введите минуту отправления: ");
        int minute;
        try {
            minute = in.nextInt();
        } catch (InputMismatchException e) {
            System.out.println("Необходимо ввести число типа
int");
            return;
        }
        Time depTime = new Time(hour, minute, 0);
        try {
            List<Train> trainsByDestinationAndDepTime =
getTrainsByDestinationAndDepTime(trains, destination, depTime);
            System.out.println("\nСписок поездов с пунктом
назначения '" + destination + "' и временем отправления > " +
depTime + ":");
            for (Train train : trainsByDestinationAndDepTime)
                System.out.println(train);
        } catch (TrainNotFoundException e) {
            System.out.println(e.getMessage());
        }

        try {
            List<Train> trainsWithGenSeatsByDestination =
getTrainsWithGenSeatsByDestination(trains, destination);
            System.out.println("\nСписок поездов со свободными
общими местами и пунктом назначения '" + destination + "':");
            for (Train train : trainsWithGenSeatsByDestination)
                System.out.println(train);
        } catch (TrainNotFoundException e) {
            System.out.println("\n" + e.getMessage());
        }
    }
}
```

Результат выполнения:

```
Введите пункт назначения: abc
Продукты с пунктом назначения "abc" не найдены

Введите час отправления: d
Необходимо ввести число типа int
```

Рисунок 4 – Результат выполнения 2.10

Вариант 3 – задание 9

Условие: входной файл содержит совокупность строк. Строка файла содержит строку квадратной матрицы. Ввести матрицу в двумерный массив (размер матрицы найти). Вывести исходную матрицу и результат ее транспонирования.

Код:

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;
```

```
public class var3_ex9 {
    public static void main(String[] args) {
        String inputFilePath = "var3_ex9.txt";

        try {
            Scanner inputFile = new Scanner(new
File(inputFilePath));

            int matrixSize = 0;
            int[][] matrixData = null;
            int i = 0;

            while (inputFile.hasNextLine()) {
                String line = inputFile.nextLine();
                String[] elements = line.split(" ");
                if (matrixSize == 0) {
                    matrixSize = elements.length;
                    matrixData = new
int[matrixSize][matrixSize];
                }
                for (int j = 0; j < matrixSize; j++) {
                    matrixData[i][j] =
Integer.parseInt(elements[j]);
                }
                i++;
            }
            inputFile.close();
            System.out.println("Original Matrix:");
            printMatrix(matrixData);
            int[][] transposedMatrixData =
transposeMatrix(matrixData);
            System.out.println("Transposed Matrix:");
            printMatrix(transposedMatrixData);
        } catch (FileNotFoundException e) {
            System.out.println("Input file not found.");
        }
    }

    public static int[][] transposeMatrix(int[][] matrix) {
        int n = matrix.length;
        int[][] transposedMatrix = new int[n][n];

        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                transposedMatrix[i][j] = matrix[j][i];
            }
        }

        return transposedMatrix;
    }

    public static void printMatrix(int[][] matrix) {
        int n = matrix.length;
```

```
        for (int[] rows : matrix) {
            for (int elem : rows) {
                System.out.print(elem + " ");
            }
            System.out.println();
        }
    }
}
```

Результат выполнения:

```
Original Matrix:
1 2 3 4 5
6 7 8 9 0
1 2 3 4 5
6 7 8 9 0
1 2 3 4 5
Transposed Matrix:
1 6 1 6 1
2 7 2 7 2
3 8 3 8 3
4 9 4 9 4
5 0 5 0 5
```

Рисунок 5 – Результат выполнения 3.9

Вариант 3 – задание 10

Условие: Входной файл хранит квадратную матрицу по принципу: строка представляет собой число. Определить размерность. Построить 2-мерный массив, содержащий матрицу. Вывести исходную матрицу и результат ее поворота на 90 градусов по часовой стрелке.

