|  |  |
| --- | --- |
| 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 | ИУ6И-21М |  |  | Ши Жуньфан |
|  | (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.

8. Создать класс Computer (компьютер) с внутренним классом, с помощью объектов которого можно хранить информацию об операционной системе, процессоре и оперативной памяти.

Computer

|  |
| --- |
| package com.company; import java.util.\*;  public class Computer\_5\_1 {  public static void main(String[] args) {  Computer\_5\_1 com = new Computer\_5\_1();  com.info.print();   }  public static Scanner *scan* = new Scanner(System.*in*);  public String computer;  public info info;  public Computer\_5\_1(){  System.*out*.println("Enter the computer name:");  setCom(*scan*.nextLine());  info = new info();  }  public String getCom(){  return computer;  }  public void setCom(String computer){  this.computer = computer;  }  public class info{  String sys,cpu;  int ram;  public info(){  System.*out*.println("Enter system:");  try{  setSys(*scan*.nextLine());  }  catch (InputMismatchException e){  System.*out*.println("The input is incorrect, please re-enter");  }  System.*out*.println("Enter cpu:");  try {  setCpu(*scan*.nextLine());  }  catch (InputMismatchException e){  System.*out*.println("The input is incorrect, please re-enter");  }  System.*out*.println("Enter ram:");  try {  setRam(*scan*.nextInt());  }  catch (InputMismatchException e){  System.*out*.println("The input is incorrect, please re-enter");  }  }  public String getSys(){  return sys;  }  public String getCpu(){  return cpu;  }  public int getRam(){  return ram;  }  public void setSys(String sys){  this.sys = sys;  }  public void setCpu(String cpu){  this.cpu = cpu;   }  public void setRam(int ram){  this.ram = ram;  }  public void print(){  System.*out*.println("Computer "+getCom()+":"+getSys()+" system,"+getCpu()+" cpu,"+getRam()+" ram.");  }  }  } |

**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

|  |
| --- |
| package com.company; import java.util.\*; public interface Furniture {  void Info();  void toPaint(); } |

Cabinet

|  |
| --- |
| package com.company; import java.util.\*; public abstract class Cabinet implements Furniture {  protected String Furniture\_type = "cupboard";  String model;  int length;  int Furniture\_id ;  public Cabinet(String model,int length,int Furniture\_id){  this.model = model;  this.length = length;  this.Furniture\_id = Furniture\_id;  }  public String getModel(){  return model;  }  public int getLength(){  return length;  }  public int getFurniture\_id(){  return Furniture\_id;  }  public void setModel(String model) {  this.model = model;  }  public void setLength(int length) {  this.length = length;  }  public void setID(int Furniture\_id) {  this.Furniture\_id = Furniture\_id;  }  public abstract void toPaint();   } |

BookCase

|  |
| --- |
| package com.company; import java.util.\*; public class Bookcase extends Cabinet {  public Bookcase(String model,int length,int Furniture\_id) {  super(model, length,Furniture\_id);  }  @Override  public void toPaint() {  Scanner scan = new Scanner(System.*in*);  System.*out*.print("choose the color: ");  String color = scan.nextLine();  System.*out*.println("Color is" + color);  }  public void readModel() throws Exception {  Scanner inModel = new Scanner(System.*in*);  System.*out*.print("choose model: ");  String newModel = inModel.nextLine();  if (newModel.equals("")){  throw new Exception(newModel);  }  else {  this.model = newModel;  }  }  public void Info(){  System.*out*.println("model:"+model);  System.*out*.println("length:"+length);  System.*out*.println("ID:"+Furniture\_id);  }    } |

Lab5\_2

|  |
| --- |
| package com.company; import java.util.\*; public class Lab5\_2 {  public static void main(String[] args) {  Bookcase bookcase1 = new Bookcase("Single desk", 210,508);  bookcase1.Info();  try{  bookcase1.readModel();  } catch (Exception e) {  System.*out*.println("error:" + e.getMessage());  }  System.*out*.println(bookcase1.getModel());  } } |

**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.

|  |
| --- |
| import java.io.\*; import java.util.\*; public class Lab5\_3\_1 {  public static void main(String[] args) {  File file = new File("C:\\Users\\Stone\\Desktop\\English.txt");  try (InputStream is = new FileInputStream(file)) {  byte[] data = new byte[1024];  is.read(data);  String str1 = new String(data);  Scanner scan = new Scanner(System.*in*);  System.*out*.println("Enter the string you want to delete:");  String str2 = scan.nextLine();  Lab5\_3\_1 test = new Lab5\_3\_1();  Object[] result = test.deleteSubString(str1,str2);  System.*out*.println("Text content:" + new String(data));  System.*out*.println("After deleting the specified string："+result[0]);  System.*out*.println("Totally deleted " +result[1] + " time(s)");  } catch (Exception e) {  e.printStackTrace();  }  }  public Object[] deleteSubString(String str1,String str2) {  StringBuffer sb = new StringBuffer(str1);  int delCount = 0;  Object[] obj = new Object[2];   while (true) {  int index = sb.indexOf(str2);  if(index == -1) {  break;  }  sb.delete(index, index+str2.length());  delCount++;  }  if(delCount!=0) {  obj[0] = sb.toString();  obj[1] = delCount;  }else {  obj[0] = -1;  obj[1] = -1;  }  return obj;  } } |

**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.

|  |
| --- |
| package com.company; import java.io.\*; import java.util.\*; public class Lab5\_4\_2 {  public static void main(String[] args) throws IOException {  BufferedReader br = new BufferedReader(new FileReader("English.txt"));  ArrayList<String> list = new ArrayList<String>();  String line ;  while((line = br.readLine()) != null){  list.add(line);  }  BufferedWriter bw = new BufferedWriter(new FileWriter("English2.txt"));  for(int i = list.size() - 1;i >= 0 ; i--){  bw.write(list.get(i));  bw.newLine();  }  br.close();  bw.close();  } } |