|  |  |
| --- | --- |
| 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 |  |  | Р.В. Баканов | |
|  | (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 в Java.

**Вариант: 2.**

**Задание 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.

Определить класс Вектор размерности n. Определить несколько конструкторов. Реализовать методы для вычисления модуля вектора, скалярного произведения, сложения, вычитания, умножения на константу. Объявить массив объектов. Написать метод, который для заданной пары векторов будет определять, являются ли они коллинеарными или ортогональными.

Код решения приведен в листингах 1, 2 и 3.

Листинг 1 — Задание 1 (1)

public class Main {  
 public static void main(String[] args) {  
 var vector1 = new Vector(2, 0);  
 var vector2 = new Vector(-2, -4, 3);  
  
 try {  
 System.*out*.println(vector1.Subtract(vector2).toString());  
 }  
 catch (VectorOperationException exception) {  
 System.*out*.printf("Operation failed. %s\n", exception.getMessage());  
 }  
 }  
}

Листинг 2 – Задание 1 (2)

import java.util.ArrayList;  
import java.util.HashSet;  
import java.util.StringJoiner;  
  
public class Vector {  
 public int VectorSpaceDimension;  
 public int[] Coordinates;  
  
 public Vector(int xCoordinate) {  
 VectorSpaceDimension = 1;  
 Coordinates = new int[] { xCoordinate };  
 }  
  
 public Vector(int xCoordinate, int yCoordinate) {  
 VectorSpaceDimension = 2;  
 Coordinates = new int[] { xCoordinate, yCoordinate };  
 }  
  
 public Vector(int xCoordinate, int yCoordinate, int zCoordinate) {  
 VectorSpaceDimension = 3;  
 Coordinates = new int[] { xCoordinate, yCoordinate, zCoordinate };  
 }  
  
 public double GetModulus() {  
 var sumOfSquares = 0;  
  
 for (int i = 0; i < VectorSpaceDimension; i++) {  
 sumOfSquares += Coordinates[i] \* Coordinates[i];  
 }  
  
 return Math.*sqrt*(sumOfSquares);  
 }  
  
 public Vector Add(Vector vector) throws Exception {  
 *EnsureVectorsHaveSameVectorSpaceDimensions*(this, vector);  
  
 var resultVectorCoordinates = new ArrayList<Integer>();  
  
 for (int i = 0; i < VectorSpaceDimension; i++) {  
 resultVectorCoordinates.add(Coordinates[i] + vector.Coordinates[i]);  
 }  
  
 return *CreateVectorFromCoordinates*(resultVectorCoordinates);  
 }  
  
 public Vector Subtract(Vector vector) throws VectorOperationException {  
 *EnsureVectorsHaveSameVectorSpaceDimensions*(this, vector);  
  
 var resultVectorCoordinates = new ArrayList<Integer>();  
  
 for (int i = 0; i < VectorSpaceDimension; i++) {  
 resultVectorCoordinates.add(Coordinates[i] - vector.Coordinates[i]);  
 }  
  
 return *CreateVectorFromCoordinates*(resultVectorCoordinates);  
 }  
  
 public Vector MultiplyBy(int scalar) throws Exception {  
 var resultVectorCoordinates = new ArrayList<Integer>();  
  
 for (int i = 0; i < VectorSpaceDimension; i++) {  
 resultVectorCoordinates.add(Coordinates[i] \* scalar);  
 }  
  
 return CreateVectorFromCoordinates(resultVectorCoordinates);  
 }  
  
 public int MultiplyBy(Vector vector) throws VectorOperationException {  
 EnsureVectorsHaveSameVectorSpaceDimensions(this, vector);  
  
 var scalarProduct = 0;  
  
 for (int i = 0; i < VectorSpaceDimension; i++) {  
 scalarProduct += Coordinates[i] \* vector.Coordinates[i];  
 }  
  
 return scalarProduct;  
 }  
  
 public VectorsRelation DefineRelationTo(Vector vector) throws VectorOperationException {  
 if (this.IsCollinearTo(vector)) {  
 return VectorsRelation.Collinear;  
 }  
  
 if (this.IsOrthogonalTo(vector)) {  
 return VectorsRelation.Orthogonal;  
 }  
  
 return VectorsRelation.None;  
 }  
  
 @Override  
 public String toString() {  
 var stringJoiner = new StringJoiner(", ");  
  
 for (int coordinate : Coordinates) {  
 stringJoiner.add(Integer.toString(coordinate));  
 }  
  
 return "(" + stringJoiner + ")";  
 }  
  
 private boolean IsCollinearTo(Vector vector) throws VectorOperationException {  
 EnsureVectorsHaveSameVectorSpaceDimensions(this, vector);  
  
 var coordinateRelations = new HashSet<Double>();  
  
 for (int i = 0; i < this.VectorSpaceDimension; i++) {  
 coordinateRelations.add((double) (Coordinates[i] / vector.Coordinates[i]));  
 }  
  