Код:

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;

public class var3_ex10 {
    public static void main(String[] args) {
        String inputFilePath = "var3_ex10.txt";

        try {
            Scanner inputFile = new Scanner(new
File(inputFilePath));

            int matrixSize = 0;
            int[][] matrixData = null;
            int i = 0;

            while (inputFile.hasNextLine()) {
                String line = inputFile.nextLine();
                if (matrixSize == 0) {
```

```
                    matrixSize = line.length();
                    matrixData = new
int[matrixSize][matrixSize];
                }
                for (int j = 0; j < matrixSize; j++) {
                    matrixData[i][j] =
Integer.parseInt(String.valueOf(line.charAt(j)));
                }
                i++;
            }
            inputFile.close();
            System.out.println("Original Matrix:");
            printMatrix(matrixData);
            int[][] transposedMatrixData =
rotateMatrix(matrixData);
            System.out.println("Transposed Matrix:");
            printMatrix(transposedMatrixData);
        } catch (FileNotFoundException e) {
            System.out.println("Input file not found.");
        }
    }

    public static int[][] rotateMatrix(int[][] matrix) {
        int size = matrix.length;
        int[][] rotatedMatrix = new int[size][size];
        for (int i = 0; i < size; i++) {
            for (int j = 0; j < size; j++) {
                rotatedMatrix[j][size - 1 - i] = matrix[i][j];
            }
        }
        return rotatedMatrix;
    }

    public static void printMatrix(int[][] matrix) {
        int n = matrix.length;

        for (int[] rows : matrix) {
            for (int elem : rows) {
                System.out.print(elem + " ");
            }
            System.out.println();
        }
    }
}
```

Результат выполнения:

```
Original Matrix:
1 2 3 4
6 7 8 9
1 2 3 4
6 7 8 9
Transposed Matrix:
6 1 6 1
7 2 7 2
8 3 8 3
9 4 9 4
```

Рисунок 6 – Результат выполнения 3.10

Вариант 4 – задание 9

Условие: прочитать строки из файла и поменять местами первое и последнее слова в каждой строке.

Код:

```
import java.io.File;
import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.util.Scanner;

public class var4_ex9 {
    public static void main(String[] args) {
        File inputFile = new File("var4_ex9.txt");
        try {
            Scanner scanner = new Scanner(inputFile);
            StringBuilder output = new StringBuilder();
            while (scanner.hasNextLine()) {
                String line = scanner.nextLine();
                String[] words = line.split("\\s+");
                if (words.length > 1) {
                    String temp = words[0];
                    words[0] = words[words.length - 1];
                    words[words.length - 1] = temp;
                }
                String modifiedLine = String.join(" ", words);
                output.append(modifiedLine).append("\n");
            }
            scanner.close();

            File outputDir = new File("var4_ex9_output_dir");
            outputDir.mkdir();

            File outputFile = new File(outputDir,
"var4_ex9_output.txt");
            PrintWriter writer = new PrintWriter(outputFile);
            writer.print(output);
            writer.close();
            System.out.println("Output file written to: " +
outputFile.getAbsolutePath());
```

