|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | МИНОБРНАУКИ РОССИИ | | | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |   Институт Информационных технологий | |
|  | |
| Кафедра Математического обеспечения и стандартизации информационных технологий | |
|  | |
|  | |

|  |  |
| --- | --- |
| **ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 3** | |
| **по дисциплине** | |
| **«**Объектно-ориентированное программирование**»**  **Тема: «Классы. Диаграммы классов в языке UML»** | |
|  | |
| Выполнил студент группы ИКБО-09-18 | Налюшный А.Е. |
| Принял преподаватель | Баранова И.А. |

|  |  |  |
| --- | --- | --- |
| Лабораторная работа выполнена | «\_\_»\_\_\_\_\_\_\_201\_\_ г. | *(подпись студента)* |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_201\_\_ г. | *(подпись руководителя)* |

Москва 2019

Задание 1

1. Условие задачи

Требуется выполнить три упражнения. Результаты каждого упражнения включить в отчет по лабораторной работе.

Все упражнения задания реализовать в одном проекте в методе main, разделяя код упражнений комментариями – условие упражнения.

2. Постановка задачи

**Упражнение 1**. Исследовать возможности класса File по созданию файлов (пустых) и папок программой. Применение конструктора и метода.

* Создать файл в папке приложения с именем MyFile1.txt.
* Проверить появление файла в папке приложения.
* Создать файл с именем MyFile2.txt в корне определенного диска.
* Проверить появление файла в папке.
* Создать файл с именем MyFile3.txt в папке *Имя диска\\Имя папки\\Имя файла*.
* Проверить появление файла в папке.
* Создать папку третьего уровня, например, Первая\\Вторая\\Третья.
* Проверить появление папки.
* Все операции заключить в блок try, с обработчиками исключений IOException, Exception, FileNotFoundException.

**Упражнение 2**. Получить параметры файлов методами класса File.

В пунктах задания использовать объекты, созданные в задании 1.

* Проверить, что вызывающий объект содержит имя файла, а не папки и

отобразить имя файла, вызывающего объекта и его родительскую папку.

* Проверить, что вызывающий объект содержит имя папки, а не файла и

отобразить имя файла, вызывающего объекта.

* Проверить, что в папке приложения существует файл с именем MyFile1.txt.
* Отобразить полный путь к файлу или папке объекта.
* Отобразить размер файла или папки объекта, указать единицу измерения. Прокомментировать вид файла – папка или файл.

**Упражнение 3.** Модификация файловой структуры приложения средствами класса File.

1. Добавить в папку приложения еще одну папку.
2. Сформировать массив файлов, находящихся в папке приложения, используя метод list(). Отобразить содержимое массива.
3. Сформировать массив файлов, находящихся в папке приложения, используя метод listFiles( ). Отобразить содержимое массива. Определить количество папок, содержащихся в файле приложения.
4. Удалить папки и файлы созданные во всех трех упражнениях

3. Разработка тестов



Рис 1. Тестирование первого задания

4. Исходный код программы

**package** task1;  
  
**import** java.io.File;  
**import** java.util.Scanner;  
  
**public class** Main {  
  
 **public static void** main(String[] args) {  
 *//Задание 1* **try** {  
 *//упражнение 1* File MyFile1 = **new** File(**"MyFile1.txt"**);  
 MyFile1.createNewFile();  
 File MyFile2 = **new** File(**"C:\\repo\\MyFile2.txt"**);  
 MyFile2.createNewFile();  
 File MyFile3 = **new** File(**"C:\\Java\\lab4\\MyFile3.txt"**);  
 MyFile3.createNewFile();  
 File folders = **new** File(**"First\\Second\\Third"**);  
 folders.mkdirs();  
  
 *//Упражнение 2* System.***out***.println(**"Введите путь к файлу:"**);  
 Scanner in = **new** Scanner(System.***in***);  
 String way = in.next();  
 File t2 = **new** File(way);  
 System.***out***.println(t2.getName());  
 System.***out***.println(t2.getParent());  
  
 **if** (!way.contains(**"."**)){  
 System.***out***.println(**"Введена папка а не файл"**);  
 }  
  
 File check = **new** File(**"MyFile1.txt"**);  
 **if**(check.exists()){  
 System.***out***.println(**"MyFile1.txt существует"**);  
 }**else**{  
 System.***out***.println(**"MyFile1.txt не существует"**);  
 }  
  
 System.***out***.println(t2.getAbsolutePath());  
  
 System.***out***.println(t2.length() + **" байт"**);  
 **if** (t2.isDirectory()){  
 System.***out***.println(**"объект - папка"**);  
 }**else**{  
 System.***out***.println(**"объект - файл"**);  
 }  
  
