|  |  |
| --- | --- |
|  | **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**

**Отчет**

**по лабораторной работе № 5**

**Название:** Исключения. Файлы

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-22М |  |  | И. Д. Капкин |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | П.В. Степанов |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2023

1. ****Введение и задание****

**Целью лабораторной работы является формирование навыков работы с языком программирования java, изучение таких элементов как исключения и файлы.**

**Номер в списке – 10.**

**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.**

**Вариант 2.**

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

**Вариант 3.**

**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.**

**Задание 10.** 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.

**Задание 1.** В каждой строке найти и удалить заданную подстроку.

**Вариант 4.**

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

**Задание 10.** Ввести из текстового файла, связанного с входным потоком, последовательность строк. Выбрать и сохранить m последних слов в каждой из последних n строк**.**

**Задание 1.** Прочитать текст Java-программы и все слова public в объявлении атрибутов и методов класса заменить на слово private**.**

1. ****Ход работы****

**Выполнение задания №1 первого варианта лабораторной работы:**

**import static java.lang.Math.\*;**

**public class Main {**

**public static void main(String[] args) {**

**Vector [] array = {new Vector(1.2, 2.3, 3.4, 5.6),**

**new Vector(2.3, 4.5, 6.7, 8.9)};**

**System.out.println(Vector.dotProduct(array[0], array[1]));**

**System.out.println(array[0].magnitude());**

**System.out.println(array[1].magnitude());**

**}**

**}**

**class Vector {**

**private double [] coords;**

**public Vector(double... coords) {**

**if (coords.length == 0) throw new IllegalArgumentException("Cannot create a 0-dimentional vector");**

**this.coords = coords;**

**}**

**public double [] getCoords() {**

**try {**

**return this.coords;**

**} catch (Exception e) {**

**System.err.println("An error occurred while getting coords: " + e.getMessage());**

**return null;**

**}**

**}**

**public static Vector add(Vector a, Vector b) {**

**try {**

**double[] res = new double[min(a.getCoords().length, b.getCoords().length)];**

**for (int i = 0; i < res.length; i++) {**

**res[i] = ind(a, i) + ind(b, i);**

**}**

**return new Vector(res);**

**} catch (Exception e) {**

**System.err.println("An error occurred while adding the vectors: " + e.getMessage());**

**return null;**

**}**

**}**

**public static Vector substract(Vector a, Vector b) {**

**try {**

**double [] res = new double[min(a.getCoords().length, b.getCoords().length)];**

**for (int i = 0; i < res.length; i++) {**

**res[i] = ind(a, i) - ind(b, i);**

**}**

**return new Vector(res);**

**} catch (Exception e) {**

**System.err.println("An error occurred while subtracting the vectors: " + e.getMessage());**

**return null;**

**}**

**}**

**public static Vector multiply(Vector a, int n) {**

**try {**

**double [] res = new double[a.getCoords().length];**

**for (int i = 0; i < res.length; i++) {**

**res[i] = ind(a, i) \* n;**

**}**

**return new Vector(res);**

**} catch (Exception e) {**

**System.err.println("An error occurred while multiplying the vectors: " + e.getMessage());**

**return null;**

**}**

**}**

**public Vector incr() {**

**try {**

**for (int i = 0; i < this.coords.length; i++) {**

**this.coords[i]++;**

**}**

**return this;**

**} catch (Exception e) {**

**System.err.println("An error occurred while incrementing the vector: " + e.getMessage());**

**return null;**

**}**

**}**

**public Vector decr() {**

**try {**

**for (int i = 0; i < this.coords.length; i++) {**

**this.coords[i]--;**

**}**

**return this;**

**} catch (Exception e) {**

**System.err.println("An error occurred while decrementing