|  |  |
| --- | --- |
| 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 |

**Название:** Исключения. Файлы

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

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

Москва, 2020

**Вариант 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.

9. Создать класс Park (парк) с внутренним классом, с помощью объектов которого можно хранить информацию об аттракционах, времени их работы и стоимости.

Park

|  |
| --- |
| public class Park {  int square;  String name;  public Park(int square, String name) {  this.square = square;  this.name = name;  }  public int getSquare() {  return square;  }  public String getName() {  return name;  }  public void setSquare(int square) {  this.square = square;  }  public void setName(String name) {  this.name = name;  }  public void displayName() {  System.out.println("Название парка: " + this.name);  }  public void displaySquare() throws IllegalArgumentException {  if (this.square < 0) {  throw new IllegalArgumentException();  }  else {  System.out.println("Площадь парка: " + this.square);  }  }  public void changeParkName (String name){  this.name = name;  }  public class Attraction {  String name\_attraction;  int ticket;  String working\_time;  public Attraction(String name\_attraction, int ticket, String working\_time) {  this.name\_attraction = name\_attraction;  this.ticket = ticket;  this.working\_time = working\_time;  }  public void changeName (String newName) throws ChangeException {  if (newName.equals("") || newName.equals("Какой-то бред")) {  throw new ChangeException (newName);  }  else {  this.name\_attraction = newName;  }  }  public void displayNameAttraction() {  System.out.println("Название аттракциона: " + this.name\_attraction);  }  public void displayWorkTime() {  System.out.println("Время работы аттракциона: " + this.working\_time);  }  public void displayTicket() {  System.out.println("Стоимость билета на аттракцион: " + this.ticket);  }  }  } |

ChangeException

|  |
| --- |
| public class ChangeException extends Exception {  String newName;  public ChangeException (String newName){  super("Название введено некорректно!");  this.newName = newName;  }  public String getNewName() {  return newName;  }  } |

|  |
| --- |
| public class LR5\_V1 {  public static void main(String[] args) {  Park disneyLand = new Park(5000, "Disney Land");  // Park.Attraction rollerCoaster = disneyLand.new Attraction("Американские горки", 250, "10:00-20:00");  Park.Attraction carousel = disneyLand.new Attraction("Карусель", 100, "09:00-21:00");  disneyLand.displayName();  disneyLand.displaySquare();  try {  disneyLand.changeParkName(null);  String namePark = disneyLand.name.toString();  } catch (NullPointerException e) {  e.printStackTrace();  }  try {  disneyLand.setSquare(-20);  disneyLand.displaySquare();  } catch (IllegalArgumentException e) {  e.printStackTrace();  }  // rollerCoaster.displayNameAttraction();  // rollerCoaster.displayTicket();  // rollerCoaster.displayWorkTime();  carousel.displayNameAttraction();  carousel.displayTicket();  carousel.displayWorkTime();  try {  carousel.changeName("Какой-то бред");  }  catch (ChangeException e) {  System.out.println("Ошибка: " + e.getMessage());  }  try {  carousel.changeName("Карусель Солнышко");  }  catch (ChangeException e) {  System.out.println("Ошибка: " + e.getMessage());  }  carousel.displayNameAttraction();  }  } |

**Вариант 2**

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

Реализовать абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующих классов:

9. interface Мебель <- abstract class Шкаф <- class Книжный Шкаф.

Furniture

|  |
| --- |
| public interface Furniture {  void toMove();  void toPaint();  } |

Cupboard

|  |
| --- |
| import java.util.Scanner;  public abstract class Cupboard implements Furniture {  String modelCupboard;  int hightCupboard;  public Cupboard(String modelCupboard, int hightCupboard) {  this.modelCupboard = modelCupboard;  this.hightCupboard = hightCupboard;  }  public String getModelCupboard() {  return modelCupboard;  }  public int getHightCupboard() {  return hightCupboard;  }  public void setModelCupboard(String modelCupboard) {  this.modelCupboard = modelCupboard;  }  public void setHightCupboard(int hightCupboard) {  this.hightCupboard = hightCupboard;  }  abstract void openDoor();  public void toMove(){  Scanner in = new Scanner(System.in);  System.out.print("Введите расстояние в см: ");  int distance = in.nextInt();  System.out.print("Введите направление: ");  Scanner in2 = new Scanner(System.in);  String direction = in2.nextLine();  System.out.printf("Шкаф передвинут на %d см в %s \n", distance, direction);  };  public abstract void toPaint();  } |