 *//Упражнение 3* File ndir = **new** File(**"newDirectory"**);  
 ndir.mkdir();  
 File folder = **new** File(**"C:\\Java\\lab4"**);  
 String[] listOfFiles\_str = folder.list();  
 File[] listOfFiles = folder.listFiles();  
  
 System.***out***.println(**"содержимое папки приложения через list()"**);  
 **for**(String y: listOfFiles\_str){  
 System.***out***.println(y);  
 }  
  
 System.***out***.println(**"содержимое папки приложения через listFiles()"**);  
 **int** fi = 0, dir = 0;  
 **for** (**int** i = 0; i < listOfFiles.**length**; i++) {  
 **if** (listOfFiles[i].isFile()) {  
 System.***out***.println(listOfFiles[i].getName());  
 fi++;  
 } **else if** (listOfFiles[i].isDirectory()) {  
 System.***out***.println(listOfFiles[i].getName());  
 dir++;  
 }  
 }  
 System.***out***.println(**"Количество файлов: "** + fi);  
 System.***out***.println(**"Количество папок: "** + dir);  
  
 MyFile1.delete();  
 MyFile2.delete();  
 MyFile3.delete();  
 folders.delete();  
 folders = **new** File (**"First\\Second"**);  
 folders.delete();  
 folders = **new** File (**"First"**);  
 folders.delete();  
 ndir.delete();  
  
 }**catch** (Exception e){  
 System.***out***.println(**"error"**);  
 }  
 }  
}

Задание 2

1. Условие задачи

1. Создать класс, указанный в варианте. Предусмотреть конструктор с параметрами для создания объекта, контролирующий поступающие данные. Данные состояния объекта вводятся с клавиатуры. Дополнительные методы, указанные в варианте
2. Создать класс по управлению файлом, который должен обеспечить следующие функции:
3. Формирование файла данных
   1. Создать выходной поток для записи объекта в файл, располагая файл в папке приложения. Имя файла пользователь вводит с клавиатуры.
   2. Обеспечить проверку существования файла с таким именем и вывод сообщения о его существовании.
   3. Выполнить запись в файл данных нескольких объектам. Данные для полей объекта вводятся с клавиатуры и записываются в файл. Расположите их в файле так, чтобы потом можно было написать алгоритм чтения из файла данных и формирование объектов.. Помните, что при вводе вы будете вводить данные и строковых и примитивных типов (смешанный ввод).
   4. Закрыть выходной поток.
4. Чтение данных
   1. Создать входной поток для чтения данных из созданного файла.
   2. Создать массив с данными о состоянии объектов, используя оду из коллекций Java: для нечетного варианта ArrayList, для четного – LinkedList.
   3. Прочитать данные из файла и разместить в коллекции те объекты, которые, удовлетворяют критериям дополнительного задания варианта (задание 1).
5. Используя класс RandomAccessFile произвольного доступа к записям файла обеспечить:
   1. Создание нового файла из записей фиксированной длины, содержащий сведения по объектам, прочитанных в массив объектов в задании 4.3. Так как поток RandomAccessFile в качестве источника может иметь только массивы примитивного типа, необходимо предусмотреть структуру записи – строка и ее размер. Для этого расширяйте строковые значения пробелами справа, так, чтобы в сумме с примитивными значениями получить нужный размер. При таком условии метод seek можно использовать для перемещения по записям.
   2. Выполнить над новым файлом операцию задания 2 из дополнительного задания варианта.

2. Определение класса варианта

Класс – Тур

Поля:

Наименование тура, Страна, Город, Условия проживания: Название отеля, класс – отеля, Срок тура (отправление; прибытие), Количество экскурсий в составе тура, Стоимость тура, Туристическая компания, организовавшая тур.

Методы:

1) По наименованию тура определить и вернуть город и страну пребывания.

2) Определить, имеют ли два тура одинаковую стоимость.

**public class** tour {  
 **private** String **name**;  
 **private** String **country**;  
 **private** String **city**;  
 **private** String **hotel\_name**;  
 **private int stars**;  
 **private int duration**;  
 **private int amount**;  
 **private double cost**;  
 **private** String **company**;  
  
 tour(String n, String c, String ci, String hn, **int** s, **int** d, **int** a, **double** cos, String com) **throws** Exception{…}  
 tour(String t){…}  
 @Override  
 **public** String toString() {…}  
  
 **public** String get\_c\_ci(String n){…}  
  
 **public boolean** same\_cost(tour t){**…**}  
  
 **public void** add10 (){**…**}  
}

3. Определение класса для управления файлом

Класс для управления файлом

Поля: путь к файлу

Методы:

1. Запись в файл
2. Чтение с файла и запись в файл туров с указанной компанией
3. Чтение с файла и добавление 10% к стоимости тура по указанному индексу