 return coordinateRelations.size() == 1;  
 }  
  
 private boolean IsOrthogonalTo(Vector vector) throws VectorOperationException {  
 EnsureVectorsHaveSameVectorSpaceDimensions(this, vector);  
  
 var scalarProduct = this.MultiplyBy(vector);  
  
 return scalarProduct == 0;  
 }  
  
 private static void EnsureVectorsHaveSameVectorSpaceDimensions(Vector left, Vector right) throws VectorOperationException {  
 if (left.VectorSpaceDimension != right.VectorSpaceDimension) {  
 throw new VectorOperationException("Vectors have different vector space dimensions");  
 }  
 }  
  
 private static Vector CreateVectorFromCoordinates(ArrayList<Integer> resultVectorCoordinates) throws VectorOperationException {  
 return switch (resultVectorCoordinates.size()) {  
 case 1 -> new Vector(resultVectorCoordinates.get(0));  
 case 2 -> new Vector(resultVectorCoordinates.get(0),  
 resultVectorCoordinates.get(1));  
 case 3 -> new Vector(resultVectorCoordinates.get(0),  
 resultVectorCoordinates.get(1),  
 resultVectorCoordinates.get(2));  
 default -> throw new VectorOperationException(String.format("Vector space dimension %d is not supported", resultVectorCoordinates.size()));  
 };  
 }  
  
 public enum VectorsRelation {  
 None,  
 Collinear,  
 Orthogonal  
 }  
}

Листинг 3 – Задание 1 (3)

class VectorOperationException extends Exception {  
 public VectorOperationException(String message) {  
 super(message);  
 }  
}

**Задание 2:** 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 Вектор в R3. Реализовать методы для проверки векторов на ортогональность, проверки пересечения не ортогональных векторов, сравнения векторов. Создать массив из m объектов. Определить, какие из векторов компланарны

Код решения приведен в листингах 4 - 9.

Листинг 4 — Задание 2 (1)

public class Main {  
 public static void main(String[] args) {  
 Vector3D[] vectors = new Vector3D[] {};  
  
 try {  
 vectors = new Vector3D[] {  
 new Vector3D(  
 new Point3D(0, 0, 46341),  
 new Point3D(1, 1, -1)  
 ),  
 new Vector3D(  
 new Point3D(0, -1, 3),  
 new Point3D(-1, 2, -2)  
 ),  
 new Vector3D(  
 new Point3D(3, 2, -3),  
 new Point3D(3, -2, 3)  
 )  
 };  
 } catch (InvalidCoordinatesException exception) {  
 System.*out*.printf("Unable to create vectors. %s\n", exception.getMessage());  
  
 return;  
 }  
  
 try {  
 System.*out*.println(Vector3DBase.*AreCoplanarVectors*(vectors[0], vectors[1], vectors[2]));  
 }  
 catch (IndexOutOfBoundsException exception) {  
 System.*out*.printf("Unable to check if vectors are coplanar. %s\n", exception.getMessage());  
 }  
 }  
}

Листинг 5 – Задание 2 (2)

public class Point2D {  
 public int X;  
 public int Y;  
 public int Z;  
  
 public Point2D(int x, int y) throws InvalidCoordinatesException {  
 if (Math.*abs*(x) > 46340 || Math.*abs*(y) > 46340) {  
 throw new InvalidCoordinatesException("Coordinate can not exceed 46340");  
 }  
  
 X = x;  
 Y = y;  
 }  
}

Листинг 6 – Задание 2 (3)

public class Point3D {  
 public int X;  
 public int Y;  
 public int Z;  
 public Point2D XYProjection;  
 public Point2D XZProjection;  
 public Point2D YZProjection;  
  
 public Point3D(int x, int y, int z) throws InvalidCoordinatesException {  
 if (Math.abs(x) > 46340 || Math.abs(y) > 46340 || Math.abs(z) > 46340) {  
 throw new InvalidCoordinatesException("Coordinate can not exceed 46340");  
 }  
  
 X = x;  
 Y = y;  
 Z = z;  
 XYProjection = new Point2D(x, y);  
 XZProjection = new Point2D(x, z);  
 YZProjection = new Point2D(y, z);  
 }  
}

