|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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 **09.04.01/07 Интеллектуальные системы анализа, обработки и интерпретации больших данных**

**Отчет**

|  |  |
| --- | --- |
| **по лабораторной работе №** | 5 |

**Название:** Исключения, файлы

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

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

Москва, 2024

**Цель работы:** освоить базовые принципы работы с исключениями и файлами на языке Java.

**Задание 1 вариант 4**

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 Матрица размерности (n x n). Класс должен содержать несколько конструкторов. Реализовать методы для сложения, вычитания, умножения матриц. Объявить массив объектов. Создать методы, вычисляющие первую и вторую нормы матрицы. Определить, какая из матриц имеет наименьшую первую и вторую нормы.

|  |
| --- |
| import java.util.InputMismatchException;  import java.util.Scanner;  public class Main {  public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  System.out.println("Укажите размер матриц:");  int size = 0;  try {  size = sc.nextInt();  sc.nextLine();  } catch (InputMismatchException e) {  System.out.println("Необходимо ввести число");  return;  }  System.out.println("Укажите количество матриц:");  int count = 0;  try {  count = sc.nextInt();  sc.nextLine();  } catch (InputMismatchException e) {  System.out.println("Необходимо ввести число");  return;  }  Matrix[] array = new Matrix[count];  for (int i = 0; i < count; i++) {  array[i] = new Matrix(size);  System.out.println("Получившаяся матрица " + i + ":\n" + array[i]);  }  System.out.println("1 и 2 нормы матриц:");  int min1Norm = array[0].norm1(), min2Norm = array[0].norm2();  int min1NormIndex = 0, min2NormIndex = 0;  for(int i = 0; i < count; i++) {  int norm1 = array[i].norm1(), norm2 = array[i].norm2();  System.out.println(i + " матрица: 1 норма = " + norm1 + ", 2 норма = " + norm2);  if (norm1 < min1Norm) {  min1Norm = norm1;  min1NormIndex = i;  }  if (norm2 < min2Norm) {  min2Norm = norm2;  min2NormIndex = i;  }  }  System.out.println("\nНаименьшую 1 норму имеет матрица " + min1NormIndex + ":\n" + array[min1NormIndex]);  System.out.println("Наименьшую 2 норму имеет матрица " + min2NormIndex + ":\n" + array[min2NormIndex]);  try {  System.out.println("Сложение первых двух матриц:\n" + array[0].add(array[1]));  System.out.println("Вычитание первых двух матриц:\n" + array[0].sub(array[1]));  System.out.println("Умножение первых двух матриц:\n" + array[0].mul(array[1]));  } catch (MatrixException e) {  System.out.println("Возникла ошибка: " + e);  } catch (ArrayIndexOutOfBoundsException e) {  System.out.println("Указано меньше чем 2 матрицы");  }  }  } |

