|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | 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 № 4** | |
| **по дисциплине** | |
| **«**Объектно-ориентированное программирование**»**  **Тема: «Ввод и вывод в Java. Пакет java.io и его классы»** | |
|  | |
| Выполнил студент группы ИКБО-07-18 | Мишанов А.Б. |
| Принял преподаватель | Колесников С.Н. |

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

**Задание 1**

Класс файл

**Требования к заданию:**

Задание. 1. Управление файловой структурой Java – класс File.

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.

Упражнения по управлению файловой структурой

**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

**Тестирование:**

В качестве результата тестирования приведено состояние директории, в которой находится приложение в момент перед удалением всех созданных файлов и папок, а также консольный вывод информации, требуемой по заданию

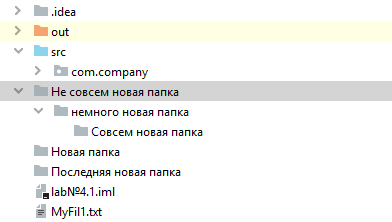


Рисунок 1 – Состояние директории перед удалением всех созданных файлов

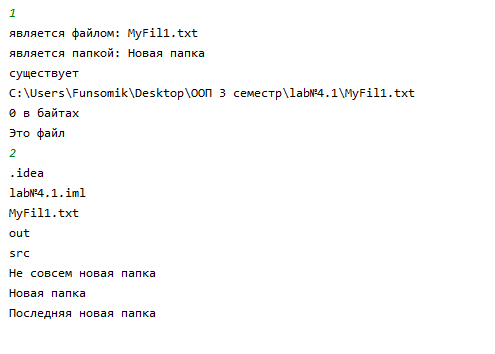


Рисунок 2 – Вывод информации, необходимой по заданию

**Задание 2:**

Использование классов байтовых потоков b потоков DataInputStream и DataOutputStream для записи в файл состояния объектов и файла RandomAccesFile

**Требования к заданию:**

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.

|  |  |  |
| --- | --- | --- |
| Вариант | Структура класса | Дополнительное задание |
| 1 | Класс Студент |  |
| *Поля:* Номер зачетной книжки, Фамилия, Имя, Дата рождения, Номер телефона, Дата зачисления, Номер группы | 1. Список студентов заданной пользователем группы |
| *Дополнительные методы:*   1. По номеру зачетной книжки вернуть фамилию и дату рождения. 2. Определить, зачислены ли два студента в одну дату. | 1. Удалить студента с заданным номером зачетной книжки, записав на место удаленной записи последнюю запись. |

**Тестирование:**

Тестирование приложения показало, что класс выполняет все необходимые функции. Информация об объекте корректно записывается в объект и обратно. Также реализована запись в файл и изменение файла с помощью класса RandomAccessFile

**Задание 3**

Исследование возможностей символьных потоков ввода и вывода

**Требования к заданию:**

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.

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.

**Тестирование:**

Тестирование показало корректность выполнения задания.

**Задание 4**

Сериализация объектов

### Требования по выполнению задания

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.

### Детализация требований по выполнению задания

1. В модели лабораторной работы 2 выберите один класс, объекты которого требуют больше данных. Создайте текстовый файл с данными для заполнения нескольких объектов. Данные разного типа лучше располагать на отдельных строках файла.
2. Разработайте класс ClassTextFile для управления текстовым файлом исходных данных.
   1. Поля

Путь к файлу

* 1. Методы
* Конструктор с параметром – путь к файлу, с проверкой существования файла.
* Заполнить объект данными файла и вернуть его. Для этого использовать поток Scanner для считывания из текстового файла данных для заполнения полей объекта. Метод считывания должен соответствовать типу поля объекта.

1. Разработать класс для представления объекта, вашего варианта лабораторной работы 2.
   1. Поля класса

* свойства выбранного объекта.
  1. Методы
* Ввод данных для объекта с консоли.
* Вернуть объект.
* Залить данные в объект.

1. Разработать класс, обеспечивающий запись в файл и чтение из файла, используя классы FileInputStream, FileOutputStream, и классы, обеспечивающие сериализацию объектов ObjectOutputStream (сериализация объектов) и ObjectInputStream (десериализация объектов).
   1. Поля класса

* Путь к файлу сериализованных объектов.
* Коллекция объектов.
  1. Методы:
* Конструктор класса – должен инициализировать путь переданным параметром.
* Запись одного сериализованного объекта в файл FileOutputStream посредством потока ObjectOutputStream.
* Чтение одного объекта из файла, используя потоки FileInputStream, ObjectInputStream.
* Метод get для объекта.
* Создание коллекции объектов, запись всей коллекции в файл. Коллекции объектов: для четного варианта ArrayList, для нечетного LinkedList.
  + Чтение объектов из файла в коллекцию.
  + Метод Get для коллекции объектов.
  + Вывод содержимого файла и коллекции.

