|  |  |
| --- | --- |
| 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 **09.04.01/04 Компьютерный анализ и интерпретация больших данных.**

**Отчет**

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

**Название:** Исключения, файлы.

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

|  |  |  |  |
| --- | --- | --- | --- |
| Студент | ИУ6-21М |  | А.А. Поляков |
|  | (Группа) | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |
| Преподаватель |  |  | П.В. Степанов |
|  |  | (Подпись, дата) | (И.О. Фамилия) |

2021 г.

# Цель лабораторной работы

Освоить и получить навыки работы с языком программирования Java для применения его в работе с большими данными.

**Вариант 1**

**Задача 1**

# Задание

Выполнить задания на основе варианта 1 лабораторной работы 4, 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.

Создать класс City (город) с внутренним классом, с помощью объектов которого можно хранить информацию о проспектах, улицах, площадях..

# Текст программы

public class Main {

static public void main(String[]args){

City ct = new City();

System.out.println(ct.toString());

}

}

class City {

private static Scanner ob = new Scanner(System.in);

private Info info;

private String city;

public City() {

boolean validInput = false;

do {

try {

System.out.print("Введите название города: ");

setCity(ob.next());

validInput = true;

}

catch(Exception e) {

System.out.println(e.getMessage());

}

} while (!validInput);

info = new Info();

}

public String getCity() {

return city;

}

@Override

public String toString() {

return info.toString();

}

public void setCity(String city) throws Exception {

if (city.equals("0"))

throw new Exception("Вы не ввели название города\n");

else

this.city=city;

}

public class Info {

int pr,st,sq;

public Info() {

boolean validInput = false;

int k = 0;

do {

try {

System.out.print("Кол-во проспектов в городе:");

k = ob.nextInt();

validInput = true;

} catch(Exception e) {

System.out.println("Введите число!");

ob.next();

}

} while (!validInput);

setPr(k);

validInput = false;

k = 0;

do {

try {

System.out.print("Количество улиц в городе:");

k = ob.nextInt();

validInput = true;

} catch(Exception e) {

System.out.println("Введите число!");

ob.next();

}

} while (!validInput);

setSt(k);

validInput = false;

k = 0;

do {

try {

System.out.print("Количествово площадей в городе:");

k = ob.nextInt();

validInput = true;

} catch(Exception e) {

System.out.println("Введите число!");

ob.next();

}

} while (!validInput);

setSq(k);

}

@Override

public String toString(){

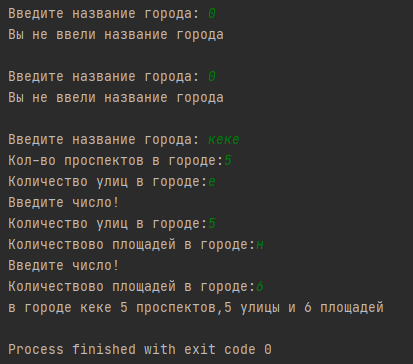
return "в городе "+getCity()+" "+getPr()+" проспектов,"+getSt()+" улицы и "+getSq()+" площадей";

}

}

}

# Результат



**Вариант 1**

**Задача 2**

# Задание

Создать класс Cinema (кино) с внутренним классом, с помощью объектов которого можно хранить информацию об адресах кинотеатров, фильмах и времени сеансов.

# Текст программы

public class Main {

static public void main(String[]args){

Cinema c = new Cinema();

c.printSeanses();

}

}

class Cinema {

List<Info> seanses = new ArrayList<>();

public Cinema() {

seanses.add(new Info("Октябрь", "Железный человек", "25 янв 19:00"));

seanses.add(new Info("Октябрь", "Джокер", "25 янв 21:30"));

seanses.add(new Info("КАРО", "1408", "26 янв 00:00"));

seanses.add(new Info("КАРО", "Ке (2018)", "26 янв 03:00"));

}

public void printSeanses() {

for (Info seanse : seanses) {

seanse.print();

}

}

public class Info {

public String k,f,t;

public Info(String k, String f, String t) {

try {

setK(k);

} catch(Exception e) {

System.out.println(e.getMessage());

try {

setK("-");

} catch(Exception ex) {

}

}

try {

setF(f);

} catch(Exception e) {

System.out.println(e.getMessage());

try {

setF("-");

} catch(Exception ex) {

}

}

setT(t);

}

public String getK() {

return k;