**public class** Filerw {  
 **private** String **name** = **""**;  
  
 **public void** write(ArrayList<tour> b) **throws** Exception{…}  
  
 **public** ArrayList read(String company) **throws** Exception{…}  
  
 **public void** raf(ArrayList<tour> b, **int** num) **throws** Exception{**…**}  
}

4. Разработка тестов

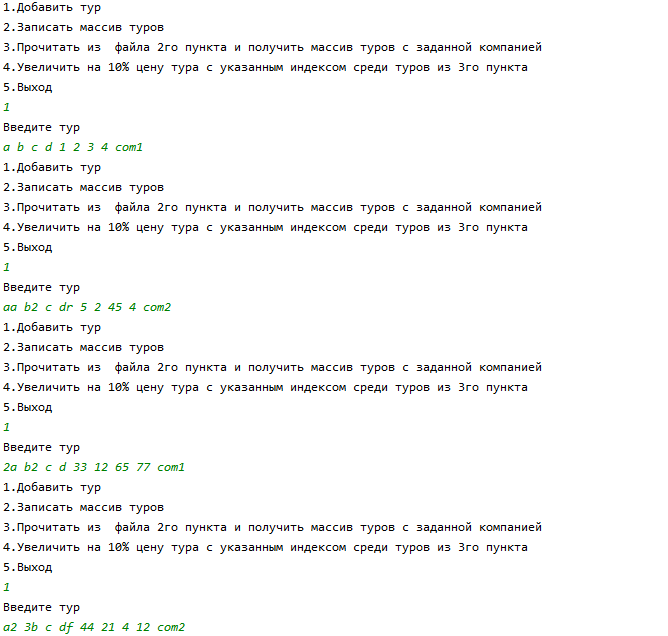


Рис 2. Добавление туров

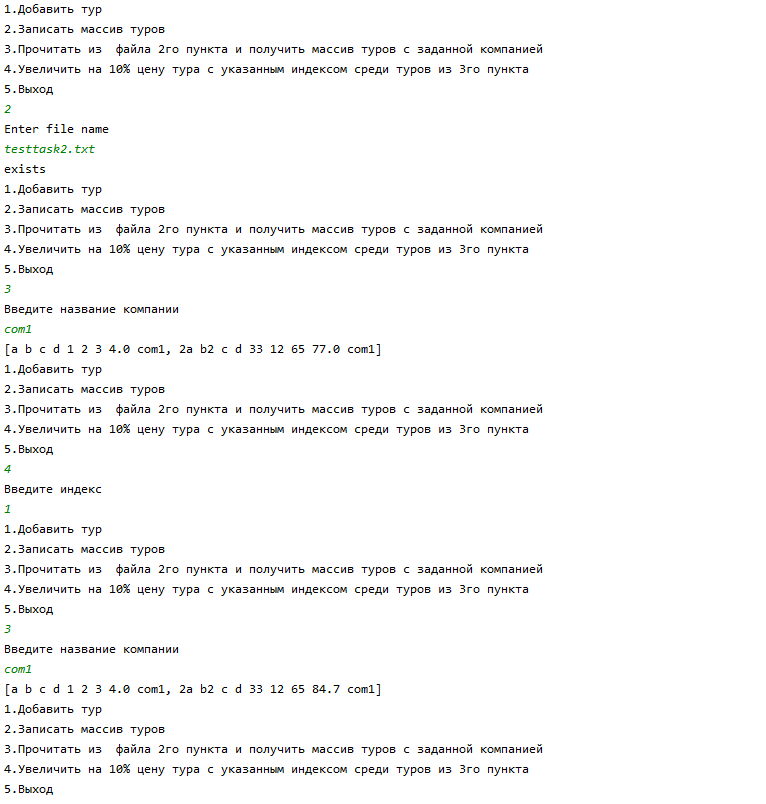


Рис 3. Тестирование второго задания

5. Исходный код программы

1. **Класс tour**

**package** task2;  
  
**public class** tour {  
 **private** String **name**;  
 **private** String **country**;  
 **private** String **city**;  
 **private** String **hotel\_name**;  
 **private int stars**;  
 **private int duration**;  
 **private int amount**;  
 **private double cost**;  
 **private** String **company**;  
  
 tour(String n, String c, String ci, String hn, **int** s, **int** d, **int** a, **double** cos, String com) **throws** Exception{  
 **name** = n;  
 **country** = c;  
 **city** = ci;  
 **hotel\_name** = hn;  
 **stars** = s;  
 **duration** = d;  
 **amount** = a;  
 **cost** = cos;  
 **company** = com;  
 }  
  
 tour(String t){  
 String[] k = t.split(**" "**);  
 **name** = k[0];  
 **country** = k[1];  
 **city** = k[2];  
 **hotel\_name** = k[3];  
 **stars** = Integer.*parseInt*(k[4]);  
 **duration** = Integer.*parseInt*(k[5]);  
 **amount** = Integer.*parseInt*(k[6]);  
 **cost** = Double.*parseDouble*(k[7]);  
 **company** = k[8];  
 }  
  
