МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ   
ВЫСШЕГО ОБРАЗОВАНИЯ «САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЁВА (НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)» (САМАРСКИЙ УНИВЕРСИТЕТ)**

Институт информатики и кибернетики  
Кафедра программных систем

Дисциплина  
**Спецкурс по языку Ява**

**ОТЧЕТ**по лабораторной работе № 4

Студент: Кочетков Е.В.  
Группа: 6202-020302D  
  
Преподаватель: Ледкова Т.А  
  
Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
Дата: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Самара 2024

**Задание**

**Задание 1,2,3**

-------IOMethods.java-------

import java.io.\*;  
import java.util.Scanner;  
  
public class IOMethods {  
  
 public static void outputInt(Int o, OutputStream out) throws IOException {  
 o.output(out);  
 }  
 public static Int inputInt(InputStream in) throws IOException {  
 int type = in.read();  
 int nameLength = in.read();  
 byte[] nameByte = new byte[nameLength];  
  
 if (nameLength == in.read(nameByte)) {  
 String name = new String(nameByte);  
 int introduce = in.read();  
 int arrayLength = in.read();  
 int[] array = new int[arrayLength];  
 for (int i = 0; i < arrayLength; i++) array[i] = in.read();  
  
 if (type == 1) return new Essays(arrayLength, name, introduce, array);  
 else if (type == 2) return new Articles(arrayLength, name, introduce, array);  
 else throw new IOException();  
 } else throw new IOException();  
 }  
 public static void writeInt(Int o, Writer out) throws IOException {  
 o.write(out);  
 }  
 public static Int readInt(Reader in) throws IOException {  
 if (in.ready()) {  
 StreamTokenizer stream = new StreamTokenizer(in);  
 stream.nextToken();  
 if (stream.ttype == StreamTokenizer.*TT\_NUMBER*) {  
 int type = (int) stream.nval;  
 stream.nextToken();  
 if (stream.ttype == StreamTokenizer.*TT\_WORD* || stream.ttype == StreamTokenizer.*TT\_NUMBER*) {  
 String name = "";  
 if (stream.sval == null) {  
 name = String.*valueOf*(stream.nval);  
 }  
 else {  
 name = stream.sval;  
 }  
 stream.nextToken();  
 if (stream.ttype == StreamTokenizer.*TT\_NUMBER*) {  
 int introduce = (int)stream.nval;  
 stream.nextToken();  
 if (stream.ttype == StreamTokenizer.*TT\_NUMBER*) {  
 int length = (int)stream.nval;  
 int[] array = new int[length];  
 for (int i = 0; i < array.length; i++) {  
 stream.nextToken();  
 if (stream.ttype == StreamTokenizer.*TT\_NUMBER*) {  
 array[i] = (int)stream.nval;  
 }  
 }  
 if (type == 1) return new Essays(length, name, introduce, array);  
 else if (type == 2) return new Articles(length, name, introduce, array);  
 else throw new IOException();  
 }  
 }  
 }  
 }  
 }  
 else throw new IOException("Поток не задан!");  
 throw new IOException("Ошибка чтения!");  
 }  
 public static void serializeInt(Int o, OutputStream out) throws IOException {  
 ObjectOutputStream objOut = new ObjectOutputStream(out);  
 objOut.writeObject(o);  
 }  
 public static Int deserializeInt(InputStream in) throws IOException, ClassNotFoundException {  
 ObjectInputStream objIn = new ObjectInputStream(in);  
 return (Int)objIn.readObject();  
 }  
 public static void writeFormatInt(Int o, Writer out) throws IOException {  
 out.write("Название серии: '" + o.getName() + "'\n");  
 out.write( "Тип серии: 'Серия статей'\n");  
 out.write( "Кол-во статей: '" + o.getLen() + "'\n");  
 out.write( "Кол-во страниц в статьях:\n");  
 for (int i = 0; i < o.getLen(); i++) {  
 out.write( o.getElement(i) + " ");  
 }  
 out.write("\n");  
 out.write( "Кол-во вводных страниц: '" + o.getIntroduce() + "'\n");  
 }  
 public static Int readFormatInt(Scanner in) {  
 String name = "";  
 int len = 0;  
 int[] elements;  
 int introduce = 0;  
 String nextLine;  
  