the vector: " + e.getMessage());**

**return null;**

**}**

**}**

**public static double ind(Vector a, int index) {**

**try {**

**return a.getCoords()[index];**

**} catch (Exception e) {**

**System.err.println("An error occurred while indexing the vector: " + e.getMessage());**

**return 0;**

**}**

**}**

**public static double dotProduct(Vector a, Vector b) {**

**try {**

**double res = 0;**

**for (int i = 0; i < min(a.getCoords().length, b.getCoords().length); i++) {**

**res += ind(a, i) \* ind(b, i);**

**}**

**return res;**

**} catch (Exception e) {**

**System.err.println("An error occurred while calculating dot product: " + e.getMessage());**

**return 0;**

**}**

**}**

**public double magnitude() {**

**try {**

**double res = 0;**

**for (double coord : this.coords) {**

**res += pow(coord, 2);**

**}**

**return sqrt(res);**

**} catch (Exception e) {**

**System.err.println("An error occurred while calculating vector magnitude: " + e.getMessage());**

**return 0;**

**}**

**}**

**public static double angle(Vector a, Vector b) {**

**try {**

**return acos(dotProduct(a, b) / (a.magnitude() \* b.magnitude()));**

**} catch (Exception e) {**

**System.err.println("An error occurred while calculating the angle: " + e.getMessage());**

**return 0;**

**}**

**}**

**}**

**Выполнение задания №10 первого варианта лабораторной работы:**

**import java.util.Arrays;**

**import java.util.Comparator;**

**import static java.lang.Math.min;**

**public class Main {**

**public static void main(String[] args) {**

**boolean [][] test = {{true, false, true, true}, {false, false, true, false}, {true, false, false, false}};**

**BoolMatrix bm1 = new BoolMatrix(test);**

**BoolMatrix bm2 = BoolMatrix.inverse(bm1);**

**System.out.println(Arrays.deepToString(BoolMatrix.conjunct(bm1, bm2).getMatrix()));**

**System.out.println(Arrays.deepToString(BoolMatrix.disjunct(bm1, bm2).getMatrix()));**

**System.out.println(bm1.trueCount());**

**System.out.println(bm2.trueCount());**

**System.out.println(Arrays.deepToString(bm1.sorted().getMatrix()));**

**System.out.println(Arrays.deepToString(bm2.sorted().getMatrix()));**

**}**

**}**

**class BoolMatrix {**

**private boolean [][] matrix;**

**private int width;**

**private int height;**

**public BoolMatrix(BoolMatrix other) {**

**try {**

**this.matrix = other.getMatrix();**

**this.width = other.getWidth();**

**this.height = other.getHeight();**

**} catch (Exception e) {**

**System.err.println("An error occured while copying other BoolMatrix " + e.getMessage());**

**}**

**}**

**public BoolMatrix(boolean [][] other) {**

**if (Arrays.stream(other).anyMatch(row -> row.length != other[0].length)) {**

**throw new IllegalArgumentException("Source matrix is not a matrix");**

**}**

**this.matrix = other;**

**this.width = other[0].length;**

**this.height = other.length;**

**}**

**public boolean [][] getMatrix() {**

**try {**

**return this.matrix;**

**} catch (Exception e) {**

**System.err.println("An error occurred while accessing BoolMatrix " + e.getMessage());**

**return null;**

**}**

**}**

**public int getWidth() {**

**try {**

**return this.width;**

**} catch (Exception e) {**

**System.err.println("An error occurred while accessing BoolMatrix " + e.getMessage());**

**return 0;**

**}**

**}**

**public int getHeight() {**

**try {**

**return this.height;**

**} catch (Exception e) {**

**System.err.println("An error occurred while accessing BoolMatrix " + e.getMessage());**

**return 0;**

**}**

**}**

**public static BoolMatrix disjunct(BoolMatrix a, BoolMatrix b) {**

**try {**

**boolean[][] res = new boolean[min(a.height, b.height)][min(a.width, b.width)];**

**for (int i = 0; i < min(a.height, b.height); i++) {**

**for (int j = 0; j < min(a.width, b.width); j++) {**

**res[i][j] = a.getMatrix()[i][j] && b.getMatrix()[i][j];**

**}**

**}**

**return new BoolMatrix(res);**

**} catch (Exception