 @Override  
 **public** String toString() {  
 **return name** + **' '** + **country** + **' '** + **city** + **' '** + **hotel\_name** + **' '** + **stars** + **' '** + **duration** + **' '** + **amount** + **' '** + **cost** + **' '** + **company**;  
 }  
  
 **public** String get\_c\_ci(String n){  
 **if** (n == **name**){  
 **return city** + **" "** + **country**;  
 }  
 **return ""**;  
 }  
  
 **public boolean** same\_cost(tour t){  
 **return cost** == t.**cost**;  
 }  
  
 **public void** add10 (){  
 **cost** += **cost**/10;  
 }  
}

1. **Класс Filerw**

**package** task2;  
  
**import** java.io.\*;  
**import** java.util.ArrayList;  
**import** java.util.Scanner;  
  
**public class** Filerw {  
 **private** String **name** = **""**;  
  
 **public void** write(ArrayList<tour> b) **throws** Exception{  
 System.***out***.println(**"Enter file name"**);  
 Scanner in = **new** Scanner(System.***in***);  
 **name** = in.next();  
  
 DataOutputStream dataOut = **new** DataOutputStream(**new** FileOutputStream(**name**));  
  
 **for**(tour t : b){  
 **byte**[] bbb = t.toString().getBytes(**"UTF-8"**);  
 dataOut.write(bbb);  
 dataOut.write(**"\n"**.getBytes(**"UTF-8"**));  
 }  
  
 File f = **new** File(**name**);  
 **if** (f.exists()){  
 System.***out***.println(**"exists"**);  
 }**else**{  
 System.***out***.println(**"don't exist"**);  
 }  
  
 dataOut.close();  
 }  
  
 **public** ArrayList read(String company) **throws** Exception{  
 DataInputStream dataIn = **new** DataInputStream(**new** FileInputStream(**name**));  
 ArrayList<tour> b = **new** ArrayList<tour>();  
 **while** (dataIn.available() > 0){  
  
 String[] k = dataIn.readLine().split(**" "**);  
 **if** (k[8].equals(company)) {  
 String n = k[0];  
 String c = k[1];  
 String ci = k[2];  
 String hn = k[3];  
 **int** s = Integer.*parseInt*(k[4]);  
 **int** d = Integer.*parseInt*(k[5]);  
 **int** a = Integer.*parseInt*(k[6]);  
 Double cos = Double.*parseDouble*(k[7]);  
 String com = k[8];  
 tour t = **new** tour(n, c, ci, hn, s, d, a, cos, com);  
  
 b.add(t);  
 }  
 }  
 dataIn.close();  
 **return** b;  
 }  
  
 **public void** raf(ArrayList<tour> b, **int** num) **throws** Exception{  
 **name** +=**"1.txt"**;  
 DataOutputStream dataOut = **new** DataOutputStream(**new** FileOutputStream(**name**));  
  
 **for**(tour t : b){  
 **byte**[] bbb = t.toString().getBytes(**"UTF-8"**);  
 dataOut.write(bbb);  
 dataOut.write(**"\r\n"**.getBytes(**"UTF-8"**));  
 }  
 dataOut.close();  
  
 RandomAccessFile raf = **new** RandomAccessFile(**name**, **"rw"**);  
 raf.seek(0);  
 String sss = **""**;  
 **for** (**int** i = 0; i<=num; i++){  
 sss = raf.readLine();  
 }  
 tour t = **new** tour(sss);  
 t.add10();  
 **byte**[] bbb = t.toString().getBytes(**"UTF-8"**);  
  
 raf.seek(0);  
 **for** (**int** i = 0; i<num; i++){  
 raf.readLine();  
 }  
 raf.write(bbb);  
 raf.close();  
 }  
}

