**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 **09.04.01/07 Интеллектуальные системы анализа, обработки и интерпретации больших данных**

О Т Ч Е Т

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

**Дисциплина:** Исключения. Файлы

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

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

Москва, 2023

**Цель:** ознакомиться с базовыми принципами языка Java для работы с большими данными

**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.

**Решение:**

package lr5\_v1\_task1;  
  
import java.util.Arrays;  
import java.util.Scanner;  
  
public class lr5\_v1\_task1 {  
 public static void main(String[] args) {  
 Scanner in = new Scanner(System.in);  
  
  
 System.out.println("Введите кол-во векторов: ");  
 String input = in.nextLine();  
  
 try {  
 int size\_array = Integer.parseInt(input);  
 }  
 catch (NumberFormatException e){  
 System.out.println("Ошибка: введенное значение не является целым числом");  
 }  
  
 Vector[] myArray = new Vector[Integer.parseInt(input)];  
  
 for (int i = 0; i < myArray.length; i++){  
  
 System.out.println("Укажите размер " + (i+1) + " вектора:");  
 String input\_size = in.nextLine();  
  
 try {  
 int size\_vector = Integer.parseInt(input\_size);  
 }  
 catch (NumberFormatException e){  
 System.out.println("Ошибка: введенное значение не является целым числом");  
 }  
  
 myArray[i] = new Vector(Integer.parseInt(input\_size));  
  
 for (int j = 0; j < myArray[i].getSize(); j++){  
  
 System.out.println("Введите число для вектора:");  
 String input\_number = in.nextLine();  
  
 try {  
 double number = Double.parseDouble(input\_number);  
 }  
 catch (NumberFormatException e){  
 System.out.println("Ошибка: введенное значение не является числом типа double");  
 }  
  
 myArray[i].set(j, Double.parseDouble(input\_number));  
 }  
  
 System.out.println(Arrays.toString(myArray[i].getValues()));  
 }  
  
 for (int i = 0; i < myArray.length; i++){  
 for (int j = i+1; j < myArray.length; j++){  
 System.out.println();  
 System.out.println("Вычисления для векторов: " + Arrays.toString(myArray[i].getValues()) + " " + Arrays.toString(myArray[j].getValues()));  
 System.out.println("Скалярное произведение: " + myArray[i].scalar(myArray[j]));  
 System.out.println("Длина первого вектора: " + myArray[i].length());  
 System.out.println("Длина второго вектора: " + myArray[j].length());  
 System.out.println("Угол между векторами(в градусах): " + myArray[i].angle(myArray[j]));  
 }  
 }  
  
 }  
}

package lr5\_v1\_task1;  
  
public class Vector {  
 private double[] values;  
 private int size;  
 public Vector(int size){  
 if (size <= 1){  
 throw new IllegalArgumentException("Неверный размер вектора: " + size);  
 }  
 this.size = size;  
 this.values = new double[size];  
 }  
  
 public void add(int value){  
 values[size] = value;  
 size++;  
 }  
  
 public double get(int index) {  
 checkIndex(index);  
 return values[index];  
 }  
  
 public void set(int index, double value) {  
 checkIndex(index);  
 values[index] = value;  
 }  
  
  
 public Vector sum(Vector V1){  
 checkDimensions(V1);  
 Vector resultV = new Vector(V1.values.length);  
 for (int i = 0; i < V1.values.length; i++){  
 resultV.values[i] = V1.values[i] + this.values[i];  
 }  
 return resultV;  
 }  
  
 public Vector sub(Vector V1){  
 checkDimensions(V1);  
 Vector resultV = new Vector(V1.values.length);  
 for (int i = 0; i < V1.values.length; i++){  
 resultV.values[i] = V1.values[i] - this.values[i];  
 }  
 return resultV;  
 }  
  
 public Vector multy(Vector V1){  
 checkDimensions(V1);  
 Vector resultV = new Vector(V1.values.length);  
 for (int i = 0; i < V1.values.length; i++){  
 resultV.values[i] = V1.values[i] \* this.values[i];  
 }  
 return resultV;  
 }  
  
  
 public Vector inc(){  
 for (int i = 0; i < getValues().length; i++){  
 values[i]++;  
 }  
 return this;  
 }  
 public Vector dec(){  
 for (int i = 0; i < getValues().length; i++){  
 values[i]--;  
 }  
 return this;  
 }  
  
 public double length(){  
 int length = 0;  
 for (int i = 0; i < getSize(); i++){  
 length += Math.pow(values[i],2);  
 }  
 return Math.sqrt(length);  
 }  
  
 public double scalar(Vector V1){  
 checkDimensions(V1);  
 double scalar = 0;  
 for(int i = 0; i < getSize(); i++){  
 scalar += V1.values[i] \* values[i];  
 }  
 return scalar;  
 }  
  
 public double angle(Vector V1){  
 checkDimensions(V1);  
 double angle = scalar(V1)/(V1.length() \* length());  
 return Math.toDegrees(Math.acos(angle));  
 }  
  
 public int getSize() {  
 return size;  
 }  
  
 public double[] getValues() {  
 return values;  
 }  
  
 public void checkIndex(int index){  
 if(index < 0 || index >= getSize()){  
 throw new IndexOutOfBoundsException("Неверный индекс: " + index);  
 }  
 }  
  
 public void checkDimensions(Vector other){  
 if (values.length != other.values.length){  
 throw new IllegalArgumentException("Размеры векторов не равны. Размер первого: " + this.getSize() + ", размер второго: " + other.getSize());  
 }  
 }  
}

На рисунке 1 представлен результат выполнения.