}

public void setK(String k) throws Exception{

List theCinemas = new ArrayList();

theCinemas.add("Октябрь");

theCinemas.add("КАРО");

theCinemas.add("СинемаСтар");

theCinemas.add("Люксор");

theCinemas.add("Синематик");

theCinemas.add("-");

if (theCinemas.contains(k))

this.k=k;

else throw new Exception("Кинотеатра '" + k + "' нет в списке\n");

}

public String getF() {

return f;

}

public void setF(String f) throws Exception{

List theFilms = new ArrayList();

theFilms.add("Железный человек");

theFilms.add("Джокер");

theFilms.add("1408");

theFilms.add("Игра");

theFilms.add("Союз спасения");

theFilms.add("-");

if (theFilms.contains(f))

this.f=f;

else throw new Exception("Фильма '" + f + "' нет в списке прокате\n");

}

public String getT(){

return t;

}

public void setT(String t){

this.t=t;

}

public void print(){

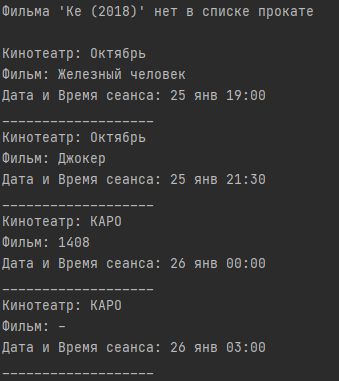
System.out.print("Кинотеатр: "+getK()+"\nФильм: "+getF()+"\nДата и Время сеанса: "+getT()+"\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

}

}

}}

# Результат



**Вариант 2**

**Задача 1**

# Задание

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

interface Корабль <- abstract class Военный Корабль <- class Авианосец.

# Текст программы

interface Ships {

int getDisplacement() throws Exception; // получить водоизмещение

int getMaxSpeed() throws Exception; // максимальую скорость

}

abstract class ArmyShip implements Ships {

protected String name; //Название

private int displacement; //остводоизмещение

private int maxSpeed; //скорь

private int distanatMax; //максимальная дистанция

private int teamCount; //Число военных в команде

public ArmyShip(String name, int displacement, int maxSpeed, int distanatMax, int teamCount) {

this.name = name;

this.displacement = displacement;

this.maxSpeed = maxSpeed;

this.distanatMax = distanatMax;

this.teamCount = teamCount;

}

public abstract String getName();

@Override

public int getDisplacement() throws Exception {

if (displacement < 0) {

throw new Exception("<Error! Отрицательное водоизмещение>");

}

else

return displacement;

}

@Override

public int getMaxSpeed() throws Exception {

if (maxSpeed <= 0 || maxSpeed > 300) {

throw new Exception("<Error! Некорректная скорость>");

}

else

return maxSpeed;

}

public int getDistanatMax() throws Exception {

if (distanatMax <= 0) {

throw new Exception("<Error! Максимальная дистанция меньше нуля>");

}

else

return distanatMax;

}

public int getTeamCount() throws Exception {

if (teamCount < 3) {

throw new Exception("<Error! Численность экипажа меньше 3-х человек>");

}

else

return teamCount;

}

}

class AeroCarrier extends ArmyShip {

private String typeShip = "Авианосец";

private int planePlaces; //Вместимость корабля

public AeroCarrier(String name, int displacement, int maxSpeed, int distanatMax, int teamCount, int planePlaces) {

super(name, displacement, maxSpeed, distanatMax, teamCount);

this.planePlaces = planePlaces;

}

@Override

public String getName() {

return name;

}

private int getPlanePlaces() throws Exception {

if (planePlaces < 0) {

throw new Exception("<Error! Отрицательное число мест для авиатехники>");

}

else

return planePlaces;

}

@Override

public String toString() {

String displacementStr = "";

String maxSpeedStr = "";

String distanatMaxStr = "";

String teamCountStr = "";

String planePlacesStr = "";

//проверка водоизмещения

try {

displacementStr = String.valueOf(getDisplacement()) + " тысяч тон";

} catch(Exception e) {

displacementStr = e.getMessage();

}

//проверка скорости

try {

maxSpeedStr = String.valueOf(getMaxSpeed()) + " узлов";

} catch(Exception e) {

maxSpeedStr = e.getMessage();

}

//проверка дальности плаванья

try {

distanatMaxStr = String.valueOf(getDistanatMax()) + " км.";

} catch(Exception e) {

distanatMaxStr = e.getMessage();