e) {**

**System.err.println("An error occurred while disjunction " + e.getMessage());**

**return null;**

**}**

**}**

**public static BoolMatrix conjunct(BoolMatrix a, BoolMatrix b) {**

**try {**

**boolean[][] res = new boolean[min(a.height, b.height)][min(a.width, b.width)];**

**for (int i = 0; i < min(a.height, b.height); i++) {**

**for (int j = 0; j < min(a.width, b.width); j++) {**

**res[i][j] = a.getMatrix()[i][j] || b.getMatrix()[i][j];**

**}**

**}**

**return new BoolMatrix(res);**

**} catch (Exception e) {**

**System.err.println("An error occurred while conjunction " + e.getMessage());**

**return null;**

**}**

**}**

**public static BoolMatrix inverse(BoolMatrix a) {**

**try {**

**boolean[][] res = new boolean[a.height][a.width];**

**for (int i = 0; i < a.height; i++) {**

**for (int j = 0; j < a.width; j++) {**

**res[i][j] = !a.getMatrix()[i][j];**

**}**

**}**

**return new BoolMatrix(res);**

**} catch (Exception e) {**

**System.err.println("An error occurred while inversion " + e.getMessage());**

**return null;**

**}**

**}**

**public int trueCount() {**

**try {**

**int res = 0;**

**for (boolean[] row : this.getMatrix()) for (boolean col : row) if (col) res++;**

**return res;**

**} catch (Exception e) {**

**System.err.println("An error occurred while counting trues " + e.getMessage());**

**return 0;**

**}**

**}**

**public BoolMatrix sorted() {**

**try {**

**boolean[][] res = Arrays.copyOf(this.getMatrix(), this.height);**

**Arrays.sort(res, Comparator.comparingInt(o -> {**

**int a = 0;**

**for (boolean d : o) {**

**a = (a << 1) | (d ? 1 : 0);**

**}**

**return a;**

**}));**

**return new BoolMatrix(res);**

**} catch (Exception e) {**

**System.err.println("An error occurred while sorting " + e.getMessage());**

**return null;**

**}**

**}**

**}**

**Выполнение задания №10 второго варианта лабораторной работы:**

**import java.sql.Time;**

**import java.time.LocalDateTime;**

**import java.util.Calendar;**

**public class Main {**

**public static void main(String[] args) {**

**Train [] trains = {**

**new Train("Msk", 1337, LocalDateTime.of(2023, 02, 12, 22, 55),**

**4, 8, 15, 16),**

**new Train("Spb", 420, LocalDateTime.of(2023, 03, 01, 12, 30),**

**7, 3, 55, 6),**

**new Train("Msk", 69, LocalDateTime.of(2023, 4, 1, 7, 31),**

**0, 0, 0, 1)**

**};**

**System.out.println("Trains to Msk");**

**for (Train train : trains) {**

**if (train.getDestination() == "Msk") {**

**System.out.println(train.toString());**

**}**

**}**

**System.out.println("Trains to Msk after 2023-03-01 00:00");**

**for (Train train : trains) {**

**if (train.getDestination() == "Msk" && train.getDeparture().isAfter(LocalDateTime.of(2023, 3, 1, 0, 0))) {**

**System.out.println(train.toString());**

**}**

**}**

**System.out.println("Trains to Msk with general seats");**

**for (Train train : trains) {**

**if (train.getDestination() == "Msk" && train.getGeneralSeats() > 0) {**

**System.out.println(train.toString());**

**}**

**}**

**}**

**}**

**class Train {**

**private String destination;**

**private int trainNumber;**

**private LocalDateTime departure;**

**private int generalSeats;**

**private int coupeSeats;**

**private int plazkartSeats;**

**private int luxSeats;**

**public Train (String destination, int trainNumber, LocalDateTime departure,**

**int generalSeats, int coupeSeats, int plazkartSeats, int luxSeats) {**

**this.destination = destination;**

**this.trainNumber = trainNumber;**

**this.departure = departure;**

**this.generalSeats = generalSeats;**

**this.coupeSeats = coupeSeats;**

**this.plazkartSeats = plazkartSeats;**

**this.luxSeats = luxSeats;**

**}**

**public Train() {**

**}**

**public String getDestination() {**

**return this.destination;**

**}**

**public void setDestination(String destination) {**

**this.destination = destination;**

**}**

**public int getTrainNumber() {**

**return