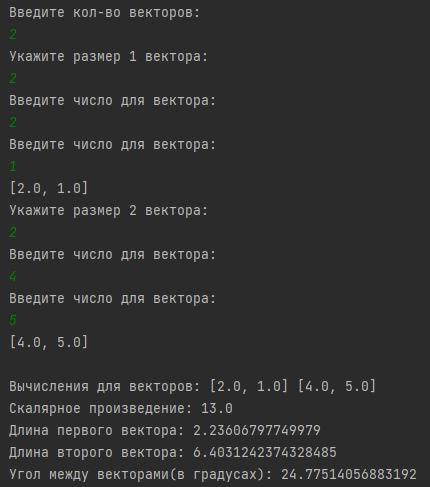


Рисунок 1 – Результат выполнения

package lt5\_v1\_task2;  
  
import java.util.Arrays;  
import java.util.Scanner;  
  
public class lr5\_v1\_task2 {  
 public static void main(String[] args) {  
 Scanner in = new Scanner(System.*in*);  
  
  
 System.*out*.println("Введите кол-во векторов: ");  
 String input = in.nextLine();  
  
 try {  
 int size\_array = Integer.*parseInt*(input);  
 }  
 catch (NumberFormatException e){  
 System.*out*.println("Ошибка: введенное значение не является целым числом");  
 }  
  
 Vector[] myArray = new Vector[Integer.*parseInt*(input)];  
  
 for (int i = 0; i < myArray.length; i++){  
  
 System.*out*.println("Укажите размер " + (i+1) + " вектора:");  
 String input\_size = in.nextLine();  
  
 try {  
 int size\_vector = Integer.*parseInt*(input\_size);  
 }  
 catch (NumberFormatException e){  
 System.*out*.println("Ошибка: введенное значение не является целым числом");  
 }  
  
 myArray[i] = new Vector(Integer.parseInt(input\_size));  
  
 for (int j = 0; j < myArray[i].getSize(); j++){  
  
 System.out.println("Введите число для вектора:");  
 String input\_number = in.nextLine();  
  
 try {  
 double number = Double.parseDouble(input\_number);  
 }  
 catch (NumberFormatException e){  
 System.out.println("Ошибка: введенное значение не является числом типа double");  
 }  
  
 myArray[i].set(j, Double.parseDouble(input\_number));  
 }  
  
 System.out.println(Arrays.toString(myArray[i].getValues()));  
 }  
  
 for (int i = 0; i < myArray.length; i++){  
 for (int j = i+1; j < myArray.length; j++){  
 System.out.println();  
 System.out.println("Вычисления для векторов: " + Arrays.toString(myArray[i].getValues()) + " " + Arrays.toString(myArray[j].getValues()));  
 myArray[i].CollinearAndOrto(myArray[j]);  
 }  
 }  
 }  
}

package lt5\_v1\_task2;  
  
import java.util.Arrays;  
  
public class Vector {  
  
 private double[] values;  
 private int size;  
 public Vector(int size){  
 if (size <= 1){  
 throw new IllegalArgumentException("Неверный размер вектора: " + size);  
 }  
 this.size = size;  
 this.values = new double[size];  
 }  
  
 public void add(int value){  
 values[size] = value;  
 size++;  
 }  
  
 public double get(int index) {  
 checkIndex(index);  
 return values[index];  
 }  
  
 public void set(int index, double value) {  
 checkIndex(index);  
 values[index] = value;  
 }  
  
  
 public Vector sum(Vector V1){  
 checkDimensions(V1);  
 Vector resultV = new Vector(V1.values.length);  
 for (int i = 0; i < V1.values.length; i++){  
 resultV.values[i] = V1.values[i] + this.values[i];  
 }  
 return resultV;  
 }  
  
 public Vector sub(Vector V1){  
 checkDimensions(V1);  
 Vector resultV = new Vector(V1.values.length);  
 for (int i = 0; i < V1.values.length; i++){  
 resultV.values[i] = V1.values[i] - this.values[i];  
 }  
 return resultV;  
 }  
  
 public Vector multy(Vector V1){  
 checkDimensions(V1);  
 Vector resultV = new Vector(V1.values.length);  
 for (int i = 0; i < V1.values.length; i++){  
 resultV.values[i] = V1.values[i] \* this.values[i];  
 }  
 return resultV;  
 }  
  
  
 public Vector inc(){  
 for (int i = 0; i < getValues().length; i++){  
 values[i]++;  
 }  
 return this;  
 }  
 public Vector dec(){  
 for (int i = 0; i < getValues().length; i++){  
 values[i]--;  
 }  
 return this;  
 }  
  