Листинг 7 – Задание 2 (4)

public class Vector3D extends Vector3DBase {  
 public Point3D A;  
 public Point3D B;  
  
 public Vector3D(Point3D a, Point3D b) {  
 super(b.X - a.X, b.Y - a.Y, b.Z - a.Z);  
 A = a;  
 B = b;  
 }  
  
 public boolean IntersectsWith(Vector3D vector) {  
 return AreOnSamePlane(this, vector)  
 && ProjectionsOntoXYPlaneIntersect(this, vector)  
 && ProjectionsOntoXZPlaneIntersect(this, vector)  
 && ProjectionsOntoYZPlaneIntersect(this, vector);  
 }  
  
 private static boolean AreOnSamePlane(Vector3D first, Vector3D second) {  
 var firstAuxVector = new Vector3D(first.A, second.A);  
 var secondAuxVector = new Vector3D(first.A, second.B);  
  
 return AreCoplanarVectors(first, firstAuxVector, secondAuxVector);  
 }  
  
 private static boolean ProjectionsOntoXYPlaneIntersect(Vector3D left, Vector3D right) {  
 var leftXYPlaneProjection = new Vector2DProjection(left.A.XYProjection, left.B.XYProjection);  
 var rightXYPlaneProjection = new Vector2DProjection(right.A.XYProjection, right.B.XYProjection);  
  
 return ProjectionsOntoPlaneIntersects(leftXYPlaneProjection, rightXYPlaneProjection);  
 }  
  
 private static boolean ProjectionsOntoXZPlaneIntersect(Vector3D left, Vector3D right) {  
 var leftXZPlaneProjection = new Vector2DProjection(left.A.XZProjection, left.B.XZProjection);  
 var rightXZPlaneProjection = new Vector2DProjection(right.A.XZProjection, right.B.XZProjection);  
  
 return ProjectionsOntoPlaneIntersects(leftXZPlaneProjection, rightXZPlaneProjection);  
 }  
  
 private static boolean ProjectionsOntoYZPlaneIntersect(Vector3D left, Vector3D right) {  
 var leftYZPlaneProjection = new Vector2DProjection(left.A.YZProjection, left.B.YZProjection);  
 var rightYZPlaneProjection = new Vector2DProjection(right.A.YZProjection, right.B.YZProjection);  
  
 return ProjectionsOntoPlaneIntersects(leftYZPlaneProjection, rightYZPlaneProjection);  
 }  
  
 private static boolean ProjectionsOntoPlaneIntersects(Vector2DProjection left, Vector2DProjection right) {  
 return ProjectionsOntoLineIntersects(new Vector1DProjection(left.A.X, left.B.X), new Vector1DProjection(right.A.X, right.B.X))  
 && ProjectionsOntoLineIntersects(new Vector1DProjection(left.A.Y, left.B.Y), new Vector1DProjection(right.A.Y, right.B.Y))  
 && GetSignedArea(left.A, left.B, right.A) \* GetSignedArea(left.A, left.B, right.B) <= 0  
 && GetSignedArea(right.A, right.B, left.A) \* GetSignedArea(right.A, right.B, left.B) <= 0;  
 }  
  
 private static boolean ProjectionsOntoLineIntersects(Vector1DProjection left, Vector1DProjection right) {  
 int leftMin;  
 int leftMax;  
  
 if (left.A > left.B) {  
 leftMin = left.B;  
 leftMax = left.A;  
 } else {  
 leftMin = left.A;  
 leftMax = left.B;  
 }  
  
 int rightMin;  
 int rightMax;  
  
 if (right.A > right.B) {  
 rightMin = right.B;  
 rightMax = right.A;  
 } else {  
 rightMin = right.A;  
 rightMax = right.B;  
 }  
  
 return Math.max(leftMin, rightMin) <= Math.min(leftMax, rightMax);  
 }  
  
 private static int GetSignedArea(Point2D a, Point2D b, Point2D c) {  
 return (b.X - a.X) \* (c.Y - a.Y) - (b.Y - a.Y) \* (c.X - a.X);  
 }  
  
 private static class Vector2DProjection {  
 public Point2D A;  
 public Point2D B;  
  
 public Vector2DProjection(Point2D a, Point2D b) {  
 A = a;  
 B = b;  
 }  
 }  
  
 private static class Vector1DProjection {  
 public int A;  
 public int B;  
  
 public Vector1DProjection(int a, int b) {  
 A = a;  
 B = b;  
 }  
 }  
}

Листинг 8 – Задание 2 (5)

public class Vector3DBase {  
 public int X;  
 public int Y;  
 public int Z;  
 public double Modulus;  
  
 public Vector3DBase(int x, int y, int z) {  
 X = x;  
 Y = y;  
 Z = z;  
 Modulus = Math.sqrt(X \* X + Y \* Y + Z \* Z);  
 }  
  
 public boolean IsOrthogonalTo(Vector3DBase vector) {  
 var scalarProduct = Multiply(this, vector);  
  
 return scalarProduct == 0;  
 }  
  
 public boolean IsEqualTo(Vector3DBase vector) {  
 return Modulus == vector.Modulus;  
 }  
  
 public static boolean AreCoplanarVectors(Vector3DBase first, Vector3DBase second, Vector3DBase third) {  
 var determinant = new int[][] {  
 { first.X, first.Y, first.Z },  
 { second.X, second.Y, second.Z },  
 { third.X, third.Y, third.Z }  
 };  
  
 var determinantValue = determinant[0][0] \* (determinant[1][1] \* determinant[2][2] - determinant[1][2] \* determinant[2][1])  
 - determinant[0][1] \* (determinant[1][0] \* determinant[2][2] - determinant[1][2] \* determinant[2][0])  
 + determinant[0][2] \* (determinant[1][0] \* determinant[2][1] - determinant[1][1] \* determinant[2][0]);  
  
 return determinantValue == 0;  
 }  
  
 private static int Multiply(Vector3DBase left, Vector3DBase right) {  
 return left.X \* right.X + left.Y \* right.Y + left.Z \* right.Z;  
 }  
}