1. **Класс Main**

**package** task2;  
  
**import** java.util.ArrayList;  
**import** java.util.Scanner;  
  
**public class** Main {  
  
 **public static void** main(String[] args) {  
 *//Задание 2* **try** {  
  
 *// a b c d 1 2 3 4 com1  
 // aa b2 c dr 5 2 45 4 com2  
 // 2a b2 c d 33 12 65 77 com1  
 // a2 3b c df 44 21 4 12 com2* **boolean** c = **true**;  
 Scanner in = **new** Scanner(System.***in***);  
 ArrayList<tour> t1 = **new** ArrayList<tour>();  
 ArrayList t2 = **new** ArrayList<tour>();  
 Filerw fi = **new** Filerw();  
 **while**(c){  
 System.***out***.println(**"1.Добавить тур\n2.Записать массив туров\n3.Прочитать из файла 2го пункта и получить массив туров с заданной компанией"** +  
 **"\n4.Увеличить на 10% цену тура с указанным индексом среди туров из 3го пункта\n5.Выход"**);  
 **int** a = Integer.*parseInt*(in.next());  
 **switch** (a){  
 **case** 1:  
 System.***out***.println(**"Введите тур "**);  
 tour t = **new** tour(in.next(), in.next(), in.next(), in.next(), Integer.*parseInt*(in.next()), Integer.*parseInt*(in.next()), Integer.*parseInt*(in.next()), Double.*parseDouble*(in.next()), in.next());  
 t1.add(t);  
 **break**;  
 **case** 2:  
 **if** (t1.isEmpty()){  
 System.***out***.println(**"Массив пуст"**);  
 **break**;  
 }  
 fi.write(t1);  
 **break**;  
 **case** 3:  
 **if** (t1.isEmpty()){  
 System.***out***.println(**"Массив пуст"**);  
 **break**;  
 }  
 System.***out***.println(**"Введите название компании"**);  
 String s = in.next();  
 t2 = fi.read(s);  
 System.***out***.println(t2.toString());  
 **break**;  
 **case** 4:  
 **if** (t2.isEmpty()){  
 System.***out***.println(**"Массив пуст"**);  
 **break**;  
 }  
 System.***out***.println(**"Введите индекс"**);  
 **int** i = in.nextInt();  
 fi.raf(t2, i);  
  
 **break**;  
 **case** 5:  
 c = **false**;  
 **break**;  
  
 }  
 }  
  
   
 }**catch** (Exception e){  
 System.***out***.println(**"Ошибка"**);  
 }  
 }  
}

Задание 3

1. Условие задачи

Выполнить три упражнения. Каждое упражнение в отдельном методе основного класса проекта.

2. Постановка задачи

Упражнение 1. Чтение из одного файла текстового файла и запись в другой

* Создайте два текстовых файл средствами среды или другого текстового редактора: T1.txt и T2.txt. В файл T1.txt запишите несколько символов. Файл T2.txt не заполняйте.
* Определите один поток для чтения из файла T1.txt, а второй выходной с добавлением символов в конец.
* Прочитайте из файла T1.txt значение методом read и перепишите его в файл T2.txt методом write. Перепишите все значения из файла T1.txt в T2.txt. Для выполнения операции создайте цикл, пока метод read не вернет -1.
* При разработке кода используйте исключения на корректность операций ввода и вывода IOException.
* Выполните отладку программы упражнения. Проверьте заполнение файла T2.txt.

Упражнение 2. Применение буферизированных потоков для чтения и записи текстовых файлов.

Используйте буфер в 128 байтов.

* Создайте текстовый файл A.txt и запишите в него 512 символов.
* Определите поток inb класса BufferReader и свяжите его с созданным файлом, установив размер буфера равным 128 байтам (файл должен быть прочитан за четыре раза).
* Создайте пустой текстовый файл B.txt
* Создайте поток outb класса BufferWriter и свяжите его с файлом B.txt установив размер буфера 128 байтов.
* Определите буферную переменную buf размером 128 символов – массив из 128 символов.
* Перепишите все данные из файла A.txt в файл B.txt, используя буфер и методы: read(buf) и write(buf). Располагая каждый вывод буфера на отдельной строке файла.

Упражнение 3. Настройка кодировки символов для входного и выходного потоков.

Создайте файл с именем A.txt, используя возможности среды (щелкните по csr правойкнопкой и в контекстном меню выполните new🡪File введите имя). Подготовьте в файле A.txt предложение на русском языке. Вы подготовили текст в кодировке UTF-8.

Определите в поток in чтения из файла

BufferedReader in = new BufferedReader(new InputStreamReader(new FileInputStream("Путь к файлу A.txt"), "Cp1251"));

Отобразите название кодировки, действующей в системе

System.out.println(Charset.defaultCharset().name());

Реализуйте алгоритм чтения из файла текста через строковую переменну, пока вся информация не будет прочитана.

Выполните программу.

Что – то не так?

Измените кодовую страницу "Cp1251" на "UTF-8"

Создайте другой текстовый файл с помощью редактора Блокнот.

Укажите его как источник данных для потока in.

Установите кодировку, в которой Блокнот подготовил текст.

Выполните программу.