 public double length(){  
 int length = 0;  
 for (int i = 0; i < getSize(); i++){  
 length += Math.*pow*(values[i],2);  
 }  
 return Math.*sqrt*(length);  
 }  
  
 public double scalar(Vector V1){  
 checkDimensions(V1);  
 double scalar = 0;  
 for(int i = 0; i < getSize(); i++){  
 scalar += V1.values[i] \* values[i];  
 }  
 return scalar;  
 }  
  
 public double angle(Vector V1){  
 checkDimensions(V1);  
 double angle = scalar(V1)/(V1.length() \* length());  
 return Math.*toDegrees*(Math.*acos*(angle));  
 }  
  
 public Vector multiplication(double con){  
 for (int i = 0; i < getSize(); i++){  
 values[i] = values[i] \* con;  
 }  
 return this;  
 }  
  
 public void CollinearAndOrto(Vector V1){  
 checkDimensions(V1);  
 double k = 0;  
 for (int i = 0; i < getSize(); i++){  
 if ((V1.values[i] != 0) && (this.values[i] != 0)){  
 k = this.values[i]/V1.values[i];  
 }  
 }  
 if (Arrays.*equals*(this.values,V1.multiplication(k).values)){  
 System.*out*.println("Данные векторы коллинеарны");  
 }  
 else {  
 System.*out*.println("Данные векторы не коллинеарны");  
 }  
 if (this.scalar(V1) == 0){  
 System.*out*.println("Векторы ортогональны");  
 }  
 else {  
 System.*out*.println("Векторы не ортогональны");  
 }  
 }  
  
 public int getSize() {  
 return size;  
 }  
  
 public double[] getValues() {  
 return values;  
 }  
  
 public void checkIndex(int index){  
 if(index < 0 || index >= getSize()){  
 throw new IndexOutOfBoundsException("Неверный индекс: " + index);  
 }  
 }  
  
 public void checkDimensions(Vector other){  
 if (values.length != other.values.length){  
 throw new IllegalArgumentException("Размеры векторов не равны. Размер первого: " + this.getSize() + ", размер второго: " + other.getSize());  
 }  
 }  
}

На рисунке 2 представлен результат выполнения.

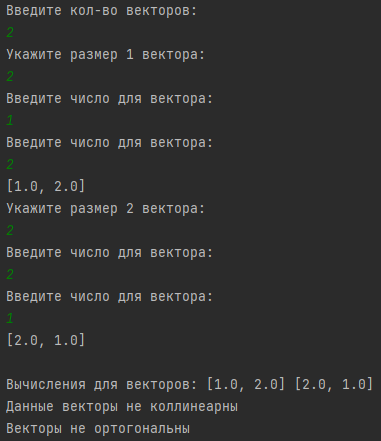


Рисунок 2 – Результат выполнения

**Вариант 2:**

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

package lr5\_v2\_task1;  
  
import java.time.LocalDate;  
import java.time.format.DateTimeFormatter;  
import java.time.format.DateTimeParseException;  
import java.util.Scanner;  
  
public class lr5\_v2\_task1 {  
 public static void main(String[] args) {  
 Scanner in = new Scanner(System.*in*);  
 StudentArray studentArray = new StudentArray();  
  
 System.*out*.println("Введите кол-во студентов: ");  
 String input = in.nextLine();  
  
 try {  
 Integer.*parseInt*(input);  
 }  
 catch (NumberFormatException e){  
 System.*out*.println("Ошибка: введенное значение не является целым числом");  
 }  
  
 for (int i=0; i<Integer.*parseInt*(input); i++){  
 System.*out*.println("Введите имя, фамилию, отчество студента:");  
 String lastName = in.nextLine();  
 String name = in.nextLine();  
 String middleName = in.nextLine();  
 System.*out*.println("Введите дату рождения в формате yyyy-MM-dd:");  
 String dateStr = in.nextLine();  
 DateTimeFormatter formatter = DateTimeFormatter.*ofPattern*("yyyy-MM-dd");  
 try {  
 LocalDate.*parse*(dateStr, formatter);  
 }  
 catch (DateTimeParseException e){  
 System.*out*.println("Ошибка: введенное значение не является датой формата yyyy-MM-dd");  
 }  
 System.*out*.println("Введите телефон студента:");  
 String telephone = in.nextLine();  
 System.*out*.println("Введите Факультет, курс, группу студента");  
 String faculty = in.nextLine();  
 String coure = in.nextLine();  
 String group = in.nextLine();  
 try {  
 Integer.*parseInt*(coure);  
 Integer.*parseInt*(group);  
 }  
 catch (NumberFormatException e){  
 System.*out*.println("Ошибка: непаравильно указана группа или курс студента");  
 }  
 studentArray.addStudent(new Student(lastName,name,middleName,LocalDate.*parse*(dateStr, formatter),"Moscow",telephone,faculty,Integer.*parseInt*(coure),Integer.*parseInt*(group)));  
  