|  |
| --- |
| import java.util.Random;  public class Matrix {  private final int[][] data;  private final int size;  public Matrix(int[][] data) {  this.data = data;  this.size = data.length;  }  public Matrix(int size) {  this.size = size;  this.data = new int[size][size];  Random r = new Random();  for(int i = 0; i < size; i++) {  for(int j = 0; j < size; j++) {  this.data[i][j]=r.nextInt() % 100;  }  }  }  public String toString() {  StringBuilder result = new StringBuilder();  for(int[] arr: this.data) {  for (int el : arr) {  result.append(el).append("\t");  }  result.append("\n");  }  return String.valueOf(result);  }  public int[][] getData(){  return this.data;  }  private void checkSize(int[][] matrix) throws MatrixException {  if (this.size != matrix.length) {  throw new MatrixException("Матрицы должны иметь одинаковую размерность");  }  }  public Matrix add(Matrix matrix) throws MatrixException {  this.checkSize(matrix.getData());  int[][] result = new int[this.size][this.size];  for(int i = 0; i < size; i++) {  for(int j = 0; j < size; j++) {  result[i][j] = this.data[i][j] + matrix.getData()[i][j];  }  }  return new Matrix(result);  }  public Matrix sub(Matrix matrix) throws MatrixException {  this.checkSize(matrix.getData());  int[][] result = new int[this.size][this.size];  for(int i = 0; i < size; i++) {  for(int j = 0; j < size; j++) {  result[i][j] = this.data[i][j] - matrix.getData()[i][j];  }  }  return new Matrix(result);  }  public Matrix mul(Matrix matrix) throws MatrixException {  this.checkSize(matrix.getData());  int[][] result = new int[this.size][this.size];  for(int l = 0; l < size; l++) {  for(int i = 0; i < size; i++) {  for(int j = 0; j < size; j++) {  result[l][i] += this.data[l][j] \* matrix.getData()[j][i];  }  }  }  return new Matrix(result);  }  public int norm1() {  Integer max\_sum = null;  for(int[] arr: this.data) {  int sum = 0;  for(int el: arr) {  sum += Math.abs(el);  }  if ((max\_sum == null) || (sum > max\_sum)) {  max\_sum = sum;  }  }  return max\_sum;  }  public int norm2() {  Integer max\_sum = null;  for(int i = 0; i < size; i++) {  int sum = 0;  for(int j = 0; j < size; j++) {  sum += Math.abs(this.data[j][i]);  }  if ((max\_sum == null) || (sum > max\_sum)) {  max\_sum = sum;  }  }  return max\_sum;  }  }  class MatrixException extends Exception {  public MatrixException(String message) {  super(message);  }  } |

**Задание 1 вариант 5**

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 Матрица размерности (m x n). Класс должен содержать несколько конструкторов. Объявить массив объектов. Передать объекты в метод, меняющий местами строки с максимальным и минимальным элементами k-го столбца. Создать метод, который изменяет i-ю матрицу путем возведения ее в квадрат.

|  |
| --- |
| import java.util.InputMismatchException;  import java.util.Scanner;  public class Main {  public static void main(String[] args) throws Exception {  Scanner sc = new Scanner(System.in);  int m, n, count;  try {  System.out.println("Укажите количество строк в матрицах:");  m = sc.nextInt();  System.out.println("Укажите количество столбцов в матрицах:");  n = sc.nextInt();  System.out.println("Укажите количество матриц:");  count = sc.nextInt();  } catch (InputMismatchException e) {  System.out.println("Необходимо ввести число");  return;  }  Matrix[] array = new Matrix[count];  for (int i = 0; i < count; i++) {  array[i] = new Matrix(m, n);  System.out.println("Получившаяся матрица " + i + ":\n" + array[i]);  }  System.out.println("Укажите относительно какого столбца менять строки:");  int k;  try {  k = sc.nextInt();  } catch (InputMismatchException e) {  System.out.println("Необходимо ввести число");  return;  }  if (k > n - 1) {  throw new Exception("Данного столбца не существует.");  }  array = swapRowsInMatrix(array, k);  System.out.println("Объекты с поменяными местами строками с максимальным и минимальным элементами " + k + "-го столбца:");  printArray(array);  System.out.println("Укажите какую матрицу надо возвести в квадрат:");  int i;  try {  i = sc.nextInt();  } catch (InputMismatchException e) {  System.out.println("Необходимо ввести число");  return;  }  if (i > count - 1) {  throw new Exception("Данной матрицы не существует.");  }  try {  array = squaringCertainMatrix(array, i);  } catch (Exception e) {  throw new RuntimeException(e);  }  System.out.println("Объекты с возведенной в квадрат " + i + " матрицей:");  printArray(array);  }  public static Matrix[] swapRowsInMatrix(Matrix[] array, int k) {  Matrix[] result = new Matrix[array.length];  for (int i = 0; i < array.length; i++) {  result[i] = array[i].swapRowsInMatrix(k);  }  return result;  }  public static Matrix[] squaringCertainMatrix(Matrix[] array, int i) throws Exception {  Matrix[] result = new Matrix[array.length];  for (int j = 0; j < array.length; j++) {  if (j == i) {  result[j] = array[j].squaring();  continue;  }  result[j] = array[j];  }  return result;  }  public static void printArray(Matrix[] array) {  for (Matrix matrix: array){  System.out.println(matrix);  }  }  } |