3. Разработка тестов

Упражнение 1: после выполнения метода содержимое файла T1.txt скопировалось в T2.txt

Упражнение 2: после выполнения метода содержимое файла A.txt скопировалось в B.txt

Упражнение 3:

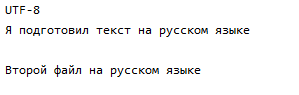


Рис 4. Тестирование третьего упражнения второго задания

4. Исходный код программы

**package** task3;  
  
**import** java.io.\*;  
**import** java.nio.charset.Charset;  
**import** java.util.Arrays;  
  
**public class** Main {  
  
 **public static void** num1 () **throws** IOException {  
 DataOutputStream dataOut = **new** DataOutputStream(**new** FileOutputStream(**"T2.txt"**));  
 DataInputStream dataIn = **new** DataInputStream(**new** FileInputStream(**"T1.txt"**));  
 **int** i;  
 **while** ((i = dataIn.read()) != -1){  
 dataOut.write(i);  
 }  
 dataIn.close();  
 dataOut.close();  
 }  
  
 **public static void** num2() **throws** Exception{  
 BufferedReader inb = **new** BufferedReader(**new** FileReader(**"A.txt"**), 128);  
 BufferedWriter outb = **new** BufferedWriter(**new** FileWriter(**"B.txt"**));  
 **char**[] buf = **new char**[128];  
  
 **while** ((inb.read(buf)) != -1){  
 outb.write(buf);  
 outb.newLine();  
 }  
 outb.flush();  
 outb.close();  
 inb.close();  
 }  
  
 **public static void** num3() **throws** Exception{  
 BufferedReader in = **new** BufferedReader(**new** InputStreamReader(**new** FileInputStream(**"src\\A.txt"**), **"UTF-8"**));  
 System.***out***.println(Charset.*defaultCharset*().name());  
 String s = **""**;  
 **while** ((s = in.readLine()) != **null**){  
 System.***out***.println(s);  
 }  
 System.***out***.println();  
 in = **new** BufferedReader(**new** InputStreamReader(**new** FileInputStream(**"src\\B.txt"**), **"UTF-16"**));  
 **while** ((s = in.readLine()) != **null**){  
 System.***out***.println(s);  
 }  
 }  
  
 **public static void** main(String[] args) {  
 *//Задание 2* **try** {  
 *num1*();  
 *num2*();  
 *num3*();  
  
 } **catch** (Exception e) {  
 System.***out***.println(**"Ошибка"**);  
 }  
 }  
}

Задание 4

1. Условие задачи

Требуется разработать три класса:

* Управление тестовым файлом. ClassTextFile - исходный файл с данными для объектов.
* Класс информационного объекта на основании одного класса из модели варианта лабораторной работы 2 – имя задайте самостоятельно.
* Управление файлами с сериализованными объектами - ClassSeriyazableFile.

Разработать приложение, демонстрирующее выполнение операций над файлом с применением потоков объектов.

2. UML-диаграмма классов

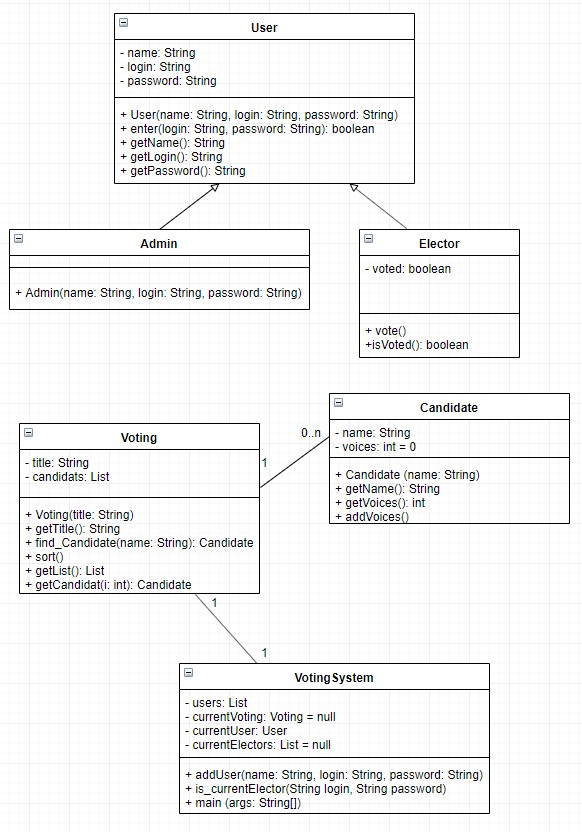


Рис. 5 UML-диаграмма классов

3. Разработка тестов

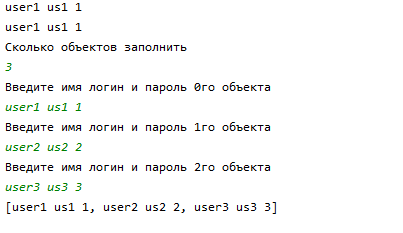


Рис. 6 Тестирование четвертого задания

5. Исходный код программы

1. **Класс User**

**package** task4;  
  
  
**import** java.util.Scanner;  
**import** java.io.Serializable;  
  
**public class** User **implements** Serializable{  
 **private** String **name**;  
 **private** String **login**;  
 **private** String **password**;  
  