this.trainNumber;**

**}**

**public void setTrainNumber(int trainNumber) {**

**this.trainNumber = trainNumber;**

**}**

**public LocalDateTime getDeparture() {**

**return this.departure;**

**}**

**public void setDeparture(LocalDateTime departure) {**

**if (departure.isBefore(LocalDateTime.now()))**

**throw new IllegalArgumentException("Departure should be after before");**

**this.departure = departure;**

**}**

**public int getGeneralSeats() {**

**return this.generalSeats;**

**}**

**public void setGeneralSeats(int generalSeats) {**

**if (generalSeats < 0)**

**throw new IllegalArgumentException("Number of seats should be more than 0");**

**this.generalSeats = generalSeats;**

**}**

**public int getCoupeSeats() {**

**return this.coupeSeats;**

**}**

**public void setCoupeSeats(int coupeSeats) {**

**if (coupeSeats < 0)**

**throw new IllegalArgumentException("Number of seats should be more than 0");**

**this.coupeSeats = coupeSeats;**

**}**

**public int getPlazkartSeats() {**

**return this.plazkartSeats;**

**}**

**public void setPlazkartSeats(int plazkartSeats) {**

**if (plazkartSeats < 0)**

**throw new IllegalArgumentException("Number of seats should be more than 0");**

**this.plazkartSeats = plazkartSeats;**

**}**

**public int getLuxSeats() {**

**return this.luxSeats;**

**}**

**public void setLuxSeats(int luxSeats) {**

**if (luxSeats < 0)**

**throw new IllegalArgumentException("Number of seats should be more than 0");**

**this.luxSeats = luxSeats;**

**}**

**public String toString() {**

**return "Train #" + this.trainNumber +**

**"\tdestinated to " + this.destination +**

**"\tdepartures " + this.departure +**

**"\tgeneral seats " + this.generalSeats +**

**"\tcoupe seats " + this.coupeSeats +**

**"\tplazkart seats " + this.plazkartSeats +**

**"\tlux seats " + this.luxSeats;**

**}**

**}**

**Выполнение задания №1 второго варианта лабораторной работы:**

**import java.time.LocalDate;**

**import java.util.\*;**

**public class Main {**

**public static void main(String[] args) {**

**Student [] students = {**

**new Student(1, "Kapkin", "Ivan", "Dmitrievich", LocalDate.of(2001, 02, 12),**

**"Moscow", "+79859300594", "IU", 1, "22M"),**

**new Student(1, "Awoind", "Oawnd", "Hawdja", LocalDate.of(2000, 02, 12),**

**"Spb", "+79823420594", "SM", 3, "63"),**

**new Student(1, "Aawdawi", "Kajwnda", "Nawjdna", LocalDate.of(2001, 02, 12),**

**"Nsk", "+79523400594", "IU", 2, "42M"),**

**new Student(1, "Iaowidjao", "Jaawid", "Najwdnawo", LocalDate.of(2002, 02, 12),**

**"York", "+79823420594", "IBM", 1, "12B"),**

**new Student(1, "Nawdajn", "Adwoanon", "Kawjnajod", LocalDate.of(2000, 02, 12),**

**"Moscow", "+7234200594", "IU", 1, "22M")**

**};**

**System.out.println("IU students");**

**for (Student student : students) {**

**if (student.getFaculty() == "IU") {**

**System.out.println(student.toString());**

**}**

**}**

**System.out.println("Students for faculties and courses");**

**Map<String, List<Student>> mapFaculty = new HashMap<>();**

**Map<Integer, List<Student>> mapCourse = new HashMap<>();**

**for (Student student : students) {**

**if (!mapFaculty.containsKey(student.getFaculty())) {**

**mapFaculty.put(student.getFaculty(), new ArrayList<>());**

**}**

**mapFaculty.get(student.getFaculty()).add(student);**

**if (!mapCourse.containsKey(student.getCourse())) {**

**mapCourse.put(student.getCourse(), new ArrayList<>());**

**}**

**mapCourse.get(student.getCourse()).add(student);**

**}**

**for (Map.Entry<String, List<Student>> entry : mapFaculty.entrySet()) {**

**System.out.println(entry.getKey());**

**for (Student student : entry.getValue()) {**

**System.out.println(student.toString());**

**}**

**}**

**for (Map.Entry<Integer, List<Student>> entry : mapCourse.entrySet()) {**

**System.out.println(entry.getKey());**

**for (Student student : entry.getValue()) {**

**System.out.println(student.toString());**

**}**

**}**