BookCase

|  |
| --- |
| import java.util.Scanner;  import java.util.regex.Matcher;  import java.util.regex.Pattern;  public class BookCase extends Cupboard{  public BookCase(String modelCupboard, int hightCupboard) {  super(modelCupboard, hightCupboard);  }  @Override  void openDoor() {  System.out.println("Дверца открыта! Можете взять книгу!");  }  @Override  public void toPaint() {  Scanner in = new Scanner(System.in);  System.out.print("Введите цвет: ");  String color = in.nextLine();  System.out.println("Шкаф перекрашен в " + color);  }  public void changePaint(String color) throws MyException {  Pattern pattern = Pattern.compile("^[1-9][0-9]\*$");  Matcher matcher = pattern.matcher(color);  if (color.equals("") || matcher.matches()) {  throw new MyException (color);  }  else {  System.out.println("Цвет шкафа изменен на " + color);  }  }  public void readModel() throws MyException {  Scanner inModel = new Scanner(System.in);  System.out.print("Введите новую модель: ");  String newModel = inModel.nextLine();  if (newModel.equals("")){  throw new MyException(newModel);  }  else {  this.modelCupboard = newModel;  }  }  } |

|  |
| --- |
| public class LR5\_V2 {  public static void main(String[] args) {  BookCase bookCase1 = new BookCase("Мечта", 210);  System.out.println("Модель: " + bookCase1.modelCupboard);  System.out.println("Высота: " + bookCase1.hightCupboard);  //bookCase1.toMove();  //bookCase1.toPaint();  bookCase1.openDoor();  try {  bookCase1.changePaint("7");  } catch (MyException e) {  System.out.println("Ошибка: " + e.getMessage());  }  try{  bookCase1.readModel();  } catch (MyException e) {  System.out.println("Ошибка: " + e.getMessage());  }  System.out.println(bookCase1.getModelCupboard());  }  } |

**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.

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

|  |
| --- |
| import java.util.Arrays;  import java.util.HashSet;  import java.util.Scanner;  import java.util.Set;  public class LR5\_V3 {  public static void main(String[] args) {  Scanner scanner = new Scanner(System.in);  Set<String> words = new HashSet<String>();  System.out.println("Для завершения ввода дважды нажмите enter.");  System.out.println("Введите несколько строк через пробел:");  String s = null;  while ( true ) {  s = scanner.nextLine();  if ( s.isEmpty() )  break;  words.addAll(Arrays.asList(s.split("\\s+")));  }  for ( String tail : words )  for ( String head : words )  if ( tail.charAt(tail.length() - 1) == head.charAt(0) )  System.out.println(tail + " -> " + head);  scanner.close();  }  } |

**Вариант 4**

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

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

|  |
| --- |
| import java.io.\*;  public class LR5\_V4 {  public static void main(String[] args) throws IOException {  String backOrderString = "";  BufferedReader br = new BufferedReader(new FileReader("C:/Users/User/IdeaProjects/LR5/reverse.txt"));  String line = "";  String FileContent = "";  String NL = System.getProperty("line.separator");  while ((line = br.readLine()) != null) {  int i = line.length();  while (i > 0) {  backOrderString = backOrderString + line.charAt(i - 1);  i--;  }  FileContent = backOrderString + NL;  }  br.close();  System.out.println(FileContent);  File file = new File("C:/Users/User/IdeaProjects/LR5/reverse\_out.txt");  if (file.createNewFile()) {  BufferedWriter bw = new BufferedWriter(new FileWriter(file));  bw.write(FileContent);  bw.close();  }  }  } |