|  |
| --- |
| import java.util.Random;  public class Matrix {  private final int[][] data;  private final int m;  private final int n;  public Matrix(int[][] data) {  this.data = data;  this.m = data.length;  this.n = data[0].length;  }  public Matrix(int m, int n) {  this.m = m;  this.n = n;  this.data = new int[m][n];  Random r = new Random();  for(int i = 0; i < m; i++) {  for(int j = 0; j < n; j++) {  this.data[i][j]=r.nextInt() % 100;  }  }  }  public String toString() {  StringBuilder result = new StringBuilder();  for(int[] arr: this.data) {  for (int el : arr) {  result.append(el).append("\t\t");  }  result.append("\n");  }  return String.valueOf(result);  }  public Matrix squaring() throws Exception {  if (this.m != this.n) {  throw new Exception("Возводить в квадрат можно только квадратные матрицы.");  }  int[][] result = new int[this.m][this.n];  for(int l = 0; l < m; l++) {  for(int i = 0; i < n; i++) {  for(int j = 0; j < n; j++) {  result[l][i] += this.data[l][j] \* this.data[j][i];  }  }  }  return new Matrix(result);  }  public Matrix swapRowsInMatrix(int k) {  int[][] result = new int[this.m][this.n];  int min = this.data[0][k], max = this.data[0][k];  int indexMin = 0, indexMax = 0;  for (int i = 0; i < this.m; i++){  if (this.data[i][k] < min) {  min = this.data[i][k];  indexMin = i;  }  if (this.data[i][k] > max) {  max = this.data[i][k];  indexMax = i;  }  }  for (int i = 0; i < this.m; i++){  for (int j = 0; j < this.n; j++) {  if (i == indexMin) {  result[i][j] = this.data[indexMax][j];  continue;  }  if (i == indexMax) {  result[i][j] = this.data[indexMin][j];  continue;  }  result[i][j] = this.data[i][j];  }  }  return new Matrix(result);  }  } |

**Задание 2 вариант 4**

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. Abiturient: id, Фамилия, Имя, Отчество, Адрес, Телефон, Оценки. Создать массив объектов. Вывести: a) список абитуриентов, имеющих неудовлетворительные оценки; b) список абитуриентов, средний балл у которых выше заданного; c) выбрать заданное число n абитуриентов, имеющих самый высокий средний балл (вывести также полный список абитуриентов, имеющих полупроходной балл).

|  |
| --- |
| import java.util.Arrays;  import java.util.InputMismatchException;  import java.util.Scanner;  public class Main {  public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  System.out.println("Укажите количество студентов:");  int count = 0;  try {  count = sc.nextInt();  sc.nextLine();  } catch (InputMismatchException e) {  System.out.println("Необходимо ввести число");  return;  }  Student[] students = new Student[count];  for (int i = 0; i < count; i++) {  try {  System.out.println("Введите данные для студента " + (i + 1) + ": ");  System.out.print("Укажите ФИО: ");  String[] fio = sc.nextLine().split(" ");  if (fio.length != 3) {  throw new InputMismatchException("ФИО должно состоять из 3 слов");  }  System.out.print("Укажите адрес: ");  String address = sc.nextLine();  System.out.print("Укажите телефон: ");  String phone = sc.nextLine();  if (!phone.matches("\\d+")) {  throw new InputMismatchException("Телефон должен состоять только из цифр");  }  System.out.print("Укажите оценки: ");  String[] marks = sc.nextLine().split(" ");  int[] marksArray = new int[marks.length];  try {  marksArray = Arrays.stream(marks).mapToInt(Integer::parseInt).toArray();  } catch (NumberFormatException e) {  throw new InputMismatchException("Оценки должны быть целыми числами");  }  students[i] = new Student(i + 1, fio[0], fio[1], fio[2], address, phone, marksArray);  } catch (InputMismatchException e) {  System.out.println("Данные для " + (i + 1) + " студента введены неверно: " + e);  return;  }  }  Collection collection = new Collection(students);  System.out.println("Список абитуриентов, имеющих неудовлетворительные оценки:");  System.out.println(collection.getStudentsWithBadMarks());  System.out.println("Укажите средний балл:");  int averageMark = sc.nextInt();  System.out.println("Список абитуриентов, средний балл у которых выше " + averageMark + ":");  try {  System.out.println(collection.getStudentsWithAverageMarkAboveGiven(averageMark));  } catch (StudentException e) {  System.out.println("Произошла ошибка: " + e);  }  System.out.println("Укажите количество абитуриентов:");  int n = sc.nextInt();  System.out.println(n + " абитуриентов, имеющих самый высокий средний балл:");  try {  System.out.println(collection.getStudentsWithMaxAverageMark(n));  } catch (StudentException e) {  System.out.println("Произошла ошибка: " + e);  }  System.out.println("Абитуриентов, имеющие полупроходной балл:");  try {  System.out.println(collection.getStudentsWithSemiPassingMark(n));  } catch (StudentException e) {  System.out.println("Произошла ошибка: " + e);  }  }  } |