 User(String name, String login, String password){  
 **this**.**name** = name;  
 **this**.**login** = login;  
 **this**.**password** = password;  
 }  
  
 User(){  
 **name** = **""**;  
 **login** = **""**;  
 **password** = **""**;  
 }  
  
 **public boolean** enter(String login, String password){  
 **if** (**this**.**login**.equals(login) && **this**.**password**.equals(password))  
 **return true**;  
 **return false**;  
 }  
  
 **public** String toString(){  
 **return name** + **" "** + **login** + **" "** + **password**;  
 }  
  
 **public void** vvodsconsoli (){  
 Scanner i = **new** Scanner(System.***in***);  
 **name** = i.next();  
 **login** = i. next();  
 **password** = i.next();  
 }  
  
 **public** String getName() {  
 **return name**;  
 }  
 **public** String getLogin() {  
 **return login**;  
 }  
 **public** String getPassword() {  
 **return password**;  
 }  
  
 **public void** setName(String name) {  
 **this**.**name** = name;  
 }  
 **public void** setLogin(String login) {  
 **this**.**login** = login;  
 }  
 **public void** setPassword(String password) {  
 **this**.**password** = password;  
 }  
}

1. **Класс ClassTextFile**

**package** task4;  
  
**import** java.io.File;  
**import** java.util.Scanner;  
**import** java.io.Serializable;  
  
**public class** ClassTextFile **implements** Serializable{  
 String **file** = **""**;  
  
 ClassTextFile(String f){  
 **if** (**new** File(f).exists()){  
 **file** = f;  
 }**else**{  
 System.***out***.println(**"Файла не существует"**);  
 }  
 }  
  
 **public void** fill(User s) **throws** Exception{  
 Scanner i = **new** Scanner(**new** File(**file**));  
  
 **while** (i.hasNextLine()){  
 s.setName(i.next());  
 s.setLogin(i.next());  
 s.setPassword(i.next());  
 }  
 i.close();  
 }  
}

1. **Класс ClassSeriyazableFile**

**package** task4;  
  
**import** java.io.\*;  
**import** java.util.LinkedList;  
**import** java.util.Scanner;  
  
**public class** ClassSeriyazableFile **implements** Serializable{  
  
 String **file** = **""**;  
 LinkedList<User> **users** = **new** LinkedList<User>();  
  
 ClassSeriyazableFile(String f){  
 **if** (**new** File(f).exists()){  
 **file** = f;  
 }**else**{  
 System.***out***.println(**"Файла не существует"**);  
 }  
 }  
  
 **public void** record(User us) **throws** Exception{  
 FileOutputStream out = **new** FileOutputStream(**file**);  
 ObjectOutputStream oos = **new** ObjectOutputStream(out);  
 oos.writeObject(us);  
 oos.close();  
 }  
  
 **public** User read() **throws** Exception{  
 FileInputStream in = **new** FileInputStream(**file**);  
 ObjectInputStream oin = **new** ObjectInputStream(in);  
 User us = **new** User();  
 us = (User)oin.readObject();  
 **return** us;  
 }  
  
 **public void** collection () **throws** Exception{  
 System.***out***.println(**"Сколько объектов заполнить"**);  
 Scanner is = **new** Scanner(System.***in***);  
 **int** n = is.nextInt();  
 **for** (**int** i = 0; i<n; i++){  
 System.***out***.println(**"Введите имя логин и пароль "** + i + **"го объекта"**);  
 User u = **new** User();  
 u.vvodsconsoli();  
 **users**.add(u);  
 }  
 FileOutputStream out = **new** FileOutputStream(**file**);  
 ObjectOutputStream oos = **new** ObjectOutputStream(out);  
 oos.writeObject(**users**);  
 }  
  
 **public void** readfile() **throws** Exception{  
 FileInputStream in = **new** FileInputStream(**file**);  
 ObjectInputStream oin = **new** ObjectInputStream(in);  
  
 **users** = (LinkedList)oin.readObject();  
  
 }  
  
 **public** LinkedList<User> gerArray(){  
 **return users**;  
 }  
  
}

1. **Класс Main**

**package** task4;  
  
**public class** Main {  
 **public static void** main(String[] args) {  
 *//Задание 2* **try** {  
 ClassTextFile ctf = **new** ClassTextFile(**"task4.txt"**);  
 User us1 = **new** User();  
 ctf.fill(us1);  
 System.***out***.println(us1.toString());  
  