 }  
  
 System.*out*.println("Полный список студентов");  
 studentArray.printAllStudents();  
 System.*out*.println();  
 System.*out*.println("Список студентов факультета Математики");  
 studentArray.printStudentsCurrentFaculty("Математика");  
 System.*out*.println();  
 System.*out*.println("Список студентов по всем курсам и факультетам");  
 studentArray.printStudentsByFacultyAndCourse();  
 System.*out*.println();  
 System.*out*.println("Список студентов, рождённых после определённой даты");  
 studentArray.printStudentsBirthAfterDate(LocalDate.*of*(2000, 4, 1));  
 System.*out*.println("Учебная группа");  
 studentArray.printStudentsCurrentlyGroup(3, 201, "Physics");  
 }  
}

package lr5\_v2\_task1;  
  
import java.time.LocalDate;  
  
public class Student {  
 private static int *Counter* = 0;  
 private int id, Course, Group;  
 private String LastName, FirstName, Patronymic, Address, Telephone, Faculty;  
 private LocalDate DateOfBirthday;  
 public Student( String lastName, String firstName, String middleName, LocalDate birthDate, String address, String phone, String faculty, int course, int group) {  
 this.id = *Counter*++;  
 this.LastName = lastName;  
 this.FirstName = firstName;  
 this.Patronymic = middleName;  
 this.DateOfBirthday = birthDate;  
 this.Address = address;  
 this.Telephone = phone;  
 this.Faculty = faculty;  
 this.Course = course;  
 this.Group = group;  
 }  
 //getters  
 public String getLastName(){  
 return this.LastName;  
 }  
 public String getFirstName(){  
 return this.FirstName;  
 }  
 public String getAddress(){  
 return this.Address;  
 }  
 public String getTelephone(){  
 return this.Telephone;  
 }  
 public String getPatronymic(){  
 return this.Patronymic;  
 }  
 public String getFaculty(){  
 return this.Faculty;  
 }  
 public int getCourse(){  
 return this.Course;  
 }  
 public int getGroup(){  
 return this.Group;  
 }  
 public LocalDate getDateOfBirthday(){  
 return this.DateOfBirthday;  
 }  
 public int getId(){  
 return this.id;  
 }  
 //setters  
 public void setId(int id) {  
 this.id = id;  
 }  
 public void setAddress(String address) {  
 this.Address = Address;  
 }  
 public void setCourse(int course) {  
 this.Course = course;  
 }  
 public void setDateOfBirthday(LocalDate dateOfBirthday) {  
 this.DateOfBirthday = dateOfBirthday;  
 }  
 public void setFaculty(String faculty) {  
 Faculty = faculty;  
 }  
 public void setTelephone(String telephone) {  
 Telephone = telephone;  
 }  
 public void setFirstName(String firstName) {  
 FirstName = firstName;  
 }  
 public void setGroup(int group) {  
 Group = group;  
 }  
 public void setLastName(String lastName) {  
 LastName = lastName;  
 }  
 public void setPatronymic(String patronymic) {  
 Patronymic = patronymic;  
 }  
 public String toString() {  
 return "Student{" +  
 "id=" + id +  
 ", lastName='" + LastName + '\'' +  
 ", firstName='" + FirstName + '\'' +  
 ", middleName='" + Patronymic + '\'' +  
 ", birthDate=" + DateOfBirthday +  
 ", address='" + Address + '\'' +  
 ", phone='" + Telephone + '\'' +  
 ", faculty='" + Faculty + '\'' +  
 ", course=" + Course +  
 ", group='" + Group + '\'' +  
 '}';  
 }  
}

package lr5\_v2\_task1;  
  
import java.time.LocalDate;  
import java.util.ArrayList;  
  
public class StudentArray {  
 private ArrayList<Student> studentList;  
 public StudentArray(){  
 studentList = new ArrayList<Student>();  
 }  
 public void addStudent(Student S){  
 studentList.add(S);  
 }  
  
 public void printStudentById(int id) {  
 try {  
 Student student = findStudentById(id);  
 System.*out*.println(student.toString());  
 } catch (StudentNotFoudnException e) {  
 System.*out*.println(e.getMessage());  
 }  
 }  
  
 public Student findStudentById(int id) throws StudentNotFoudnException {  
 for (Student student : studentList) {  
 if (student.getId() == id) {  
 return student;  
 }  
 }  
 throw new StudentNotFoudnException("Студент с id " + id + " не найден.");  
 }  
  
 public void printAllStudents(){  
 for (Student student : studentList){  
 System.*out*.println(student.toString());  
 }  
 }  
 public void printStudentsCurrentFaculty(String Faculty){  
 for (Student student: studentList){  
 if(student.getFaculty().equals(Faculty)){  
 System.*out*.println(student.toString());  
 }  
 }  
 }  
 public void printStudentsByFacultyAndCourse() {  
 ArrayList ListOfFaculty = new ArrayList<String>();  
 ArrayList ListOfCourse = new ArrayList<String>();  
 for (Student student : studentList) {  
 if (!(ListOfFaculty.contains(student.getFaculty())) || !(ListOfCourse.contains(student.getCourse()))){  
 System.*out*.println(student.getFaculty() + ", Course " + student.getCourse() + ":");  
 for (Student s : studentList) {  
 if (s.getFaculty().equals(student.getFaculty()) && s.getCourse() == student.getCourse()) {  
 System.*out*.println(s);  
 }  
 }  
 System.*out*.println();  
 }  
 ListOfFaculty.add(student.getFaculty());  
 ListOfCourse.add(student.getCourse());  
 }  
 }  
 public void printStudentsBirthAfterDate(LocalDate DateOfBirth){  
 for (Student student: studentList){  
 if (student.getDateOfBirthday().isAfter(DateOfBirth)){  
 System.out.println(student.toString());  
 }  
 }  
 }  
 public void printStudentsCurrentlyGroup(int Course,int Group, String Faculty){  
 for (Student student: studentList){  
 if (student.getGroup() == Group && student.getCourse() == Course && student.getFaculty() == Faculty){  
 System.out.println(student.toString());  
 }  
 }  
 }  
}