|  |
| --- |
| import java.util.Arrays;  import java.util.Comparator;  import java.util.stream.Stream;  public class Collection {  private final Student[] students;  public Collection(Student[] students) {  this.students = students;  }  public String toString() {  StringBuilder result = new StringBuilder();  for(Student student: this.students) {  result.append(student);  }  return String.valueOf(result);  }  public Collection getStudentsWithBadMarks() {  Stream<Student> result = Arrays.stream(this.students).filter(Student::hasBadMarks);  return new Collection(result.toArray(Student[]::new));  }  public Collection getStudentsWithAverageMarkAboveGiven(float mark) throws StudentException {  if (mark < 0 || mark > 10) {  throw new StudentException("Оценка должна быть больше 0 и меньше или равно 10");  }  Stream<Student> result = Arrays.stream(this.students).filter((s) -> s.getAverageMark() > mark);  return new Collection(result.toArray(Student[]::new));  }  public Collection getStudentsWithMaxAverageMark(int count) throws StudentException {  Stream<Student> result = Arrays.stream(this.students).sorted(Comparator.comparing(Student::getAverageMark).reversed()).limit(count);  if (result.toArray(Student[]::new).length < count) {  throw new StudentException("Студентов меньше заявленного числа");  }  return new Collection(result.toArray(Student[]::new));  }  public Collection getStudentsWithSemiPassingMark(int count) throws StudentException {  Student[] sortedStudents = Arrays.stream(this.students).sorted(  Comparator.comparing(Student::getAverageMark).reversed()  ).toArray(Student[]::new);  try {  float semiPassingMark = sortedStudents[count].getAverageMark();  Stream<Student> result = Arrays.stream(sortedStudents).filter((s) -> s.getAverageMark() == semiPassingMark);  return new Collection(result.toArray(Student[]::new));  } catch (ArrayIndexOutOfBoundsException e) {  throw new StudentException("Указано слишком большое число");  }  }  } |