1. Разработать приложение, демонстрирующее выполнение всех операций над файлами объектов, предусмотренных в задании.

**Тестирование:**

Тестирование показало корректность выполнения задания.

**Исходный код**

Класс Main задание 1

**package** com.company;  
**import** java.awt.\*;  
**import** java.io.\*;  
**import** java.io.IOException;  
**import** java.nio.channels.ScatteringByteChannel;  
**import** java.util.Scanner;  
  
**public class** Main {  
  
 **public static void** main(String[] args) **throws** IOException {  
 **try** {  
 Scanner stop\_u\_violated\_the\_low = **new** Scanner(System.***in***);  
 File F1 = **new** File(**"MyFil1.txt"**);  
 F1.createNewFile();  
 File folder = **new** File(**"Новая папка"**);  
 folder.mkdir();  
 String dir = System.*getProperty*(**"user.dir"**);  
 File folderin = **new** File(dir + **"/Не совсем новая папка/немного новая папка/"**, **"Совсем новая папка"**);  
 folderin.mkdirs();  
  
 *//упражнение 2* stop\_u\_violated\_the\_low.nextInt();  
 **if** (F1.isFile())  
 System.***out***.println(**"является файлом: "** + F1.getName());  
 **if** (folder.isDirectory())  
 System.***out***.println(**"является папкой: "** + folder.getName());  
 **if** (F1.exists())  
 System.***out***.println(**"существует "**);  
 System.***out***.println(F1.getAbsolutePath() + **""**);  
 System.***out***.println(F1.length() + **" в байтах"**);  
 System.***out***.println(F1.isFile() ? **"Это файл"** : **"Ето папка"**);  
  
 *//упражнение 3* stop\_u\_violated\_the\_low.nextInt();  
 File folder3 = **new** File(**"Последняя новая папка"**);  
 folder3.mkdir();  
 File mainFile = **new** File(dir);  
  
 String[] arr = mainFile.list();  
 File[] farr = mainFile.listFiles();  
 **int** count = 0;  
 **for** (File t : farr  
 ) {  
 **if** (t.isDirectory())  
 count++;  
 System.***out***.println(t.getName());  
 }  
  
 **int** n = stop\_u\_violated\_the\_low.nextInt();  
 F1.delete();  
 folderin.delete();  
 folder.delete();  
 folder3.delete();  
 folderin.delete();  
 mainFile.delete();  
 File F2 = **new** File(**"Не совсем новая папка/немного новая папка"**);  
 F2.delete();  
 File F3 = **new** File(**"Не совсем новая папка"**);  
 F3.delete();  
  
 }**catch** (IOException e){  
 System.***out***.println(e.toString());  
 }  
 }  
}

Класс Student задание 2

**package** com.company;  
  
**public class** Student {  
 **private** String **firstName**;  
 **private** String **lastName**;  
 **private int recBookNum**;  
 **private** String **birthDate**;  
 **private** String **phoneNumber**;  
 **private** String **enrollmentDate**;  
 **private** String **groupNumber**;  
 Student(){  
  
 }  
 Student(String fn,String ln,**int** rbn,String bd, String pn, String ed, String gn)**throws** Exception{**try**{  
 **firstName**=fn; **lastName**=ln; **recBookNum**=rbn; **birthDate**=bd; **phoneNumber**=pn; **enrollmentDate**=ed; **groupNumber**=gn;}  
 **catch** (Exception e){  
 System.***out***.println(e.toString());  
 }  
 }  
 **public** String[] checkRecBook(**int** recBookNum) {  
 **if** (**this**.**recBookNum** == recBookNum)  
 **return new** String[] {**firstName**, **lastName**};  
 **return null**;  
 }  
 **public boolean** checkOneDate(Student s){  
 **if** (**this**.**enrollmentDate**.equals(s.**enrollmentDate**))  
 **return true**;  
 **return false**;  
 }  
  