package lr5\_v2\_task1;  
  
public class StudentNotFoudnException extends Exception {  
 public StudentNotFoudnException(String message) {  
 super(message);  
 }  
}

На рисунке 3 представлен результат выполнения.

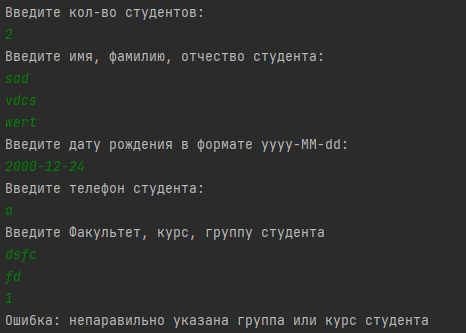


Рисунок 3 – Результат выполнения

package lr5\_v2\_task2;  
  
public class Customer {  
 private int id;  
 private String lastName;  
 private String firstName;  
 private String middleName;  
 private String address;  
 private int creditCardNumber;  
 private int bankAccountNumber;  
  
 public Customer(int id, String lastName, String firstName, String middleName, String address,  
 int creditCardNumber, int bankAccountNumber) {  
 this.id = id;  
 this.lastName = lastName;  
 this.firstName = firstName;  
 this.middleName = middleName;  
 this.address = address;  
 this.creditCardNumber = creditCardNumber;  
 this.bankAccountNumber = bankAccountNumber;  
 }  
  
 public int getId() {  
 return id;  
 }  
  
 public void setId(int id) {  
 this.id = id;  
 }  
  
 public String getLastName() {  
 return lastName;  
 }  
  
 public void setLastName(String lastName) {  
 this.lastName = lastName;  
 }  
  
 public String getFirstName() {  
 return firstName;  
 }  
  
 public void setFirstName(String firstName) {  
 this.firstName = firstName;  
 }  
  
 public String getMiddleName() {  
 return middleName;  
 }  
  
 public void setMiddleName(String middleName) {  
 this.middleName = middleName;  
 }  
  
 public String getAddress() {  
 return address;  
 }  
  
 public void setAddress(String address) {  
 this.address = address;  
 }  
  
 public long getCreditCardNumber() {  
 return creditCardNumber;  
 }  
  
 public void setCreditCardNumber(int creditCardNumber) {  
 this.creditCardNumber = creditCardNumber;  
 }  
  
 public long getBankAccountNumber() {  
 return bankAccountNumber;  
 }  
  
 public void setBankAccountNumber(int bankAccountNumber) {  
 this.bankAccountNumber = bankAccountNumber;  
 }  
 public String toString() {  
 return "Customer{" +  
 "id=" + id +  
 ", lastName='" + lastName + '\'' +  
 ", firstName='" + firstName + '\'' +  
 ", middleName='" + middleName + '\'' +  
 ", address='" + address + '\'' +  
 ", creditCardNumber=" + creditCardNumber +  
 ", bankAccountNumber=" + bankAccountNumber +  
 '}';  
 }  
}

package lr5\_v2\_task2;  
  
import java.util.ArrayList;  
import java.util.Collections;  
  
public class CustomerArray {  
 private ArrayList<Customer> CustomerArray;  
 public CustomerArray(){  
 CustomerArray = new ArrayList<Customer>();  
 }  
 public void addCustomer(Customer C){  
 CustomerArray.add(C);  
 }  
 public void ptintlAllCustomer(){  
 for (Customer customer : CustomerArray){  
 System.*out*.println(customer.toString());  
 }  
 }  
 public void printCustomersInAlphabeticalOrder(){  
 Collections.sort(CustomerArray, new CustomerCompare());  
 ptintlAllCustomer();  
 }  
 public void printCreidtCardNumberInRange(int n1, int n2) throws CustomerNotFoundException{  
 boolean flag = false;  
 for (Customer customer : CustomerArray){  
 if (customer.getCreditCardNumber() >= n1 && customer.getCreditCardNumber() <= n2){  
 System.out.println(customer.toString());  
 flag = true;  
 }  
 }  
 if (flag == false) {  
 throw new CustomerNotFoundException("В предложенном диапазоне нет кредитных карт");  
 }  
 }  
}

package lr5\_v2\_task2;  
  
import java.util.Comparator;  
  