Листинг 9 – Задание 2 (6)

public class InvalidCoordinatesException extends Exception {  
 public InvalidCoordinatesException(String message) {  
 super(message);  
 }  
}

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

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

Код решения приведен в листинге 10 - 15.

Листинг 10 – Задание 3 (1)

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("Number of customers:");  
  
 int count = 0;  
  
 while (count <= 0) {  
 try {  
 count = sc.nextInt();  
 } catch (InputMismatchException exception) {  
 System.*out*.printf("Unable to read number of customers. %s%n", exception.getMessage());  
  
 continue;  
 } finally {  
 sc.next();  
 }  
  
 if (count <= 0) {  
 System.*out*.println("Number of customers must be greater than zero");  
 }  
 }  
  
 var customerInputReader = new CustomerInputReader(sc);  
 var readCustomers = customerInputReader.readCustomers(count);  
  
 sc.close();  
  
 var customers = new Customer[readCustomers.size()];  
 customers = readCustomers.toArray(customers);  
  
 try {  
 var customerCollection = new CustomerCollection(customers);  
  
 System.*out*.println(customerCollection);  
  
 System.*out*.println();  
  
 System.*out*.println(customerCollection.sort(CustomerComparator.*bySurname*()));  
  
 System.*out*.println();  
  
 System.*out*.println(customerCollection.filterByCardNumber(1, 2));  
 } catch (CustomerCreationException exception) {  
 System.*out*.printf("Unable to create customer collection. %s%n", exception.getMessage());  
 }  
 }  
}

Листинг 11 – Задание 3 (2)

import java.util.StringJoiner;  
  
public class Customer {  
 private int Id;  
 private String Name;  
 private String Surname;  
 private String Patronymic;  
 private String Address;  
 private int CardNumber;  
 private int BankAccountNumber;  
  
 public Customer(int id,  
 String name,  
 String surname,  
 String patronymic,  
 String address,  
 int cardNumber,  
 int bankAccountNumber) {  
 Id = id;  
 Name = name;  
 Surname = surname;  
 Patronymic = patronymic;  
 Address = address;  
 CardNumber = cardNumber;  
 BankAccountNumber = bankAccountNumber;  
 }  
  
 @Override  
 public String toString() {  
 var stringJoiner = new StringJoiner(", ");  
  
 stringJoiner  
 .add("id: " + Id)  
 .add("name: " + Name)  
 .add("surname: " + Surname)  
 .add("patronymic: " + Patronymic)  
 .add("address: " + Address)  
 .add("card number: " + CardNumber)  
 .add("bank account number: " + BankAccountNumber);  
  
 return stringJoiner.toString();  
 }  
  
 public int getId() {  
 return Id;  
 }  
  
 public String getName() {  
 return Name;  
 }  
  
 public String getSurname() {  
 return Surname;  
 }  
  
 public String getPatronymic() {  
 return Patronymic;  
 }  
  
 public String getAddress() {  
 return Address;  
 }  
  
 public int getCardNumber() {  
 return CardNumber;  
 }  
  
 public int getBankAccountNumber() {  
 return BankAccountNumber;  
 }  
  
 public void setId(int id) {  
 Id = id;  
 }  
  
 public void setName(String name) {  
 Name = name;  
 }  
  
 public void setSurname(String surname) {  
 Surname = surname;  
 }  
  
 public void setPatronymic(String patronymic) {  
 Patronymic = patronymic;  
 }  
  
 public void setAddress(String address) {  
 Address = address;  
 }  
  
 public void setCardNumber(int cardNumber) {  
 CardNumber = cardNumber;  
 }  
  
 public void setBankAccountNumber(int bankAccountNumber) {  
 BankAccountNumber = bankAccountNumber;  
 }  
}

Листинг 12 – Задание 3 (3)

import java.util.Arrays;  
import java.util.HashSet;  
import java.util.StringJoiner;  
import java.util.function.Predicate;  
  
public class CustomerCollection {  
 public final Customer[] Data;  
  
 public CustomerCollection(Customer[] data) throws CustomerCreationException {  
 var ids = new HashSet<Integer>();  
  
 for (var customer : data) {  
 ids.add(customer.getId());  
 }  
  
 if (ids.size() < data.length) {  
 throw new CustomerCreationException("Data contains customers with same IDs");  
 }  
  
