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

**Отчет по лабораторной работе 5**

***по дисциплине «Языки программирования для работы с большими данными»***

Студент ИУ6-21М **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Гришина А. М.**

(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Степанов П. В.**

(Подпись, дата) (И.О.Фамилия)

*2020 г.*

**Задание 1:**

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.

**Код программы:**

Файл Main.java:

package com.company;  
  
public class Main {  
  
 public static void main(String[] args) { *//Variant 1* try {  
 City city = new City();  
 City.CityAvenue avenue = city.new CityAvenue(City.Avenue.*Balaclava*, "Moscow, Butovo");  
 City.Square square = city.new Square(City.Street.*Tverskaya*, "Plane");  
 }  
 catch (OutOfMemoryError ex){  
 System.*out*.println("Ошибка работы динамической памяти");  
 ex.printStackTrace();  
 }  
 catch (IllegalArgumentException ex) {  
 System.*out*.println("Недопустимый аргумент, используемый для вызова метода");  
 ex.printStackTrace();  
 }  
 catch (Exception ex){  
 ex.printStackTrace();  
 }  
 }  
}

Файл City.java:

package com.company;  
import java.util.Date;  
  
public class City {  
 public enum Avenue{  
 *Balaclava*,  
 *Budyonny*,  
 *Vernadsky* }  
 public enum Street{  
 *Tverskaya*,  
 *Taganskaya*,  
 *Kantemirovskaya* }  
  
 public class CityAvenue{  
 public Avenue name;  
 public String adress;  
  
 public CityAvenue(){}  
 public CityAvenue(Avenue city, String adress){  
 this.name = city;  
 this.adress = adress;  
 }  
 }  
 public class Square{  
 public Street name;  
 public String monument;  
  
 public Square(){}  
 public Square(Street name, String monument){  
 this.name = name;  
 this.monument = monument;  
 }  
 }  
  
 public City(){}  
}

**Вывод:** Программа, созданная в лабораторной работе 4, была модифицирована в части проверки вводимых пользователем данных и добавлена обработка исключений, возникающих при нехватке памяти. Программа работает корректно.

**Задание 2:**

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

**Код программы:**

Файл Main.java:

package com.company;  
  
public class Main {  
  
 public static void main(String[] args) { *//variant 8* try {  
 Tanker bigTheater = new Tanker("Black Pearl", "Rum", 400,  
 100, "Moscow");  
 bigTheater.Raid();  
 bigTheater.change\_allowed("danger");  
 bigTheater.Raid();  
 bigTheater.Raid();  
 }  
 catch (Exception ex){  
 System.*out*.println(ex.getMessage());  
 ex.printStackTrace();  
 }  
 }  
}

Файл Ship.java:

package com.company;  
  
public interface Ship {  
 void allowed();  
 void prohibited();  
}

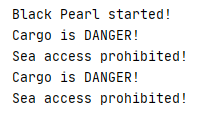
Файл CargoShip.java:

package com.company;  
  
public abstract class CargoShip implements Ship {  
 private int width;  
 private int heidth;  
 protected String cargo;  
  
 public CargoShip(int width, int heidth, String cargo){  
 this.width = width;  
 this.heidth = heidth;  
 this.cargo = cargo;  
 }  
  
 public void Raid(){  
 try {  
 if (this.cargo == "danger")  
 throw new Exception("Cargo is DANGER!");  
 allowed();  
 } catch (Exception ex){  
 System.*out*.println(ex.getMessage());  
 prohibited();  
 }  
 }  
 @Override  
 public void allowed(){  
 System.*out*.println("Sea access allowed!");  
 }  
 @Override  
 public void prohibited(){  
 System.*out*.println("Sea access prohibited!");  
 }  
  
 }

Файл Tanker.java:

package com.company;  
  
public class Tanker extends CargoShip {  
 public String name;  
  
 public Tanker(String name, String cargo, int width, int heidth, String Raid){  
 super(width, heidth, cargo);  
 this.name = name;  
 }  
  
 @Override  
 public void allowed(){  
 System.*out*.println(this.name + " started!");  
 }  
  
 public void change\_allowed(String departure){  
 cargo = departure;  
 }  
}

Проверка правильности работы:



**Вывод:** Программа, созданная в лабораторной работе 4, была модифицирована в части проверки вводимых пользователем данных. Программа работает корректно.