|  |
| --- |
| import java.util.Arrays;  public class Student {  private int id;  private String lastName;  private String firstName;  private String patronymic;  private String address;  private String phone;  private int[] marks;  public Student() {}  public Student(int id, String lastName, String firstName, String patronymic, String address, String phone, int[] marks) {  this.id = id;  this.lastName = lastName;  this.firstName = firstName;  this.patronymic = patronymic;  this.address = address;  this.phone = phone;  this.marks = marks;  }  public String toString() {  return String.format(  "Student(%d) %s %s %s.\tAddress: %s.\tPhone: %s.\tMarks: %s\n",  this.id, this.lastName, this.firstName, this.patronymic, this.address, this.phone,  Arrays.toString(this.marks)  );  }  public void setId(int id) {  this.id = id;  }  public int getId() {  return this.id;  }  public void setLastName(String lastName) {  this.lastName = lastName;  }  public String getLastName() {  return this.lastName;  }  public void setFirstName(String firstName) {  this.firstName = firstName;  }  public String getFirstName() {  return this.firstName;  }  public void setPatronymic(String patronymic) {  this.patronymic = patronymic;  }  public String getPatronymic() {  return this.patronymic;  }  public void setAddress(String address) {  this.address = address;  }  public String getAddress() {  return this.address;  }  public void setPhone(String phone) {  this.phone = phone;  }  public String getPhone() {  return this.phone;  }  public void setMarks(int[] marks) {  this.marks = marks;  }  public int[] getMarks() {  return this.marks;  }  public boolean hasBadMarks(){  for(int mark: this.marks) {  if (mark < 4) {  return true;  }  }  return false;  }  public float getAverageMark() {  return (float) Arrays.stream(this.marks).sum() / this.marks.length;  }  }  class StudentException extends Exception {  public StudentException(String message) {  super(message);  }  } |

**Задание 2 вариант 5**

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. Book: id, Название, Автор(ы), Издательство, Год издания, Количество страниц, Цена, Переплет. Создать массив объектов. Вывести: a) список книг заданного автора; b) список книг, выпущенных заданным издательством; c) список книг, выпущенных после заданного года.

|  |
| --- |
| import java.util.InputMismatchException;  import java.util.Scanner;  public class Main {  public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  System.out.println("Укажите количество книг:");  int count = 0;  try {  count = sc.nextInt();  sc.nextLine();  } catch (InputMismatchException e) {  System.out.println("Необходимо ввести число");  return;  }  Book[] books = new Book[count];  for (int i = 0; i < count; i++) {  try {  System.out.println("Введите данные для книги " + (i + 1) + ": ");  System.out.print("Укажите название: ");  String name = sc.nextLine();  System.out.print("Укажите авторов: ");  String[] authors = sc.nextLine().split(" ");  System.out.print("Укажите издательство: ");  String publishingHouse = sc.nextLine();  System.out.print("Укажите год издания: ");  int year = 0;  try {  year = sc.nextInt();  sc.nextLine();  } catch (InputMismatchException e) {  throw new InputMismatchException("Год должен быть указан числом");  }  System.out.print("Укажите количество страниц: ");  int numberOfPages = 0;  try {  numberOfPages = sc.nextInt();  sc.nextLine();  } catch (InputMismatchException e) {  throw new InputMismatchException("Количество страниц должно быть указано числом");  }  System.out.print("Укажите цену: ");  float price = 0;  try {  price = sc.nextFloat();  sc.nextLine();  } catch (InputMismatchException e) {  throw new InputMismatchException("Цена должна быть числом с плавающей точкой");  }  System.out.print("Укажите переплет: ");  String binding = sc.nextLine();  books[i] = new Book(i + 1, name, authors, publishingHouse, year, numberOfPages, price, binding);  } catch (InputMismatchException e) {  System.out.println("Данные для " + (i + 1) + " книги введены неверно: " + e);  return;  }  }  Collection collection = new Collection(books);  System.out.println("Укажите автора:");  String author = sc.next();  System.out.println("Список книг автора " + author + ":");  System.out.println(collection.getBooksByAuthor(author));  System.out.println("Укажите издательство:");  String publishingHouse = sc.next();  System.out.println("Список книг, выпущенных " + publishingHouse + ":");  System.out.println(collection.getBooksPublishedByGivenPublisher(publishingHouse));  System.out.println("Укажите год:");  int year = sc.nextInt();  System.out.println("Список книг, выпущенных после " + year + " года:");  try {  System.out.println(collection.getBooksPublishedAfterGivenYear(year));  } catch (BookException e) {  System.out.println("Произошла ошибка: " + e);  }  }  } |