**System.out.println("Students after 2001");**

**for (Student student : students) {**

**if (student.getBirth().getYear() >= 2001) {**

**System.out.println(student.toString());**

**}**

**}**

**System.out.println("Students of IU 1 course Group 22M");**

**for (Student student : students) {**

**if (student.getFacultyCourseGroup().equals("IU 1 22M")) {**

**System.out.println(student.toString());**

**}**

**}**

**}**

**}**

**class Student {**

**private int id;**

**private String surname;**

**private String name;**

**private String lastname;**

**private LocalDate birth;**

**private String address;**

**private String phone;**

**private String faculty;**

**private int course;**

**private String group;**

**Student (int id, String surname, String name, String lastname, LocalDate birth,**

**String address, String phone, String faculty, int course, String group) {**

**this.id = id;**

**this.surname = surname;**

**this.name = name;**

**this.lastname = lastname;**

**this.birth = birth;**

**this.address = address;**

**this.phone = phone;**

**this.faculty = faculty;**

**this.course = course;**

**this.group = group;**

**}**

**Student() {}**

**public int getId() {**

**return this.id;**

**}**

**public void setId(int id) {**

**this.id = id;**

**}**

**public String getFullName() {**

**return this.surname + " " + this.name + " " + this.lastname;**

**}**

**public void setFullName(String surname, String name, String lastname) {**

**if (!surname.matches("^[a-zA-Z\\\\-]+$"))**

**throw new IllegalArgumentException("Surname must contain only letters");**

**this.surname = surname;**

**if (!name.matches("^[a-zA-Z\\\\-]+$"))**

**throw new IllegalArgumentException("Name must contain only letters");**

**this.name = name;**

**if (!lastname.matches("^[a-zA-Z\\\\-]+$"))**

**throw new IllegalArgumentException("Lastname must contain only letters");**

**this.lastname = lastname;**

**}**

**public LocalDate getBirth() {**

**return this.birth;**

**}**

**public void setBirth(LocalDate birth){**

**if (birth.isAfter(LocalDate.now()))**

**throw new IllegalArgumentException("Date of birth should be before now");**

**this.birth = birth;**

**}**

**public String getAddress() {**

**return this.address;**

**}**

**public void setAddress(String address) {**

**this.address = address;**

**}**

**public String getPhone() {**

**return this.phone;**

**}**

**public void setPhone(String phone) {**

**if (!phone.matches("^\\+\\d{11}"))**

**throw new IllegalArgumentException("Wrong phone format");**

**this.phone = phone;**

**}**

**public String getFaculty() {**

**return this.faculty;**

**}**

**public int getCourse() {**

**return this.course;**

**}**

**public String getFacultyCourseGroup() {**

**return this.faculty + " " + this.course + " " + this.group;**

**}**

**public void setFacultyCourseGroup(String faculty, int course, String group) {**

**if (!Constants.FACULTIES.contains(faculty))**

**throw new IllegalArgumentException("There are no such faculty: " + faculty);**

**this.faculty = faculty;**

**if (course < 1 || course > 6)**

**throw new IllegalArgumentException("There are no such course: " + course);**

**this.course = course;**

**if (!group.matches("^\\d.\*"))**

**throw new IllegalArgumentException("Group name should start with a number");**

**this.group = group;**

**}**

**public String toString() {**

**return "id " + this.id +**

**"\tFull name: " + this.getFullName() +**

**"\tbirth: " + this.birth +**

**"\taddress: " + this.address +**

**"\tphone: " + this.phone +**

**"\tgroup: " + this.getFacultyCourseGroup();**

**}**

**}**

**class Constants {**

**public static final List<String> FACULTIES = Arrays.asList("IU", "IBM", "SM", "E", "MT", "BMT", "");**

**}**

1. ****Заключение****

**В процессе выполнения лабораторной работы были получены базовые навыки работы с языком программирования java, были изучены основные элементы – исключения и файлы.**