**Задание 3:**

**Вариант 1:** 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.

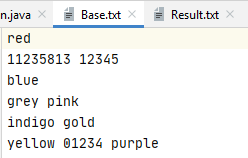
Имена входного и выходного файлов, а также абсолютный путь к ним могут быть введены как параметры командной строки или храниться в файле.

1. В каждой строке найти и удалить заданную подстроку.

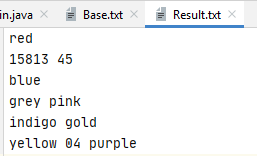
**Код программы:**

package com.company;  
  
import java.io.\*;  
import java.nio.file.attribute.FileStoreAttributeView;  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Scanner;  
  
public class Main {  
  
 public static void main(String[] args) {  
 String dir = System.*getProperty*("user.dir");  
 File b = new File(dir + "\\Base.txt");  
 File r = new File(dir + "\\Result.txt");  
  
 List<String> bl = new ArrayList<String>();  
  
 try(FileReader reader = new FileReader(b)){  
 BufferedReader br = new BufferedReader(reader);  
 String line;  
 while((line = br.readLine()) != null){  
 bl.add(line);  
 System.*out*.println(line);  
 }  
 System.*out*.println("Чтение файла завершено");  
 }  
 catch (Exception ex){  
 ex.printStackTrace();  
 }  
  
 Scanner cmd = new Scanner(System.*in*);  
 System.*out*.print("Введите подстроку для удаления: ");  
 String sub = cmd.nextLine();  
  
 try(FileWriter writer = new FileWriter(r, false)){  
 *//BufferedWriter w = new BufferedWriter(writer);* for (String s : bl)  
 writer.write(s.replaceAll(sub, "") + "\n");  
 System.*out*.println("Запись файла завершена");  
 }  
 catch (Exception ex) {  
 ex.printStackTrace();  
 }  
 }  
}

Содержимое исходного файла:



Проверка правильности работы:



**Вывод:** Программа в каждой строке находит и удаляет заданную подстроку. Программа работает корректно.

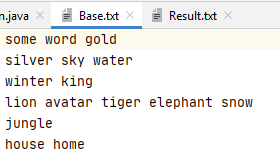
**Задание 4:**

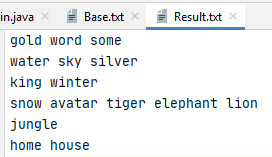
**Вариант 9:** При выполнении следующих заданий для вывода результатов создавать новую директорию и файл средствами класса File: Прочитать строки из файла и поменять местами первое и последнее слова в каждой строке.

**Код программы:**

package com.company;  
  
import java.io.\*;  
import java.nio.file.attribute.FileStoreAttributeView;  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Scanner;  
  
public class Main {  
  
 public static void main(String[] args) {  
 String dir = System.*getProperty*("user.dir");  
 File b = new File(dir + "\\Base.txt");  
 File r = new File(dir + "\\Result.txt");  
  
 List<String> bl = new ArrayList<String>();  
  
 try(FileReader reader = new FileReader(b)){  
 BufferedReader br = new BufferedReader(reader);  
 String line;  
 while((line = br.readLine()) != null){  
 bl.add(line);  
 System.*out*.println(line);  
 }  
 System.*out*.println("Чтение файла завершено");  
 }  
 catch (Exception ex){  
 ex.printStackTrace();  
 }  
  
 try(FileWriter writer = new FileWriter(r, false)){  
 for (String s : bl){  
 String[] words = s.split("\\s");  
 if (words.length > 1){  
 String line = words[words.length - 1] + " ";  
 for (int i = 1; i < words.length - 1; i++)  
 line += words[i] + " ";  
 line += words[0];  
 writer.write(line + "\n");  
 }  
 else  
 writer.write(s + "\n");  
 }  
 System.*out*.println("Запись файла завершена");  
 }  
 catch (Exception ex) {  
 ex.printStackTrace();  
 }  
 }  
}

Проверка правильности работы:





**Вывод:** Программа считывает строки из файла и меняет местами первое и последнее слова в каждой строке. Программа работает корректно.