public class CustomerCompare implements Comparator<Customer> {  
 public int compare(Customer p1, Customer p2) {  
 int result = p1.getLastName().compareTo(p2.getLastName());  
 if (result != 0) {  
 return result;  
 }  
  
 result = p1.getFirstName().compareTo(p2.getFirstName());  
 if (result != 0) {  
 return result;  
 }  
  
 return p1.getMiddleName().compareTo(p2.getMiddleName());  
 }  
  
}

package lr5\_v2\_task2;  
  
public class CustomerNotFoundException extends Exception{  
 public CustomerNotFoundException(String message){  
 super(message);  
 }  
}

package lr5\_v2\_task2;  
  
  
import java.util.Scanner;  
  
public class lr5\_v2\_task2 {  
 public static void main(String[] args) throws CustomerNotFoundException {  
 CustomerArray customerArray = new CustomerArray();  
 Scanner in = new Scanner(System.*in*);  
  
 System.*out*.println("Введите кол-во покупателей:");  
 String input\_size = in.nextLine();  
  
 try {  
 Integer.*parseInt*(input\_size);  
 }  
 catch (NumberFormatException e){  
 System.*out*.println("Ошибка: введенное значение не является целым числом");  
 }  
  
 for (int i=0; i<Integer.*parseInt*(input\_size); i++){  
 System.*out*.println("Введите id, имя, фамилию, отчество покупателя:");  
 String id = in.nextLine();  
 String lastName = in.nextLine();  
 String name = in.nextLine();  
 String middleName = in.nextLine();  
 try {  
 Integer.parseInt(id);  
 }  
 catch (NumberFormatException e){  
 System.out.println("Ошибка: id может быть только целым числом");  
 }  
 System.out.println("Введите номер карты и номер счёта покупателя:");  
 String cardNumber = in.nextLine();  
 String accountNumber = in.nextLine();  
 try {  
 Integer.parseInt(cardNumber);  
 Integer.parseInt(accountNumber);  
 }  
 catch (NumberFormatException e){  
 System.out.println("Ошибка: непаравильно указан номер карты или номер аккаунта покупателя");  
 }  
 customerArray.addCustomer(new Customer(Integer.parseInt(id), lastName, name, middleName,"Moscow", Integer.parseInt(cardNumber), Integer.parseInt(accountNumber)));  
  
 }  
  
 System.out.println("Полный список покупателей:");  
 customerArray.ptintlAllCustomer();  
 System.out.println();  
 System.out.println("Покупатели в алфавитном порядке:");  
 customerArray.printCustomersInAlphabeticalOrder();  
 System.out.println();  
 System.out.println("Номера кредитных карт в заданном диапазоне:");  
 customerArray.printCreidtCardNumberInRange(20\_000, 30\_000);  
 }  
}

На рисунке 4 представлен результат выполнения.

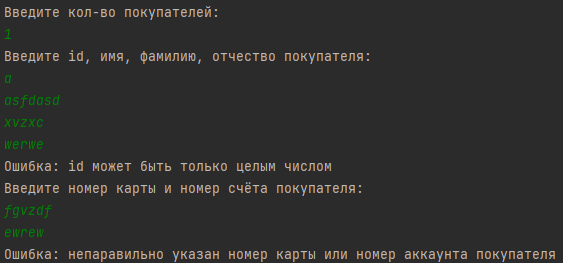


Рисунок 4 – Результат выполнения

**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.

1. В каждой строке найти и удалить заданную подстроку.

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

**Решение:**

package lr5\_v3\_task1;  
  
import java.util.ArrayList;  
import java.util.Scanner;  
  
public class lr5\_v3\_task1 {  
 public static void main(String[] args) {  
 Scanner in = new Scanner(System.in);  
 ArrayList<String> strings = new ArrayList<String>();  
  
 System.out.println("Введите подстроку для удаления: ");  
 String searchString = in.nextLine();  
  
 System.out.println("Введите последовательность строк (пустая строка для завершения ввода):");  
 while (true){  
 String currentString = in.nextLine();  
 if(currentString.isEmpty()){  
 break;  
 }  
 strings.add(currentString);  
 }  
  
 for (String string : strings){  
 string = string.replace(searchString,"");  
 System.out.println(string);  
 }  
 }  
}

На рисунке 5 представлен результат выполнения.

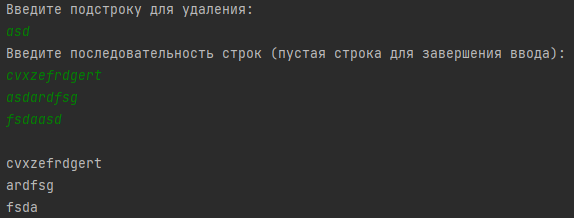


Рисунок 5 – Результат выполнения

package lr5\_v3\_task2;  
  
import java.io.File;  
import java.io.FileNotFoundException;  
import java.io.FileWriter;  
import java.io.IOException;  
import java.util.ArrayList;  
import java.util.Scanner;  
  
public class lr5\_v3\_task2 {  
 public static void main(String[] args) throws IOException {  
 String inputFileName;  
 String outputFileName;  
 String oldSubstr = "фонарь"; // заданная подстрока  
 String newSubstr = "лампа"; // заменяемая подстрока  
 ArrayList<String> strings = new ArrayList<String>();  
  
 if (args.length == 2){  
 inputFileName = args[0];  
 outputFileName = args[1];  
 }  
 else{  
 Scanner scanner = new Scanner(System.*in*);  
 System.*out*.print("Введите имя входного файла: ");  
 inputFileName = scanner.nextLine();  
 System.*out*.print("Введите имя выходного файла: ");  
 outputFileName = scanner.nextLine();  
 scanner.close();  
 }  
  