```
        } catch (FileNotFoundException e) {
            System.out.println("File not found: " +
inputFile.getName());
        }
    }
}
```

Результат выполнения:

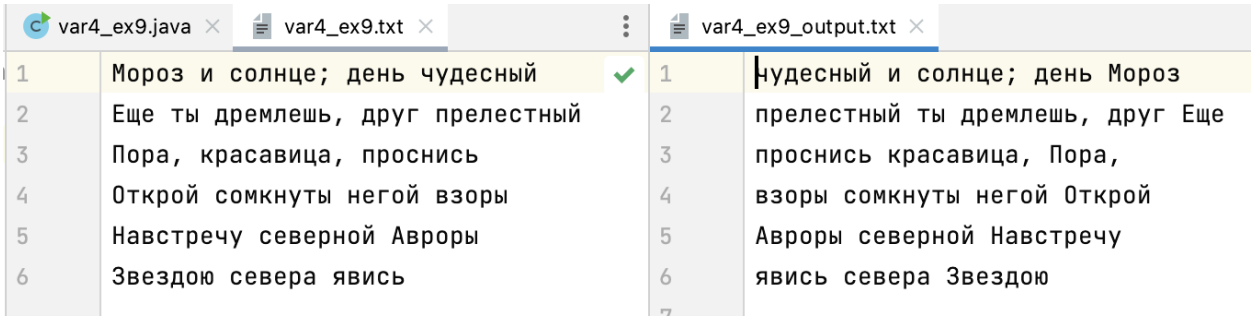

Рисунок 7 – Результат выполнения 4.9

Вариант 4 – задание 10

Условие: Ввести из текстового файла, связанного с входным потоком, последовательность строк. Выбрать и сохранить m последних слов в каждой из последних n строк

Код:

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Scanner;

public class var4_ex10 {

    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.print("Enter n: ");
        int n = in.nextInt();
        System.out.print("Enter m: ");
        int m = in.nextInt();
        System.out.println("Enter input file name:");
        String file_name = in.nextLine();
        File inputFile = new File("var4_ex9.txt");
        try {
            FileReader fileReader = new FileReader(inputFile);
            BufferedReader bufferedReader = new
BufferedReader(fileReader);
            String line;
            String[] lastWords = new String[n];
            int lineCount = 0;
            while ((line = bufferedReader.readLine()) != null) {
                if (lineCount < n) {
                    lastWords[lineCount] = getWordsFromEnd(line,
m);
```

```
                } else {
                    for (int i = 1; i < n; i++) {
                        lastWords[i-1] = lastWords[i];
                    }
                    lastWords[n-1] = getWordsFromEnd(line, m);
                }
                lineCount++;
            }
            bufferedReader.close();

            File outputDir = new File("var4_ex10_output_dir");
            outputDir.mkdir();

            File outputFile = new File(outputDir,
"var4_ex10_output.txt");
            FileWriter writer = new FileWriter(outputFile);
            for (String words : lastWords) {
                writer.write(words + "\n");
            }
            writer.close();
            System.out.println("Output file written to: " +
outputFile.getAbsolutePath());

        } catch (IOException e) {
            System.out.println("Error reading file: " +
e.getMessage());
        }
    }

    private static String getWordsFromEnd(String line, int m) {
        String[] words = line.split("\\s+");
        StringBuilder sb = new StringBuilder();
        for (int i = Math.max(0, words.length - m); i <
words.length; i++) {
            sb.append(words[i]).append(" ");
        }
        return sb.toString().trim();
    }
}
```

Результат выполнения:

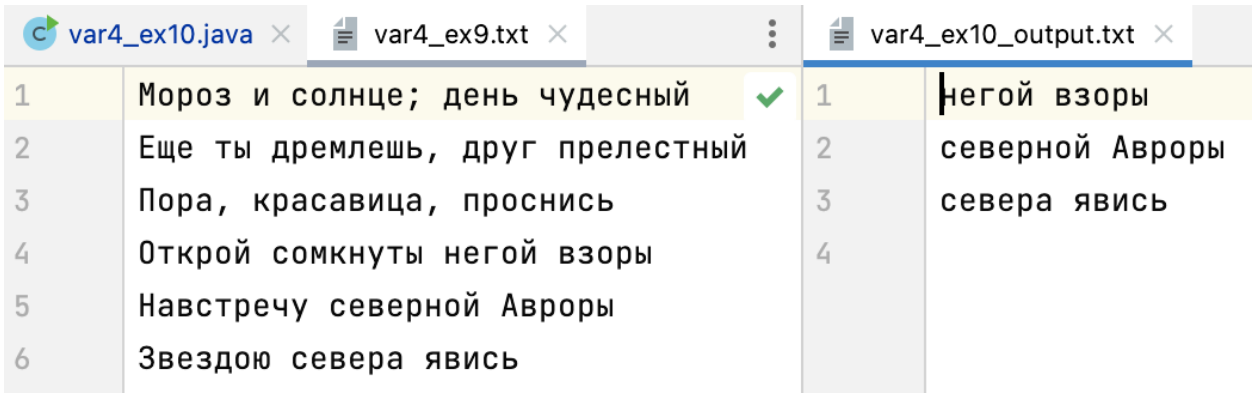

Рисунок 8 – Результат выполнения 4.10

Вывод: в ходе выполнения лабораторной работы был получен опыт работы с исключениями и файлами.