|  |
| --- |
| import java.util.Arrays;  import java.util.Objects;  import java.util.stream.Stream;  public class Collection {  private final Book[] books;  public Collection(Book[] books) {  this.books = books;  }  public String toString() {  StringBuilder result = new StringBuilder();  for(Book book: this.books) {  result.append(book);  }  return String.valueOf(result);  }  public Collection getBooksByAuthor(String author) {  Stream<Book> result = Arrays.stream(this.books).filter((b) -> b.belongsToAuthor(author));  return new Collection(result.toArray(Book[]::new));  }  public Collection getBooksPublishedByGivenPublisher(String publishingHouse) {  Stream<Book> result = Arrays.stream(this.books).filter((b) -> Objects.equals(b.getPublishingHouse(), publishingHouse));  return new Collection(result.toArray(Book[]::new));  }  public Collection getBooksPublishedAfterGivenYear(int year) throws BookException {  if (year < 1700) {  throw new BookException("Год должен быть больше 1700");  }  Stream<Book> result = Arrays.stream(this.books).filter((b) -> b.getYearOfPublishing() > year);  return new Collection(result.toArray(Book[]::new));  }  } |

|  |
| --- |
| import java.util.Arrays;  import java.util.Objects;  public class Book {  private int id;  private String name;  private String[] authors;  private String publishingHouse;  private int yearOfPublishing;  private int numberOfPages;  private float price;  private String binding;  public Book() {}  public Book(int id, String name, String[] authors, String publishingHouse, int yearOfPublishing, int numberOfPages, float price, String binding) {  this.id = id;  this.name = name;  this.authors = authors;  this.publishingHouse = publishingHouse;  this.yearOfPublishing = yearOfPublishing;  this.numberOfPages = numberOfPages;  this.price = price;  this.binding = binding;  }  public String toString() {  return String.format(  "Book(%d) %s.\tAuthors: %s.\tPublished: %s (%d).\tPrice: %f.\t%d pages.\tBinding: %s.\n",  this.id, this.name, Arrays.toString(this.authors), this.publishingHouse, this.yearOfPublishing,  this.price, this.numberOfPages, this.binding  );  }  public void setId(int id) {  this.id = id;  }  public int getId() {  return this.id;  }  public void setName(String name) {  this.name = name;  }  public String getName() {  return this.name;  }  public void setAuthors(String[] authors) {  this.authors = authors;  }  public String[] getAuthors() {  return this.authors;  }  public void setPublishingHouse(String publishingHouse) {  this.publishingHouse = publishingHouse;  }  public String getPublishingHouse() {  return this.publishingHouse;  }  public void setYearOfPublishing(int yearOfPublishing) {  this.yearOfPublishing = yearOfPublishing;  }  public int getYearOfPublishing() {  return this.yearOfPublishing;  }  public void setNumberOfPages(int numberOfPages) {  this.numberOfPages = numberOfPages;  }  public int getNumberOfPages() {  return this.numberOfPages;  }  public void setPrice(float price) {  this.price = price;  }  public float getPrice() {  return this.price;  }  public void setBinding(String binding) {  this.binding = binding;  }  public String getBinding() {  return this.binding;  }  public boolean belongsToAuthor(String author) {  for(String el: this.authors) {  if (Objects.equals(el, author)) {  return true;  }  }  return false;  }  }  class BookException extends Exception {  public BookException(String message) {  super(message);  }  } |

**Задание 3 вариант 4**

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. Найти и вывести слова текста, для которых последняя буква одного слова совпадает с первой буквой следующего слова.

|  |
| --- |
| import java.io.\*;  public class Main {  public static void main(String[] args) {  try {  BufferedReader reader = new BufferedReader(new FileReader("input.txt"));  BufferedWriter writer = new BufferedWriter(new FileWriter("out.txt"));  String line;  while ((line = reader.readLine()) != null) {  String[] words = line.split(" ");  String previosWord = " ";  for (String word: words) {  String tempStr = word.replaceAll("[.,?!:;]+$", "");  if (previosWord.substring(previosWord.length() - 1).equals(tempStr.substring(0, 1))) {  writer.write(previosWord + " " + tempStr + System.lineSeparator());  }  previosWord = tempStr;  }  }  reader.close();  writer.close();  } catch (IOException e) {  e.printStackTrace();  }  }  } |