 Scanner input;  
 FileWriter output;  
 try {  
 input = new Scanner(new File(inputFileName));  
 output = new FileWriter(new File(outputFileName));  
 }  
 catch (FileNotFoundException e){  
 System.*out*.println("Один из файлов не найден.");  
 return;  
 }  
 while (input.hasNextLine()){  
 strings.add(input.nextLine());  
 }  
 input.close();  
 System.*out*.println("Исходное стихотворение:");  
 for (String string : strings) {  
 System.*out*.println(string);  
 }  
  
 System.*out*.println();  
 System.*out*.println("Результат работы программы:");  
 for (String string: strings){  
 string = string.replace(oldSubstr, newSubstr);  
 output.write(string + System.*lineSeparator*());  
 System.*out*.println(string);  
 }  
 output.close();  
 }  
}

На рисунке 6 представлен результат выполнения.

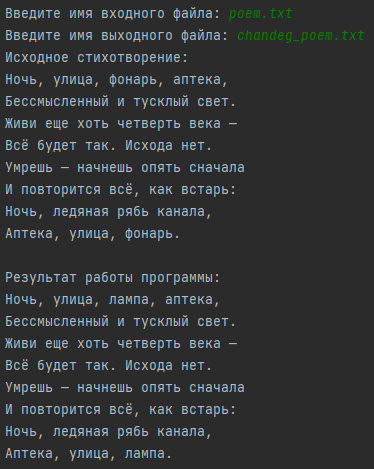


Рисунок 6 – Результат выполнения

**Вариант 4:**

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

1. Прочитать текст Java-программы и все слова public в объявлении атрибутов и методов класса заменить на слово private.
2. Прочитать текст Java-программы и записать в другой файл в обратном порядке символы каждой строки.

**Решение:**

package lr5\_v4\_task1;  
  
import java.io.File;  
import java.io.FileNotFoundException;  
import java.io.FileWriter;  
import java.io.IOException;  
import java.util.ArrayList;  
import java.util.Scanner;  
  
public class lr5\_v4\_task1 {  
 public static void main(String[] args) throws IOException {  
 ArrayList<String> strings = new ArrayList<String>();  
 Scanner inputFile;  
 FileWriter outputFile;  
  
 File directory = new File("lr5\_v4\_task1");  
 if(!directory.exists()){  
 directory.mkdirs();  
 }  
  
 File input = new File(directory,"input.txt");  
 File output = new File(directory,"output.txt");  
 if(!input.exists() && !output.exists()){  
 try {  
 input.createNewFile();  
 output.createNewFile();  
 }  
 catch (IOException e){  
 System.*out*.println("IOException");  
 }  
  
 }  
  
 try {  
 inputFile = new Scanner(input);  
 outputFile = new FileWriter(output);  
 } catch (FileNotFoundException e) {  
 System.*out*.println("Файл не найден.");  
 return;  
 }  
  
 while (inputFile.hasNextLine()){  
 strings.add(inputFile.nextLine());  
 }  
  
 for (String string : strings){  
 if (!string.contains("class") && !string.contains("interface")){  
 string = string.replace("public", "private");  
 }  
 outputFile.write(string + System.*lineSeparator*());  
 }  
 outputFile.close();  
 }  
}

На рисунке 7 представлен результат выполнения.