}

//проверка численности экипажа

try {

teamCountStr = String.valueOf(getTeamCount()) + " человек";

} catch(Exception e) {

teamCountStr = e.getMessage();

}

//проверка числа мест для авиатехники

try {

planePlacesStr = String.valueOf(getPlanePlaces()) + " единиц";

} catch(Exception e) {

planePlacesStr = e.getMessage();

}

return "~~~~~~'" + name + "\'~~~~~~\n" +

"Тип корабля '" + typeShip + "\';\n" +

"Численность экипажа = " + teamCountStr + ";\n" +

"Максимальная дальность плаванья = " + distanatMaxStr + "\n" +

"Максимальная скорость = " + maxSpeedStr + "\n" +

"Водоизмещение = " + displacementStr + "\n" +

"Количество мест для авиатехники = " + planePlacesStr + "\n" +

"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_";

}

}

public class Main {

public static void main(String[] args) {

ArmyShip [] ships = new ArmyShip[2];

ships[0] = new AeroCarrier("Иван IV", 20000, 30, 10000, 120, 10);

ships[1] = new AeroCarrier("Иван Третий", -500, -13, -100, -40, -4);

for (ArmyShip tmpSh : ships)

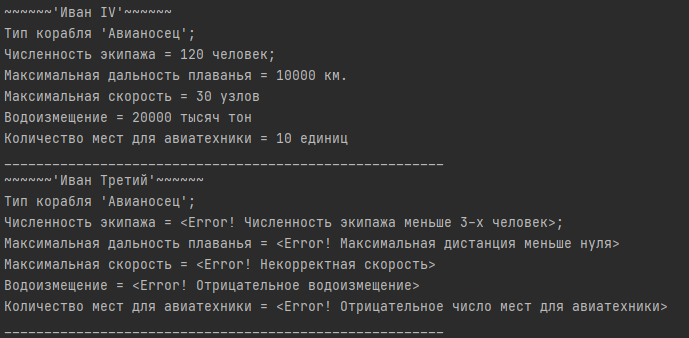
System.out.println(tmpSh);

}

}

}

# Результат



**Вариант 2**

**Задача 2**

# Задание

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

interface Здание <- abstract class Общественное Здание <- class Театр.

# Текст программы

interface Building {

int getHeight() throws Exception; // высота

int getSquareMeters() throws Exception; // площадь

}

abstract class PublicBuilding implements Building {

protected String adress; //Адрес

private int height; //высота

private int squareMeters; //площадь

public PublicBuilding(String adress, int height, int squareMeters) {

this.adress = adress;

this.height = height;

this.squareMeters = squareMeters;

}

public abstract String getAdress();

@Override

public int getHeight() throws Exception {

if (height < 3 || height > 1000) {

throw new Exception("<Error! Некорректная высота здания>");

}

else

return height;

}

@Override

public int getSquareMeters() throws Exception {

if (squareMeters < 10 || squareMeters > 1000000) {

throw new Exception("<Error! Некорректная площадь здания>");

}

else

return squareMeters;

}

}

class Theatre extends PublicBuilding {

private String type = "Театр ";

private String name;

private int halls; //Число залов

private int places; //Число мест в залах

public Theatre(String adress, int height, int squareMeters, String name, int halls, int places) {

super(adress, height, squareMeters);

this.name = name;

this.halls = halls;

this.places = places;

}

@Override

public String getAdress() {

return adress;

}

public int getHalls() throws Exception {

if (halls < 0) {

throw new Exception("<Error! Число залов меньше нуля>");

}

else

return halls;

}

public int getPlaces() throws Exception {

if (places < 0) {

throw new Exception("<Error! Число мест меньше нуля>");

}

else

return places;

}

@Override

public String toString() {

String heightStr = "";

String squareMetersStr = "";

String hallsStr = "";

String placesStr = "";

//проверка высоты здания

try {

heightStr = String.valueOf(getHeight()) + " м";

} catch(Exception e) {

heightStr = e.getMessage();

}

//проверка площади здания

try {

squareMetersStr = String.valueOf(getSquareMeters()) + " кв. м.";

} catch(Exception e) {

squareMetersStr = e.getMessage();

}

//проверка числа залов

try {

hallsStr = String.valueOf(getHalls());

} catch(Exception e) {

hallsStr = e.getMessage();

}

//проверка числа мест

try {

placesStr = String.valueOf(getPlaces());