**Задание 3 вариант 5**

В следующих заданиях требуется ввести последовательность строк из текстового потока и выполнить указанные действия. При этом могут рассматриваться два варианта: каждая строка состоит из одного слова; каждая строка состоит из нескольких слов. Имена входного и выходного файлов, а также абсолютный путь к ним могут быть введены как параметры командной строки или храниться в файле. Найти в строке наибольшее число цифр, идущих подряд.

|  |
| --- |
| import java.io.\*;  import java.util.regex.Matcher;  import java.util.regex.Pattern;  public class Main {  public static void main(String[] args) {  try {  BufferedReader reader = new BufferedReader(new FileReader("input.txt"));  BufferedWriter writer = new BufferedWriter(new FileWriter("out.txt"));  String line;  Pattern pattern = Pattern.compile("\\d+");  while ((line = reader.readLine()) != null) {  Matcher matcher = pattern.matcher(line);  int count = 0;  while (matcher.find()) {  int tempCount = matcher.end() - matcher.start();  if (tempCount > count) {  count = tempCount;  }  }  writer.write(count + System.lineSeparator());  }  reader.close();  writer.close();  } catch (IOException e) {  e.printStackTrace();  }  }  } |

**Задание 4 вариант 4**

При выполнении следующих заданий для вывода результатов создавать новую директорию и файл средствами класса File В файле, содержащем фамилии студентов и их оценки, записать прописными буквами фамилии тех студентов, которые имеют средний балл более “7”.

|  |
| --- |
| import java.io.\*;  import java.util.Arrays;  public class Main {  public static void main(String[] args) {  try {  File file = new File("./output/out.txt");  file.getParentFile().mkdirs();  file.createNewFile();  BufferedReader reader = new BufferedReader(new FileReader("input.txt"));  BufferedWriter writer = new BufferedWriter(new FileWriter(file));  String line;  while ((line = reader.readLine()) != null) {  String[] parts = line.split(":");  String name = parts[0].trim();  String marksStr = parts[1].trim();  String[] marks = marksStr.split(", ");  int sum = Arrays.stream(marks).mapToInt(Integer::parseInt).sum();  if ((float)sum / marks.length > 7) {  writer.write(name.toUpperCase() + System.lineSeparator());  }  }  reader.close();  writer.close();  } catch (IOException e) {  e.printStackTrace();  }  }  } |

**Задание 4 вариант 5**

При выполнении следующих заданий для вывода результатов создавать новую директорию и файл средствами класса File Файл содержит символы, слова, целые числа и числа с плавающей запятой. Определить все данные, тип которых вводится из командной строки.

|  |
| --- |
| import java.io.\*;  import java.util.Objects;  public class Main {  public static void main(String[] args) {  String type = args[0];  try {  File file = new File("./output/out.txt");  file.getParentFile().mkdirs();  file.createNewFile();  BufferedReader reader = new BufferedReader(new FileReader("input.txt"));  BufferedWriter writer = new BufferedWriter(new FileWriter(file));  String line = reader.readLine();  String[] elements = line.split(" ");  StringBuilder result = new StringBuilder();  for (String el: elements) {  if (el.matches("[-+]?\\d+")) {  if (Objects.equals(type, "int")) {result.append(el).append(" ");}  } else if (el.matches("[-+]?[0-9]+\\.[0-9]+")) {  if (Objects.equals(type, "float")) {result.append(el).append(" ");}  } else if (el.length() == 1) {  if (Objects.equals(type, "char")) {result.append(el).append(" ");}  } else if (Objects.equals(type, "string")) {  result.append(el).append(" ");  }  }  writer.write(String.valueOf(result));  reader.close();  writer.close();  } catch (IOException e) {  e.printStackTrace();  }  }  } |

**Вывод:** в ходе лабораторной работы были освоены базовые принципы работы с исключениями и файлами на языке Java.