 Data = data;  
 }  
  
 public CustomerCollection sort(CustomerComparator comparator) throws CustomerCreationException {  
 var dataCopy = Data;  
  
 Arrays.*sort*(dataCopy, comparator);  
  
 return new CustomerCollection(dataCopy);  
 }  
  
 public CustomerCollection filterByCardNumber(int lowerBound, int upperBound) throws CustomerCreationException {  
 Predicate<Customer> predicate = (Customer customer) -> {  
 var cardNumber = customer.getCardNumber();  
  
 return cardNumber >= lowerBound && cardNumber <= upperBound;  
 };  
  
 var filteredData = Arrays.*stream*(Data).filter(predicate).toArray(Customer[]::new);  
  
 return new CustomerCollection(filteredData);  
 }  
  
 @Override  
 public String toString() {  
 var stringJoiner = new StringJoiner("\n");  
  
 for (var customer : Data) {  
 stringJoiner.add(customer.toString());  
 }  
  
 return stringJoiner.toString();  
 }  
}

Листинг 13 – Задание 3 (4)

import java.util.Comparator;  
  
public abstract class CustomerComparator implements Comparator<Customer> {  
 public static CustomerComparator bySurname() {  
 return new CustomerSurnameComparator();  
 }  
  
 private static class CustomerSurnameComparator extends CustomerComparator {  
  
 @Override  
 public int compare(Customer o1, Customer o2) {  
 return o1.getSurname().compareTo(o2.getSurname());  
 }  
 }  
}

Листинг 13 – Задание 3 (5)

public class CustomerCreationException extends Exception {  
 public CustomerCreationException(String message) {  
 super(message);  
 }  
}

Листинг 14 – Задание 3 (6)

import java.util.ArrayList;  
import java.util.InputMismatchException;  
import java.util.Scanner;  
  
public class CustomerInputReader {  
 private ArrayList<Customer> \_readCustomers;  
 private Scanner \_sc;  
  
 public CustomerInputReader(Scanner scanner) {  
 \_readCustomers = new ArrayList<>();  
 \_sc = scanner;  
 }  
  
 public ArrayList<Customer> readCustomers(int numberOfCustomers) {  
 \_readCustomers = new ArrayList<>();  
  
 for (int i = 0; i < numberOfCustomers; i++) {  
 var customerId = readId("'Customer ID'");  
 var fullName = readFullName();  
 var address = readAddress();  
 var cardNumber = readId("'Card number'");  
 var bankAccountNumber = readId("'Bank account number'");  
  
 var customer = new Customer(customerId,  
 fullName.Name,  
 fullName.Surname,  
 fullName.Patronymic,  
 address,  
 cardNumber,  
 bankAccountNumber);  
  
 \_readCustomers.add(customer);  
 }  
  
 return \_readCustomers;  
 }  
  
 private int readId(String parameterName) {  
 System.*out*.printf("%s%n", parameterName);  
  
 int id = -1;  
  
 while (id < 0) {  
 try {  
 id = \_sc.nextInt();  
 } catch (InputMismatchException exception) {  
 System.*out*.printf("Unable to read %s. %s%n", parameterName, exception.getMessage());  
  
 continue;  
 }  
  
 if (id < 0) {  
 System.*out*.printf("%s must be greater than or equal to zero%n", parameterName);  
 }  
 }  
  
 return id;  
 }  
  
 private FullName readFullName() {  
 System.*out*.println("Full name (format: Surname Name Patronymic):");  
 \_sc.nextLine();  
  
 boolean fullNameRead = false;  
 String[] fullName = new String[] {};  
  
 while (!fullNameRead) {  
 try {  
 fullName = \_sc.nextLine().split(" ");  
 } catch (InputMismatchException exception) {  
 System.*out*.printf("Unable to read full name. %s%n",exception.getMessage());  
  
 continue;  
 }  
  
 if (fullName.length != 3) {  
 System.*out*.println("Full name must be provided in following format: Surname Name Patronymic");  
  
 continue;  
 }  
  
 fullNameRead = true;  
 }  
  
 return new FullName(fullName[1], fullName[0], fullName[2]);  
 }  
  
 private String readAddress()  
 {  
 System.*out*.println("Address:");  
  
 var address = "";  
  
 while (address == "") {  
 try {  
 address = \_sc.nextLine();  
 } catch (InputMismatchException exception) {  
 System.out.printf("Unable to read address. %s%n",exception.getMessage());  
 }  
 }  
  
 return address;  
 }  
  
 private class FullName {  
 public String Name;  
 public String Surname;  
 public String Patronymic;  
  
 public FullName(String name, String surname, String patronymic) {  
 Name = name;  
 Surname = surname;  
 Patronymic = patronymic;  
 }  
 }  
}

**Задание 4:** Выполнить задания из варианта 2 лабораторной работы 3, реализуя собственные обработчики исключений и исключения ввода/вывода.