 ClassSeriyazableFile csf = **new** ClassSeriyazableFile(**"task42.txt"**);  
 csf.record(us1);  
 us1 = csf.read();  
 System.***out***.println(us1.toString());  
 csf.collection();  
 csf.readfile();  
 System.***out***.println(csf.gerArray().toString());  
  
 } **catch** (Exception e) {  
 System.***out***.println(**"Ошибка"**);  
 }  
  
 }  
}

Работа с git

Инициализация, добавление измененных и новых файлов в git и коммит изменений

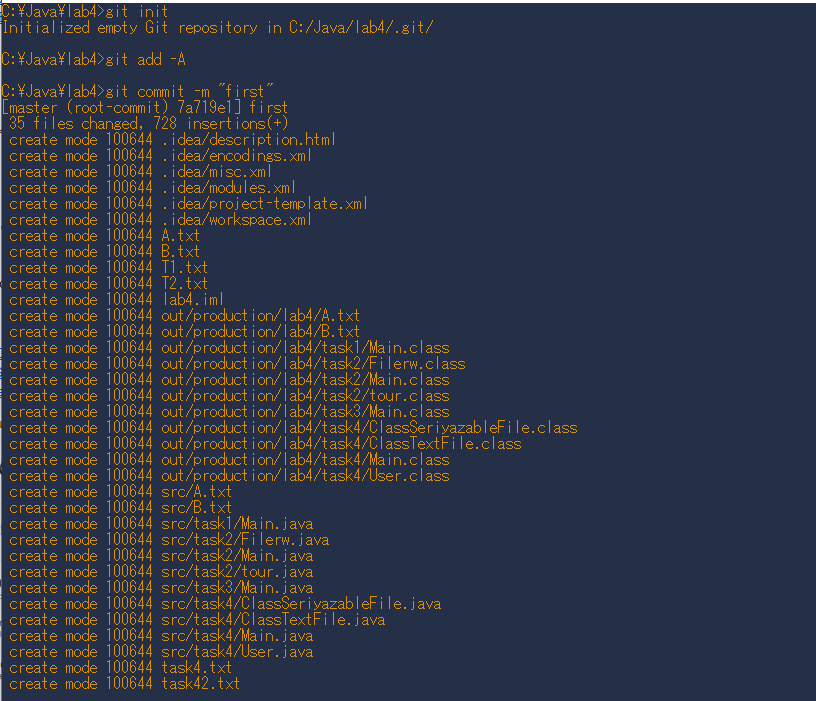


Рис. 7 Инициализация, добавление и коммит

Отправление изменений на удаленный репозиторий

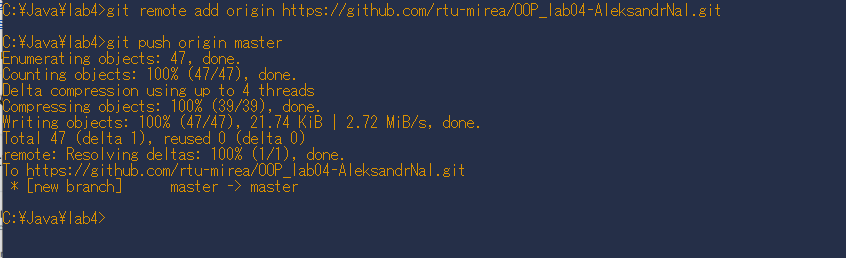


Рис 8.отправка изменений на удаленный репозиторий

Проект в удаленном репозитории.

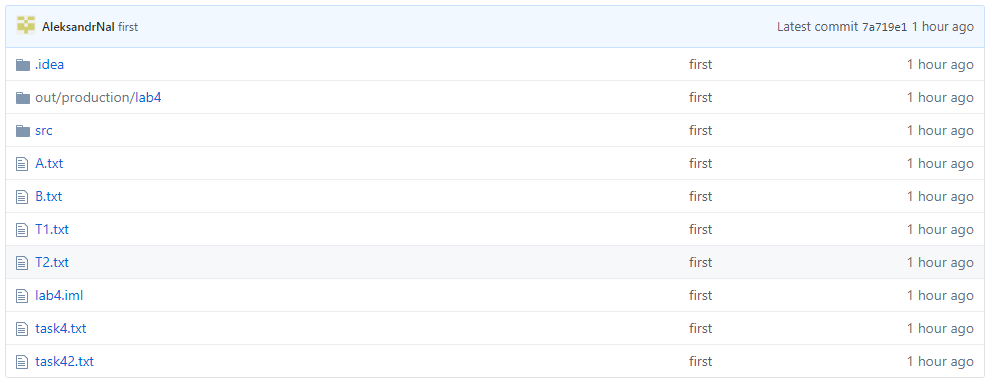
<https://github.com/rtu-mirea/OOP_lab04-AleksandrNal> 

Рис 9. Проект в удаленном репозитории