 if (in.hasNextLine()) {  
 nextLine = in.nextLine();  
 name = nextLine.substring("Название серии: '".length(), nextLine.length() - 1);  
 }  
  
 in.nextLine();  
  
 if (in.hasNextLine()) {  
 nextLine = in.nextLine();  
 len = Integer.*parseInt*(nextLine.substring("Кол-во статей: '".length(), nextLine.length() - 1));  
 }  
  
 in.nextLine();  
  
 elements = new int[len];  
 if (in.hasNextLine()) {  
 String[] pages = in.nextLine().split(" ");  
 for (int i = 0; i < len; i++) {  
 elements[i] = Integer.*parseInt*(pages[i]);  
 }  
 }  
  
 if (in.hasNextLine()) {  
 nextLine = in.nextLine();  
 introduce = Integer.*parseInt*(nextLine.substring("Кол-во вводных страниц: '".length(), nextLine.length() - 1));  
 }  
 return new Articles(elements.length, name, introduce, elements);  
 }  
}

-------Articles.java-------  
import java.util.Scanner;  
  
public class Articles implements Int {  
  
 private final int[] **pages**;  
 private final String **name**;  
 private final int **introduce**;  
  
 public Articles() {  
 this(0, "NULL", 0);  
 }  
  
 public Articles(int \_count, String \_name, int \_introduce) {  
 if (\_count >= 0) {  
 **pages** = new int[\_count];  
 int check;  
 Scanner scan = new Scanner(System.***in***);  
 for (int i = 0; i < **pages**.**length**;) {  
 System.***out***.printf("Введите страницы %d-ой книги: ", i);  
 check = Integer.*parseInt*(scan.nextLine());  
 if (check < 0) { throw new RuntimeException("Страниц не может быть меньше нуля!"); }  
 else {  
 **pages**[i] = check;  
 i++;  
 }  
 }  
 }  
 else { throw new RuntimeException("Размер массива не может быть отрицательным!"); }  
  
 if (!\_name.isEmpty()) { this.**name** = \_name; }  
 else { throw new RuntimeException("Имя не может быть пустым!"); }  
  
 if (\_introduce >= 0) { this.**introduce** = \_introduce; }  
 else { throw new RuntimeException("Страницы вступления не могут быть отрицательны!"); }  
 }  
  
 @Override  
 public int getElement(int n) {  
 if ( (n >= 0) && (n < **pages**.**length**) ) { return **pages**[n]; }  
 else { throw new IndexOutOfBoundsException(); }  
 }  
  
 @Override  
 public void setElement(int n, int k) {  
 if ( (n >= 0) && (n < **pages**.**length**) ) { **pages**[n] = k; }  
 else { throw new IndexOutOfBoundsException(); }  
 }  
  
 @Override  
 public String getName() {  
 return **name**;  
 }  
  
 @Override  
 public int getIntroduce() {  
 return **introduce**;  
 }  
  
 @Override  
 public int calcPages() throws PagesLessZeroException {  
 int res = 0;  
 int midTerm = 0;  
 for (int page: **pages**) {  
 midTerm = page - **introduce**;  
 if (midTerm > -1) { res += midTerm; }  
 else { throw new PagesLessZeroException("Кол-во страниц без вступления меньше нуля!"); }  
 }  
 return res;  
 }  
  