} catch(Exception e) {

placesStr = e.getMessage();

}

return "~~~~~~'" + type + name + "\'~~~~~~\n" +

"Адрес: " + getAdress() + "\n" +

"Число залов: " + hallsStr + "\n" +

"Количество мест: " + placesStr + "\n" +

"Площадь = " + squareMetersStr + "\n" +

"Высота задния = " + heightStr + "\n" +

"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_";

}

}

public class Main {

public static void main(String[] args) {

PublicBuilding [] ships = new PublicBuilding[2];

ships[0] = new Theatre("Москва, ул. Кирова, 10", 40, 450, "им. Высоцкого", 1, 3000);

ships[1] = new Theatre("Москва, ул. Смирнова, 34а", -60, -920, "им. Савичева", -3, -10000);

for (PublicBuilding tmpSh : ships)

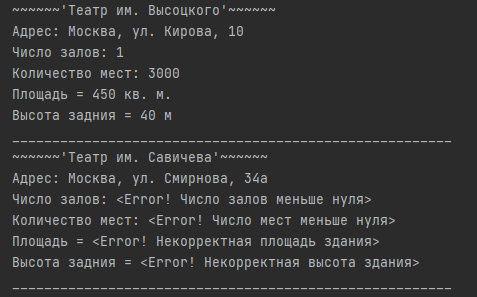
System.out.println(tmpSh);

}

}

}

# Результат



**Вариант 3**

**Задача 1**

# Задание

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...

# Текст программы

public class Main {

public static void main(String[] args) throws IOException{

Scanner scanner = new Scanner(System.in);

List<String> words = new ArrayList<String>();

Path fileName = Path.of("demo.txt");

String actual = Files.readString(fileName);

System.out.println("Содержимое файла:");

System.out.println(actual + "\n");

words.addAll(Arrays.asList(actual.split("\\s+")));

System.out.println("Слова, для которых последняя буква одного слова совпадает с первой буквой следующего слова:");

for (int i = 0; i < words.size() - 1; i++) {

if (words.get(i).charAt(words.get(i).length() - 1) == words.get(i + 1).charAt(0))

System.out.println(words.get(i) + " -> " + words.get(i+1));

}

System.out.println(words);

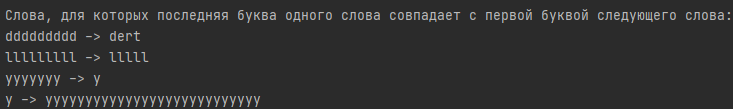
scanner.close();

}

}

}

# Результат



**Вариант 3**

**Задача 2**

# Задание

В каждой строке найти слова, начинающиеся с гласной буквы..

# Текст программы

public class Main {

public static void main(String[] args) throws IOException{

List<String> lines = new ArrayList<String>();

Path fileName = Path.of("demo2.txt");

String actual = Files.readString(fileName);

System.out.println("Содержимое файла:");

System.out.println(actual + "\n");

lines.addAll(Arrays.asList(actual.split("\\r?\\n")));

System.out.println("Слова, начинающиеся с гласной буквы");

for (int i = 0; i < lines.size(); i++) {

List<String> words = new ArrayList<String>();

String line = lines.get(i);

words.addAll(Arrays.asList(line.split("\\s+")));

System.out.print(i + " строка: ");

for (int j = 0; j < words.size() ; j++) {

String ch = Character.toString(words.get(j).charAt(0));

if (ch.matches("[aouiyeAOUIYEуеыаоэяиюУЕЫАОЭЯИЮ]"))

System.out.print(words.get(j) + ", ");

}

System.out.println("");

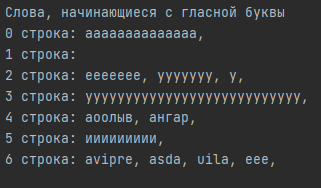
}

}

}

}

# Результат



**Вариант 4**

**Задача 1**

# Задание

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

# Текст программы

public class Main {

public static void main(String[] args) throws IOException {

List<String> lines = new ArrayList<String>();

List<String> backlines = new ArrayList<>();

Path fileName = Path.of("src\\var4\\ex1\\Main.java");

Path fileNameResult = Path.of("JavaProgramResult.txt");

String actual = Files.readString(fileName);

lines.addAll(Arrays.asList(actual.split("\\r?\\n")));

for (int i = 0; i < lines.size(); i++) {

List<String> backline = new ArrayList<>();