Patient: id, Фамилия, Имя, Отчество, Адрес, Телефон, Номер медицинской карты, Диагноз. Создать массив объектов. Вывести: a) список пациентов, имеющих данный диагноз; b) список пациентов, номер медицинской карты у которых находится в заданном интервале.

Код решения приведен в листинге 15 - 18.

Листинг 15 — Задание 4 (1)

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("Number of patients:");  
  
 int count = 0;  
  
 while (count <= 0) {  
 try {  
 count = sc.nextInt();  
 sc.nextLine();  
 } catch (InputMismatchException exception) {  
 System.*out*.printf("Unable to read number of patients. %s%n", exception.getMessage());  
  
 continue;  
 }  
  
 if (count <= 0) {  
 System.*out*.println("Number of patients must be greater than zero");  
 }  
 }  
  
 var patientInputReader = new PatientInputReader(sc);  
 var readPatients = patientInputReader.readPatients(count);  
  
 sc.close();  
  
 var patients = new Patient[readPatients.size()];  
 patients = readPatients.toArray(patients);  
  
 var patientCollection = new PatientCollection(patients);  
  
 System.*out*.println(patientCollection);  
 System.*out*.println();  
  
 System.*out*.println(patientCollection.filterByDiagnosis("B"));  
 System.*out*.println();  
  
 System.*out*.println(patientCollection.filterByCardNumber(1, 2));  
 }  
}

Листинг 16 – Задание 4 (2)

import java.util.StringJoiner;  
  
public class Patient {  
 private int Id;  
 private String Name;  
 private String Surname;  
 private String Patronymic;  
 private String Address;  
 private String PhoneNumber;  
 private int CardNumber;  
 private String Diagnosis;  
  
 public Patient(int id,  
 String name,  
 String surname,  
 String patronymic,  
 String address,  
 String phoneNumber,  
 int cardNumber,  
 String diagnosis) {  
 Id = id;  
 Name = name;  
 Surname = surname;  
 Patronymic = patronymic;  
 Address = address;  
 PhoneNumber = phoneNumber;  
 CardNumber = cardNumber;  
 Diagnosis = diagnosis;  
 }  
  
 @Override  
 public String toString() {  
 var stringJoiner = new StringJoiner(", ");  
  
 stringJoiner  
 .add("id: " + Id)  
 .add("name: " + Name)  
 .add("surname: " + Surname)  
 .add("patronymic: " + Patronymic)  
 .add("address: " + Address)  
 .add("phone number: " + PhoneNumber)  
 .add("card number: " + CardNumber)  
 .add("diagnosis: " + Diagnosis);  
  
 return stringJoiner.toString();  
 }  
  
 public int getId() {  
 return Id;  
 }  
  
 public String getName() {  
 return Name;  
 }  
  
 public String getSurname() {  
 return Surname;  
 }  
  
 public String getPatronymic() {  
 return Patronymic;  
 }  
  
 public String getAddress() {  
 return Address;  
 }  
 public String getPhoneNumber() {  
 return PhoneNumber;  
 }  
  
 public int getCardNumber() {  
 return CardNumber;  
 }  
  
 public String getDiagnosis() {  
 return Diagnosis;  
 }  
  
 public void setId(int id) {  
 Id = id;  
 }  
  
 public void setName(String name) {  
 Name = name;  
 }  
  
 public void setSurname(String surname) {  
 Surname = surname;  
 }  
  
 public void setPatronymic(String patronymic) {  
 Patronymic = patronymic;  
 }  
  
 public void setAddress(String address) {  
 Address = address;  
 }  
 public void setPhoneNumber(String phoneNumber) {  
 PhoneNumber = phoneNumber;  
 }  
  
 public void setCardNumber(int cardNumber) {  
 CardNumber = cardNumber;  
 }  
  
 public void setDiagnosis(String diagnosis) {  
 Diagnosis = diagnosis;  
 }  
}

Листинг 17 – Задание 4 (3)

import java.util.Arrays;  
import java.util.StringJoiner;  
import java.util.function.Predicate;  
  
public class PatientCollection {  
 public final Patient[] Data;  
  
 public PatientCollection(Patient[] data) {  
 Data = data;  
 }  
  
 public PatientCollection filterByDiagnosis(String targetDiagnosis) {  
 var filteredData = Arrays.stream(Data)  
 .filter(patient -> patient.getDiagnosis().equals(targetDiagnosis))  
 .toArray(Patient[]::new);  
  
 return new PatientCollection(filteredData);  
 }  
  
 public PatientCollection filterByCardNumber(int lowerBound, int upperBound) {  
 Predicate<Patient> predicate = (Patient customer) -> {  
 var cardNumber = customer.getCardNumber();  
  
 return cardNumber >= lowerBound && cardNumber <= upperBound;  
 };  
  
 var filteredData = Arrays.stream(Data).filter(predicate).toArray(Patient[]::new);  
  