  
 @Override  
 public String toString() {  
 String res = "Название серии: " + **name** + "\n";  
 res += "Тип серии: Серия статей\n";  
 res += "Кол-во статей: " + **pages**.**length** + "\n";  
 res += "Кол-во страниц в статьях:\n";  
 for (int page : **pages**) res += page + " ";  
 res += "\nКол-во вводных страниц: " + **introduce** + "\n";  
 return res;  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this.getClass() == o.getClass()) {  
 Articles cmp = (Articles)o;  
 if (**name**.equals(cmp.**name**)) {  
 if (**introduce** == cmp.getIntroduce()) {  
 if (**pages**.**length** == cmp.getLen()) {  
 for (int i = 0; i < **pages**.**length**; ) {  
 try {  
 if (**pages**[i] == cmp.getElement(i)) { i++; }  
 else { return false; }  
 }  
 catch (Exception e) { return false; }  
 } return true;  
 } else { return false; }  
 } else { return false; }  
 } else { return false; }  
 } else { return false; }  
 }  
  
 @Override  
 public int getLen() {  
 return **pages**.**length**;  
 }  
  
 @Override  
 public int hashCode() {  
 return **introduce** + **name**.hashCode() + **pages**.**length** \* 1000;  
 }  
  
}

-------Essays.java-------  
import java.util.Scanner;  
  
public class Essays implements Int {  
  
 private final int[] **pages**;  
 private final String **name**;  
 private final int **introduce**;  
  
 public Essays() {  
 this(0, "NULL", 0);  
 }  
  
 public Essays(int \_count, String \_name, int \_introduce) {  
 if (\_count >= 0) {   
 **pages** = new int[\_count];  
 int check;  
 Scanner scan = new Scanner(System.***in***);  
 for (int i = 0; i < **pages**.**length**;) {  
 System.***out***.printf("Введите страницы %d-ой книги: ", i);  
 check = Integer.*parseInt*(scan.nextLine());  
 if (check < 0) { throw new RuntimeException("Страниц не может быть меньше нуля!"); }  
 else {  
 **pages**[i] = check;  
 i++;  
 }  
 }  
 }  
 else { throw new RuntimeException("Размер массива не может быть отрицательным!"); }  
  
 if (!\_name.isEmpty()) { this.**name** = \_name; }  
 else { throw new RuntimeException("Имя не может быть пустым!"); }  
  
 if (\_introduce >= 0) { this.**introduce** = \_introduce; }  
 else { throw new RuntimeException("Страницы вступления не могут быть отрицательны!"); }  
 }  
  
  
 @Override  
 public int getElement(int n) {  
 if ((n >= 0) && (n < **pages**.**length**)) {  
 return **pages**[n];  
 } else { throw new IndexOutOfBoundsException(); }  
 }  
  
 @Override  
 public void setElement(int n, int k) {  
 if ((n >= 0) && (n < **pages**.**length**)) {  
 **pages**[n] = k;  
 } else { throw new IndexOutOfBoundsException(); }  
 }  
  
 public int getLen() {  
 return **pages**.**length**;  
 }  
  
 @Override  
 public String getName() {  
 return **name**;  
 }  
  
 @Override  
 public int getIntroduce() {  
 return **introduce**;  
 }  
  
 @Override  
 public int calcPages() throws PagesLessZeroException {  
 int res = 0;  
 int midTerm = 0;  
 for (int page: **pages**) {  
 midTerm = page - **introduce**;  
 if (midTerm > -1) { res += midTerm; }  
 else { throw new PagesLessZeroException("Кол-во страниц без вступления меньше нуля!"); }  
 }  
 return res;  
 }  
  