String line = lines.get(i);

for (int j = line.length()-1; j >= 0 ; j--) {

String ch = Character.toString(line.charAt(j));

backline.add(ch);

}

String backstring = backline.toString().substring(1, backline.toString().length() - 1);

backstring += "\n";

backlines.add(backstring);

}

String fileString = backlines.toString().substring(1, backlines.toString().length() - 1);

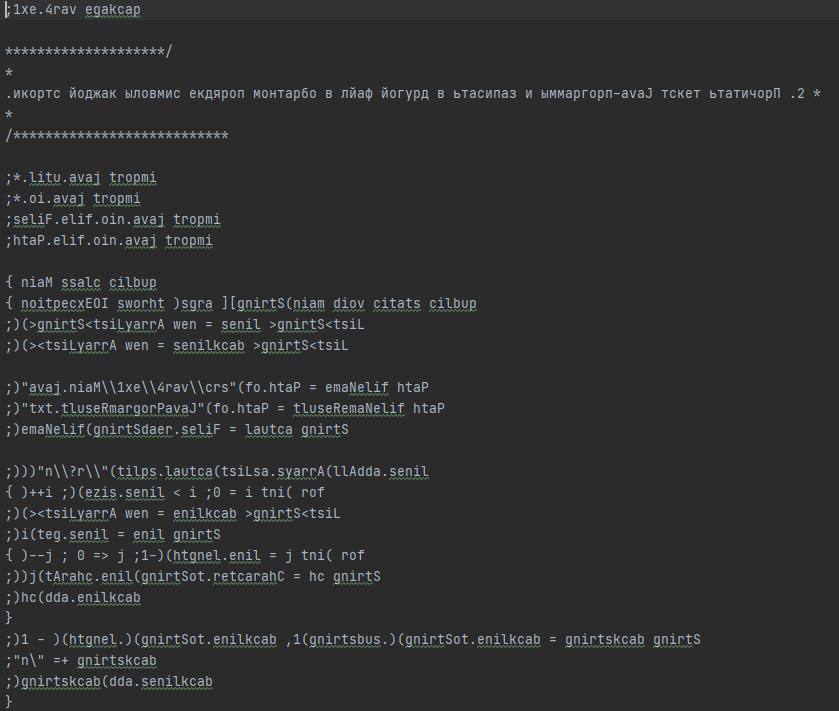
Files.writeString(fileNameResult, fileString.replace(", ", ""));

}

}

}

# Результат



**Вариант 4**

**Задача 2**

# Задание

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

# Текст программы

public class Main {

public static void main(String[] args) throws IOException {

List<String> lines = new ArrayList<String>();

List<String> changedLines = new ArrayList<>();

Path fileName = Path.of("src\\var4\\ex2\\Main.java");

Path fileNameResult = Path.of("JavaProgram\_4\_2\_Result.txt");

String actual = Files.readString(fileName);

lines.addAll(Arrays.asList(actual.split("\\r?\\n")));

for (int i = 0; i < lines.size(); i++) {

List<String> words = new ArrayList<String>();

String line = lines.get(i);

line = line.replace("public class ", "reserved class ");

line = line.replace("public ", "private ");

line = line.replace("reserved class ", "public class ");

line += "\n";

changedLines.add(line);

}

String fileString = changedLines.toString().substring(1, changedLines.toString().length() - 1);

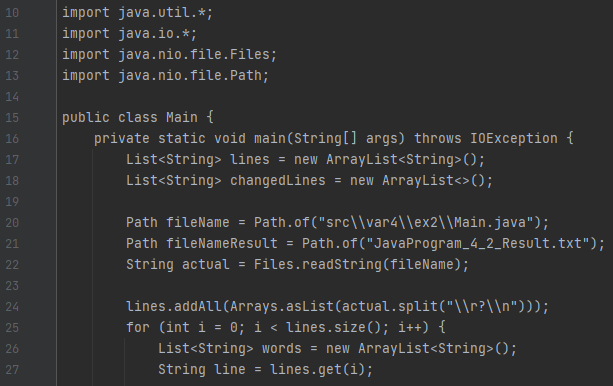
Files.writeString(fileNameResult, fileString.replace(", ", ""));

}

}

}

# Результат



**Ссылка на репозиторий**

<https://github.com/bmstu-iu6-21m/pract5-alpolyakov>

# Вывод

Был получен опыт работы с исключениями и файлами при написании программ на языке Java.