 return new PatientCollection(filteredData);  
 }  
  
 @Override  
 public String toString() {  
 var stringJoiner = new StringJoiner("\n");  
  
 for (var patient : Data) {  
 stringJoiner.add(patient.toString());  
 }  
  
 return stringJoiner.toString();  
 }  
}

Листинг 18 – Задание 4 (4)

import java.util.ArrayList;  
import java.util.InputMismatchException;  
import java.util.Scanner;  
import java.util.regex.Matcher;  
import java.util.regex.Pattern;  
  
public class PatientInputReader {  
 private ArrayList<Patient> \_readPatients;  
 private Scanner \_sc;  
  
 public PatientInputReader(Scanner scanner) {  
 \_readPatients = new ArrayList<>();  
 \_sc = scanner;  
 }  
  
 public ArrayList<Patient> readPatients(int numberOfCustomers) {  
 \_readPatients = new ArrayList<>();  
  
 for (int i = 0; i < numberOfCustomers; i++) {  
 var patientId = readId("'Patient ID'");  
 var fullName = readFullName();  
 var address = readAddress();  
 var phoneNumber = readPhoneNumber();  
 var cardNumber = readId("'Card number'");  
 var diagnosis = readDiagnosis();  
  
 var customer = new Patient(  
 patientId,  
 fullName.Name,  
 fullName.Surname,  
 fullName.Patronymic,  
 address,  
 phoneNumber,  
 cardNumber,  
 diagnosis);  
  
 \_readPatients.add(customer);  
 }  
  
 return \_readPatients;  
 }  
  
 private int readId(String parameterName) {  
 System.out.printf("%s%n", parameterName);  
  
 int id = -1;  
  
 while (id < 0) {  
 try {  
 id = \_sc.nextInt();  
 } catch (InputMismatchException exception) {  
 System.out.printf("Unable to read %s. %s%n", parameterName, exception.getMessage());  
  
 continue;  
 }  
  
 if (id < 0) {  
 System.out.printf("%s must be greater than or equal to zero%n", parameterName);  
 }  
 }  
  
 return id;  
 }  
  
 private FullName readFullName() {  
 System.out.println("Full name (format: Surname Name Patronymic):");  
 \_sc.nextLine();  
  
 boolean fullNameRead = false;  
 String[] fullName = new String[] {};  
  
 while (!fullNameRead) {  
 try {  
 fullName = \_sc.nextLine().split(" ");  
 } catch (InputMismatchException exception) {  
 System.out.printf("Unable to read full name. %s%n",exception.getMessage());  
  
 continue;  
 }  
  
 if (fullName.length != 3) {  
 System.out.println("Full name must be provided in following format: Surname Name Patronymic");  
  
 continue;  
 }  
  
 fullNameRead = true;  
 }  
  
 return new FullName(fullName[1], fullName[0], fullName[2]);  
 }  
  
 private String readAddress()  
 {  
 System.out.println("Address:");  
  
 var address = "";  
  
 while (address == "") {  
 try {  
 address = \_sc.nextLine();  
 } catch (InputMismatchException exception) {  
 System.out.printf("Unable to read address. %s%n",exception.getMessage());  
 }  
 }  
  
 return address;  
 }  
  
 private String readPhoneNumber()  
 {  
 System.out.println("Phone number:");  
  
 boolean phoneNumberRead = false;  
 var phoneNumber = "";  
  
 String patterns  
 = "^(\\+\\d{1,3}( )?)?((\\(\\d{3}\\))|\\d{3})[- .]?\\d{3}[- .]?\\d{4}$"  
 + "|^(\\+\\d{1,3}( )?)?(\\d{3}[ ]?){2}\\d{3}$"  
 + "|^(\\+\\d{1,3}( )?)?(\\d{3}[ ]?)(\\d{2}[ ]?){2}\\d{2}$";  
  
 Pattern pattern = Pattern.compile(patterns);  
  
 while (!phoneNumberRead) {  
 try {  
 phoneNumber = \_sc.nextLine();  
 } catch (InputMismatchException exception) {  
 System.out.printf("Unable to read phone number. %s%n",exception.getMessage());  
  
 continue;  
 }  
  
 Matcher matcher = pattern.matcher(phoneNumber);  
  
 if (!matcher.matches()) {  
 System.out.println("Phone number has invalid format");  
  
 continue;  
 }  
  
 phoneNumberRead = true;  
 }  
  
 return phoneNumber;  
 }  
  
 private String readDiagnosis()  
 {  
 System.out.println("Diagnosis:");  
  
 var diagnosis = "";  
  
 while (diagnosis == "") {  
 try {  
 diagnosis = \_sc.nextLine();  
 } catch (InputMismatchException exception) {  
 System.out.printf("Unable to read diagnosis. %s%n",exception.getMessage());  
 }  
 }  
  
 return diagnosis;  
 }  
  
 private class FullName {  
 public String Name;  
 public String Surname;  
 public String Patronymic;  
  
 public FullName(String name, String surname, String patronymic) {  
 Name = name;  
 Surname = surname;  
 Patronymic = patronymic;  
 }  
 }  
}

**Задание 5:** В каждой строке стихотворения Александра Блока найти и заменить заданную подстроку на подстроку иной длины.