 @Override  
 public String toString() {  
 String res = "Название серии: " + **name** + "\n";  
 res += "Тип серии: Серия сочинений\n";  
 res += "Кол-во сочинений: " + **pages**.**length** + "\n";  
 res += "Кол-во страниц в сочинениях:\n";  
 for (int page : **pages**) res += page + " ";  
 res += '\n';  
 res += "Кол-во вводных страниц: " + **introduce** + "\n";  
  
 return res;  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (o.getClass() == this.getClass()) {  
 Essays cmp = (Essays)o;  
 if (**name**.equals(cmp.**name**)) {  
 if (**introduce** == cmp.getIntroduce()) {  
 if (**pages**.**length** == cmp.getLen()) {  
 for (int i = 0; i < **pages**.**length**; ) {  
 try {  
 if (**pages**[i] == cmp.getElement(i)) { i++; }  
 else { return false; }  
 }  
 catch (Exception e) { return false; }  
 } return true;  
 } else { return false; }  
 } else { return false; }  
 } else { return false; }  
 }  
 else { return false; }  
 }  
  
 @Override  
 public int hashCode() {  
 return **introduce** + **name**.hashCode() + **pages**.**length** \* 1000;  
 }  
  
}

-------Main.java-------

import javax.naming.spi.DirectoryManager;  
import java.io.\*;  
import java.nio.file.Files;  
import java.nio.file.Path;  
import java.nio.file.Paths;  
import java.util.Scanner;  
  
public class Main {  
  
 private static final Scanner *scan* = new Scanner(System.*in*);  
  
 public static Int[] *array*;  
 public static void main(String[] s) {  
 System.*out*.println("====ЛАБОРАТОРНАЯ РАБОТА №4====");  
 while (*array* == null) {  
 try {  
 System.*out*.println("====СОЗДАНИЕ МАССИВА====");  
 System.*out*.print("Введите размерность массива: ");  
 int count = Integer.*parseInt*(*scan*.nextLine());  
 *array* = new Int[count];  
 } catch (NumberFormatException e) {  
 System.*out*.println("====ОШИБКА====");  
 System.*out*.println("Неверный формат ввода!\n");  
 } catch (NegativeArraySizeException e) {  
 System.*out*.println("====ОШИБКА====");  
 System.*out*.println("Размерность не может быть отрицательной\n");  
 }  
 }  
 for (int i = 0; i < *array*.length;) {  
 System.*out*.println("1. Серия сочинений");  
 System.*out*.println("2. Серия статей");  
 System.*out*.printf("Выберите тип %d-го элемента: ", i);  
 String name = *scan*.nextLine();  
 System.*out*.println();  
 switch (name) {  
 case "1":  
 try {  
 System.*out*.print("Введите название серии: ");  
 name = *scan*.nextLine();  
 System.*out*.print("Введите количество сочинений в серии: ");  
 int count = Integer.*parseInt*(*scan*.nextLine());  
 System.*out*.print("Введите, сколько вступительных страниц в серии: ");  
 int introduce = Integer.*parseInt*(*scan*.nextLine());  
  
 *array*[i] = new Essays(count, name, introduce);  
 i++;  
 System.*out*.println("====УСПЕШНО====\n");  
 } catch (NumberFormatException e) {  
 System.*out*.println("====ОШИБКА====");  
 System.*out*.println("Неверный формат ввода!\n");  
 } catch (RuntimeException e) {  
 System.*out*.println("====ОШИБКА====");  
 System.*out*.println(e.getMessage() + "\n");  
 }  
 break;  
 case "2":  
 try {  
 System.*out*.print("Введите название серии: ");  
 name = *scan*.nextLine();  
 System.*out*.print("Введите количество сочинений в серии: ");  
 int count = Integer.*parseInt*(*scan*.nextLine());  
 System.*out*.print("Введите, сколько вступительных страниц в серии: ");  
 int introduce = Integer.*parseInt*(*scan*.nextLine());  
  