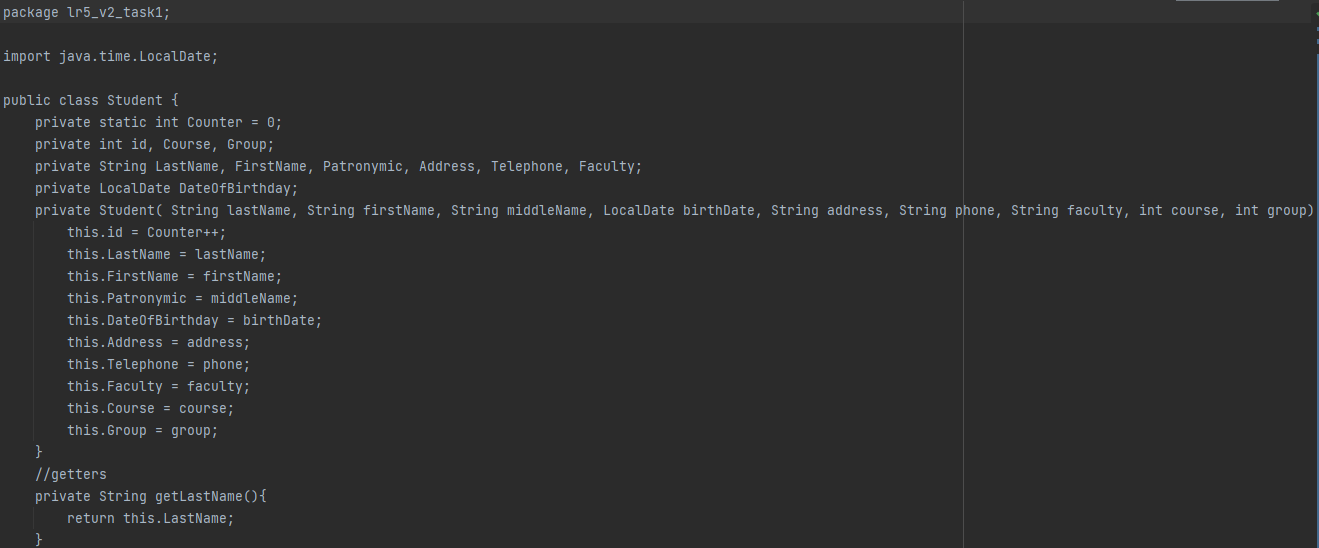


Рисунок 7 – Результат выполнения

package lr5\_v4\_task2;  
  
import java.io.File;  
import java.io.FileNotFoundException;  
import java.io.FileWriter;  
import java.io.IOException;  
import java.util.ArrayList;  
import java.util.Scanner;  
  
public class lr5\_v4\_task2 {  
 public static void main(String[] args) throws IOException {  
 ArrayList<String> strings = new ArrayList<String>();  
 Scanner inputFile;  
 FileWriter outputFile;  
  
 File directory = new File("lr5\_v4\_task2");  
 if(!directory.exists()){  
 directory.mkdirs();  
 }  
  
 File input = new File(directory,"input.txt");  
 File output = new File(directory,"output.txt");  
 if(!input.exists() && !output.exists()){  
 try {  
 input.createNewFile();  
 output.createNewFile();  
 }  
 catch (IOException e){  
 System.*out*.println("IOException");  
 }  
  
 }  
  
 try {  
 inputFile = new Scanner(input);  
 outputFile = new FileWriter(output);  
 } catch (FileNotFoundException e) {  
 System.*out*.println("Файл не найден.");  
 return;  
 }  
  
 while (inputFile.hasNextLine()){  
 strings.add(inputFile.nextLine());  
 }  
  
 for (String string : strings){  
 outputFile.write((new StringBuilder(string).reverse().toString()) + System.*lineSeparator*());  
 }  
 outputFile.close();  
 }  
}

На рисунке 8 представлен результат выполнения.

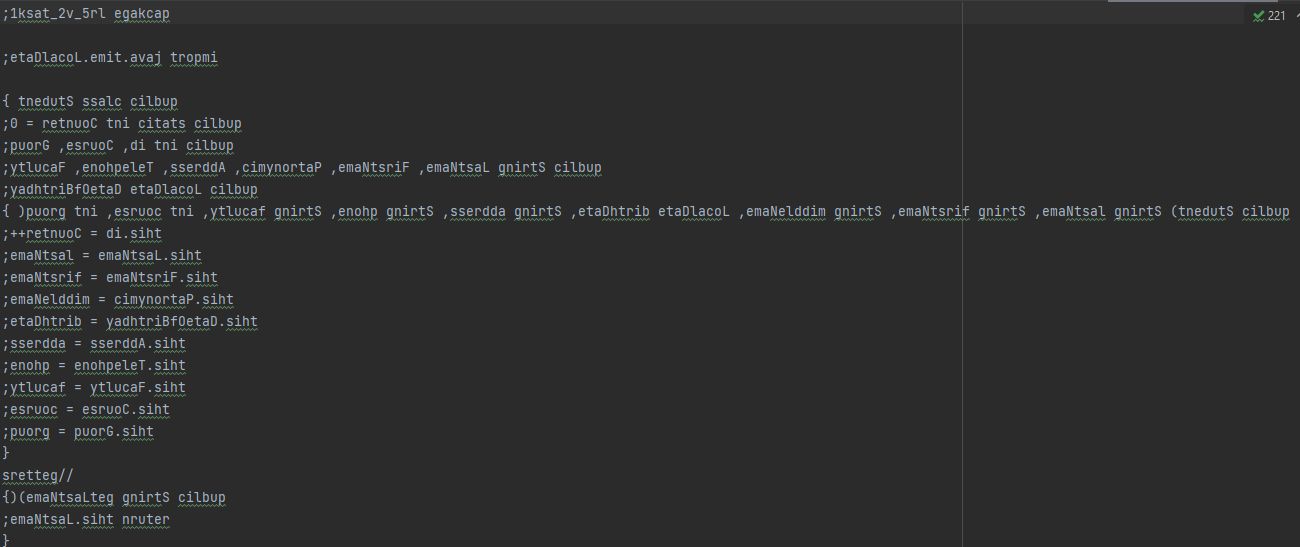


Рисунок 8 – Результат выполнения

**Вывод:** были разработаны классы и методы согласно варианту