Код решения приведен в листинге 19.

Листинг 19 – Задание 5

import java.io.\*;  
  
public class Main {  
 public static void main(String[] args) {  
 try {  
 var reader = new BufferedReader(new FileReader(args[0] + '/' + args[1]));  
 var writer = new BufferedWriter(new FileWriter(args[0] + '/' + args[2]));  
  
 String currentLine = reader.readLine();  
  
 while (currentLine != null) {  
 String newLine = currentLine.replace("пьян", "трезв");  
  
 writer.write(newLine + '\n');  
  
 currentLine = reader.readLine();  
 }  
  
 reader.close();  
 writer.close();  
 } catch (IOException e) {  
 System.*out*.println(e.getMessage());  
 }  
 }  
}

**Задание 6:** В каждой строке найти слова, начинающиеся с гласной буквы.

Код решения приведен в листинге 20.

Листинг 20 – Задание 6

import java.io.\*;  
import java.util.HashSet;  
  
public class Main {  
 public static void main(String[] args) {  
 try {  
 var reader = new BufferedReader(new FileReader(args[0] + '/' + args[1]));  
 var writer = new BufferedWriter(new FileWriter(args[0] + '/' + args[2]));  
  
 String currentLine = reader.readLine();  
  
 var vowels = *getVowels*();  
  
 while (currentLine != null) {  
 StringBuilder newLine = new StringBuilder();  
  
 var words = currentLine.split(" ");  
  
 for (var word : words) {  
 if (vowels.contains(Character.*toLowerCase*(word.charAt(0)))) {  
 newLine.append(word).append(' ');  
 }  
 }  
  
 if (!newLine.toString().equals("")) {  
 writer.write(newLine.toString() + '\n');  
 }  
  
 currentLine = reader.readLine();  
 }  
  
 reader.close();  
 writer.close();  
 } catch (IOException e) {  
 System.*out*.println(e.getMessage());  
 }  
 }  
  
 private static HashSet<Character> getVowels() {  
 var vowels = new HashSet<Character>();  
  
 vowels.add('а');  
 vowels.add('о');  
 vowels.add('у');  
 vowels.add('е');  
 vowels.add('ы');  
 vowels.add('э');  
 vowels.add('я');  
 vowels.add('и');  
 vowels.add('ю');  
  
 return vowels;  
 }  
}

**Задание 7:** Прочитать текст Java-программы и записать в другой файл в обратном порядке символы каждой строки.

Код решения приведен в листинге 21.

Листинг 21 – Задание 7

import java.io.\*;  
  
public class Main {  
 public static void main(String[] args) {  
 try {  
 var reader = new BufferedReader(new FileReader(args[0] + "/src/" + "Main.java"));  
 var writer = new BufferedWriter(new FileWriter(args[0] + '/' + args[1]));  
  
 String currentLine = reader.readLine();  
  
 while (currentLine != null) {  
 StringBuilder newLine = new StringBuilder();  
  
 for (int i = currentLine.length() - 1; i >= 0; i--) {  
 newLine.append(currentLine.charAt(i));  
 }  
  
 writer.write(newLine.toString() + '\n');  
  
 currentLine = reader.readLine();  
 }  
  
 reader.close();  
 writer.close();  
 } catch (IOException e) {  
 System.out.println(e.getMessage());  
 }  
 }  
}

**Задание 8:** Прочитать текст Java-программы и в каждом слове длиннее двух символов все строчные символы заменить прописными.

Код решения приведен в листинге 22.

Листинг 22 – Задание 8

import java.io.\*;  
  
public class Main {  
 public static void main(String[] args) {  
 try {  
 var reader = new BufferedReader(new FileReader(args[0] + "/src/" + "Main.java"));  
 var writer = new BufferedWriter(new FileWriter(args[0] + '/' + args[1]));  
  
 String currentLine = reader.readLine();  
  
 while (currentLine != null) {  
 StringBuilder newLine = new StringBuilder();  
  
 var words = currentLine.split(" ");  
  
 for (var word : words) {  
 var wordToAdd = word;  
  
 if (word.replaceAll("\\W+","").length() > 2) {  
 wordToAdd = word.toUpperCase();  
 }  
  
 newLine.append(wordToAdd).append(' ');  
 }  
  
 writer.write(newLine.toString() + '\n');  
  
 currentLine = reader.readLine();  
 }  
  
 reader.close();  
 writer.close();  
 } catch (IOException e) {  
 System.out.println(e.getMessage());  
 }  
 }  
}

**Вывод:** в результате выполнения лабораторной работы были разработаны несколько программ в соответствие с требованиями задания.