 *array*[i] = new Articles(count, name, introduce);  
 i++;  
 System.*out*.println("====УСПЕШНО====\n");  
 } catch (NumberFormatException e) {  
 System.*out*.println("====ОШИБКА====");  
 System.*out*.println("Неверный формат ввода!\n");  
 } catch (RuntimeException e) {  
 System.*out*.println("====ОШИБКА====");  
 System.*out*.println(e.getMessage() + "\n");  
 }  
 break;  
 default:  
 System.*out*.println("====ОШИБКА====");  
 System.*out*.println("Введено не правильное значение!\n");  
 break;  
 }  
 }  
 boolean end = false;  
 while (!end) {  
 System.*out*.println("=========================МЕНЮ=========================");  
 System.*out*.println("1. Работа с байтовым потоком");  
 System.*out*.println("2. Работа с текстовым потоком");  
 System.*out*.println("3. Сериализация");  
 System.*out*.println("4. Форматный ввод/вывод");  
 System.*out*.println("5. Вывод массива в консоль");  
 System.*out*.println("0. Выход из программы");  
 System.*out*.print("Выберите пункт меню: ");  
 switch (*scan*.nextLine()) {  
 case "1":  
 System.*out*.println("\n====БАЙТОВЫЙ ПОТОК====");  
 System.*out*.println("1. Записать массив в байтовый поток");  
 System.*out*.println("2. Считать массив из байтового потока");  
 System.*out*.println("0. Назад в меню");  
 System.*out*.print("Выберите пункт меню: ");  
 String menu = *scan*.nextLine();  
 if (menu.equals("1")) {  
 try {  
 *binaryOutput*();  
 System.*out*.println("====УСПЕШНО====");  
 } catch (IOException e) {  
 System.*out*.println("====ОШИБКА====");  
 System.*out*.println("Возникла ошибка.\n" + e.getLocalizedMessage());  
 }  
 } else if (menu.equals("2")) {  
 try {  
 *array* = *binaryInput*();  
 System.*out*.println("====УСПЕШНО====");  
 } catch (IOException e) {  
 System.*out*.println("====ОШИБКА====");  
 System.*out*.println("Возникла ошибка.\n" + e.getLocalizedMessage());  
 }  
 } else {  
 System.*out*.println("====ОШИБКА====");  
 System.*out*.println("Введено не правильное значение!\n");  
 }  
 break;  
 case "2":  
 System.*out*.println("\n====ТЕКСТОВЫЙ ПОТОК====");  
 System.*out*.println("1. Записать массив в текстовый поток");  
 System.*out*.println("2. Считать массив из текстового потока");  
 System.*out*.println("0. Назад в меню");  
 System.*out*.print("Выберите пункт меню: ");  
 switch (*scan*.nextLine()) {  
 case "1":  
 try {  
 *textOutput*();  
 System.*out*.println("====УСПЕШНО====");  
 } catch (IOException e) {  
 System.*out*.println("====ОШИБКА====");  
 System.*out*.println("Возникла ошибка.\n" + e.getLocalizedMessage());  
 }  
 break;  
 case "2":  
 try {  
 *array* = *textInput*();  
 System.*out*.println("====УСПЕШНО====");  
 } catch (IOException e) {  
 System.*out*.println("====ОШИБКА====");  
 System.*out*.println("Возникла ошибка.\n" + e.getLocalizedMessage());  
 }  
 break;  
 }  
 break;  
 case "3":  
 System.*out*.println("\n====СЕРИАЛИЗАЦИЯ====");  
 System.*out*.println("1. Сериализовать массив");  
 System.*out*.println("2. Десериализовать массив");  
 System.*out*.println("0. Назад в меню");  
 System.*out*.print("Выберите пункт меню: ");  
 switch (*scan*.nextLine()) {  
 case "1":  
 try {  
 *serialize*();  
 System.*out*.println("====УСПЕШНО====");  
 } catch (IOException e) {  
 System.*out*.println("====ОШИБКА====");  
 System.*out*.println("Возникла ошибка.\n" + e.getLocalizedMessage());  
 }  
 break;  
 case "2":  
 try {  
 *array* = *deserialize*();  
 System.*out*.println("====УСПЕШНО====");  
 } catch (IOException | ClassNotFoundException e) {  
 System.*out*.println("====ОШИБКА====");  
 System.*out*.println("Возникла ошибка.\n" + e.getLocalizedMessage());  
 }  
 break;  
 }  
 break;  
 case "4":  
 System.*out*.println("\n====ФОРМАТНЫЙ ВВОД/ВЫВОД====");  
 System.*out*.println("1. Форматный вывод в файл");  
 System.*out*.println("2. Форматный ввод из файла");  
 System.*out*.println("0. Назад в меню");  
 System.*out*.print("Выберите пункт меню: ");  
 switch (*scan*.nextLine()) {  
 case "1":  
 try {  
 *writeFormat*();  
 System.*out*.println("====УСПЕШНО====");  
 } catch (Exception e) {  
 System.*out*.println("====ОШИБКА====");  
 System.*out*.println("Возникла ошибка.\n" + e.getLocalizedMessage());  
 }  
 break;  
 case "2":  
 try {  
 *array* = *readFormat*();  
 System.*out*.println("====УСПЕШНО====");  
 } catch (Exception e) {  
 System.*out*.println("====ОШИБКА====");  
 System.*out*.println("Возникла ошибка.\n" + e.getLocalizedMessage());  
 }  
 break;  
 }  
 break;  
 case "5":  
 *PrintOutArray*(*array*);  
 break;  
 case "0":  
 System.*out*.println("====ЗАВЕРШЕНИЕ РАБОТЫ====");  
 end = true;  
 break;  
 default:  
 System.*out*.println("====ОШИБКА====");  
 System.*out*.println("Введено не правильное значение!\n");  
 break;  
 }  
 }  
  
  
  
 }  
 private static Int[] readFormat() throws IOException {  
  
 System.*out*.println("\n====ФОРМАТНЫЙ ВВОД====");  
 File file = *CreateNewFile*("r", ".txt");  
 if (file.exists() && file.canRead()) {  
 Scanner in = new Scanner(file);  
 Int[] res;  
 int arrayLength;  
 if (in.hasNextInt()) {  
 arrayLength = Integer.*parseInt*(in.nextLine());  
 res = new Int[arrayLength];  
 for (int i = 0; i < arrayLength; i++) {  
 res[i] = IOMethods.*readFormatInt*(in);  
 }  
 return res;  
 }  
 in.close();  
 }  
 throw new IOException();  
 }  
 private static void writeFormat() throws IOException {  
 System.*out*.println("\n====ФОРМАТНЫЙ ВЫВОД====");  
 File file = *CreateNewFile*("w", ".txt");  
 while (!(file.exists() && file.canWrite())) {  
 if (file.delete()) {  
 if (!file.createNewFile()) {  
 throw new IOException("Ошибка создания файла");  
 }  
 }  
 }  
 Writer out = new FileWriter(file);  
 out.write(*array*.length + "\n");  
 for (Int el: *array*) {  
 IOMethods.*writeFormatInt*(el, out);  
 }  
 try {  
 out.close();  
 } catch (IOException e) {e.getLocalizedMessage();}  
}  
 private static Int[] deserialize() throws IOException, ClassNotFoundException {  
 System.*out*.println("\n====ДЕСЕРИАЛИЗАЦИЯ====");  
 File file = *CreateNewFile*("r", ".bin");  
 if (file.exists() && file.canRead()) {  
 InputStream in = new FileInputStream(file);  
 int arrayLength = in.read();  
 Int[] res = new Int[arrayLength];  
 for (int i = 0; i < arrayLength; i++) {  
 res[i] = IOMethods.*deserializeInt*(in);  
 }  
 try {  
 in.close();  
 } catch (IOException ignored) {}  
 return res;  
 }  
 else { throw new IOException(); }  
 }  
 private static void serialize() throws IOException {  
 System.*out*.println("\n====СЕРИАЛИЗАЦИЯ====");  
 File file = *CreateNewFile*("w", ".bin");  
 OutputStream out = new FileOutputStream(file);  
 out.write(*array*.length);  
 for (Int el: *array*) {  
 IOMethods.*serializeInt*(el, out);  
 }  
 try {  
 out.close();  
 } catch (IOException ignored) {}  
 }  
 private static void binaryOutput() throws IOException {  
 System.*out*.println("\n====БИНАРНЫЙ ВЫВОД====");  
 File file = *CreateNewFile*("w", ".bin");  
 OutputStream out = new FileOutputStream(file);  
 out.write(*array*.length);  
 for (Int el: *array*) {  
 IOMethods.*outputInt*(el, out);  
 }  
 try {  
 out.close();  
 } catch (IOException ignored) {}  
  
  
 }  
 private static File CreateNewFile(String way, String type) throws IOException {  
 System.*out*.print("Введите название файла: ");  
 String fileName = *scan*.nextLine();  
 File directory = new File("./data/");  
 if (!directory.exists()) {  
 directory.mkdir();  
 }  
 File file = new File(directory, fileName + type);  
 if (way.equals("w")) {  
 file.createNewFile();  
 while (!(file.exists() && file.canWrite())) {  
 if (file.delete()) {  
 if (!file.createNewFile()) {  
 throw new IOException("Ошибка создания файла");  
 }  
 }  
 }  
 }  
 else {  
 if (file.exists()) {  
 while(!file.canRead()) {  
 try {  
 Thread.*sleep*(100);  
 } catch (InterruptedException ignored) {}  
 }  
 }  
 else {  
 throw new IOException("Ошибка открытия файла");  
 }  
 }  
 return file;  
 }  
 private static Int[] binaryInput() throws IOException {  
 System.*out*.println("\n====БИНАРНЫЙ ВВОД====");  
 File file = *CreateNewFile*("r", ".bin");  
 if (file.exists() && file.canRead()) {  
 InputStream in = new FileInputStream(file);  
 int arrayLength = in.read();  
 Int[] res = new Int[arrayLength];  
 for (int i = 0; i < arrayLength; i++) {  
 res[i] = IOMethods.*inputInt*(in);  
 }  
  
 try {  
 in.close();  
 } catch (IOException ignored) {}  
 return res;  
 }  
 else { throw new IOException(); }  
  
  
 }  
 private static Int[] textInput() throws IOException {  
 System.*out*.println("\n====ТЕКСТОВЫЙ ВВОД====");  
 File file = *CreateNewFile*("r", ".txt");  
 if (file.exists() && file.canRead()) {  
 Reader in = new FileReader(file);  
 StreamTokenizer stream = new StreamTokenizer(in);  
 stream.nextToken();  
 Int[] res;  
 int arrayLength;  
 if (stream.ttype == StreamTokenizer.*TT\_NUMBER*) {  
 arrayLength = (int)stream.nval;  
 res = new Int[arrayLength];  
 for (int i = 0; i < arrayLength; i++) {  
 res[i] = IOMethods.*readInt*(in);  
 }  
 return res;  
 }  
 try {  
 in.close();  
 } catch (IOException e ) {e.getLocalizedMessage();}  
 }  
 throw new IOException();  
 }  
 private static void textOutput() throws IOException {  
 System.*out*.println("\n====ТЕКСТОВЫЙ ВЫВОД====");  
 File file = *CreateNewFile*("w", ".txt");  
 Writer out = new FileWriter(file);  
 out.write(*array*.length + "\n");  
 for (Int el: *array*) {  
 IOMethods.*writeInt*(el, out);  
 out.write("\n");  
 }  
 try {  
 out.close();  
 } catch (IOException e) {e.getLocalizedMessage();}  
 }  
 private static void PrintOutArray(Int[] array) {  
 for (int i = 0; i < array.length; i++) {  
 System.*out*.println("====ЭЛЕМЕНТ #" + (i + 1) + "====");  
 System.*out*.println(array[i].toString());  
 }  
 }  
}

