 **public** String getFirstName() {  
 **return firstName**;  
 }  
  
 **public** String getLastName() {  
 **return lastName**;  
 }  
  
 **public** String getBirthDate() {  
 **return birthDate**;  
 }  
  
 **public** String getEnrollmentDate() {  
 **return enrollmentDate**;  
 }  
  
 **public** String getGroupNumber() {  
 **return groupNumber**;  
 }  
  
 **public** String getPhoneNumber() {  
 **return phoneNumber**;  
 }  
  
 **public int** getRecBookNum() {  
 **return recBookNum**;  
 }  
  
 **public void** setFirstName(String firstName) {  
 **this**.**firstName** = firstName;  
 }  
  
 **public void** setLastName(String lastName) {  
 **this**.**lastName** = lastName;  
 }  
  
 **public void** setBirthDate(String birthDate) {  
 **this**.**birthDate** = birthDate;  
 }  
  
 **public void** setEnrollmentDate(String enrollmentDate) {  
 **this**.**enrollmentDate** = enrollmentDate;  
 }  
  
 **public void** setGroupNumber(String groupNumber) {  
 **this**.**groupNumber** = groupNumber;  
 }  
  
 **public void** setRecBookNum(**int** recBookNum) {  
 **this**.**recBookNum** = recBookNum;  
 }  
  
 **public void** setPhoneNumber(String phoneNumber) {  
 **this**.**phoneNumber** = phoneNumber;  
 }  
}

Класс StudStream

**package** com.company;  
  
**import** java.io.\*;  
**import** java.util.ArrayList;  
  
**public class** StudStream {  
 **private** File **file**;  
 **private** DataInputStream **input**;  
 **private** DataOutputStream **output**;  
 **private** RandomAccessFile **randFile**;  
 **public boolean** FileCreate(String fileName) **throws** IOException{  
 **try** {  
 **file**=**new** File(fileName);  
 **if** (!**file**.exists()){  
 **file**.createNewFile();  
 **input**=**new** DataInputStream(**new** FileInputStream(**file**));  
 **output** = **new** DataOutputStream(**new** FileOutputStream(**file**));  
 **return true**;  
 }  
 **else**{  
 **input**=**new** DataInputStream(**new** FileInputStream(**file**));  
 **output** = **new** DataOutputStream(**new** FileOutputStream(**file**));  
 **return false**;  
 }  
 }**catch** (IOException e){  
 System.***out***.println(e.toString());  
 **return false**;  
 }  
 }  
 **public void** OutputInfo(Student student) **throws** IOException{  
 **try** {  
 **output**.writeUTF(student.getFirstName());  
 **output**.writeUTF(student.getLastName());  
 **output**.writeInt(student.getRecBookNum());  
 **output**.writeUTF(student.getBirthDate());  
 **output**.writeUTF(student.getEnrollmentDate());  
 **output**.writeUTF(student.getPhoneNumber());  
 **output**.writeUTF(student.getGroupNumber());  
 }**catch** (IOException e){  
 System.***out***.println(e.toString());  
 }  
 }  
 **public void** InputInfo(Student student) **throws** IOException{  
 **try** {  
 student.setFirstName(**input**.readUTF());  
 student.setLastName(**input**.readUTF());  
 student.setRecBookNum(**input**.readInt());  
 student.setBirthDate(**input**.readUTF());  
 student.setEnrollmentDate(**input**.readUTF());  
 student.setPhoneNumber(**input**.readUTF());  
 student.setGroupNumber(**input**.readUTF());  
 }**catch** (IOException e){  
 System.***out***.println(e.toString());  
 }  
 }  
 **public** ArrayList<Student> FullInput(String group) **throws** IOException{  
 **try**{  
 ArrayList<Student> result=**new** ArrayList<>();  
 Student tempS;  
 **while** ((**input**.available()>0)){  
 tempS = **new** Student();  
 **this**.InputInfo(tempS);  
 **if** (group.equals(tempS.getGroupNumber()))  
 result.add(tempS);  
 }  
 **return** result;  
 }**catch** (IOException e){  
 System.***out***.println(e.toString());  
 **return null**;  
 }  
 }  
 **public** String StringSlitter(Student student){  
 **return ""**;  
 }  
 **public static** String StringCreate(Student s){ *//преобразует в строку* StringBuffer result = **new** StringBuffer();  
 result.append(s.getFirstName()); result.setLength(20);  
 result.append(s.getLastName()); result.setLength(40);  
 result.append(s.getRecBookNum()); result.setLength(60);  
 result.append(s.getBirthDate()); result.setLength(80);  
 result.append(s.getEnrollmentDate()); result.setLength(100);  
 result.append(s.getPhoneNumber()); result.setLength(120);  
 result.append(s.getGroupNumber()); result.setLength(140);  
 result.append(**"\n"**);  
 **return** result.toString();  
 }  
 **public void** Task5(ArrayList<Student>students,File file,**int** num) **throws** IOException {  
 **try** {  
 **long** position = 0, position2 = 40;  
 String tempStr = **""**;  
 **randFile** = **new** RandomAccessFile(file, **"rw"**);  
 **for** (Student s : students *//записываем все в файл* ) {  
 tempStr = *StringCreate*(s);  
 **randFile**.writeBytes(tempStr);  
 position = **randFile**.getFilePointer();  
 }  
 tempStr = **""**;  
 **while** (position2 < position) { *//проходим весь файл* **randFile**.seek(position2);  
 tempStr = **""**;  
 **int** c;  
 **while** (47 < (c = **randFile**.read()) && c < 58)  
 tempStr += (**char**) c; *//получаем весь номер* **if** (Integer.*parseInt*(tempStr) == num) {  
 StringBuffer gaf = **new** StringBuffer();  
 gaf.setLength(140); *//создаем пустую строку на 140 символов* **randFile**.seek(position2 - 40); *//перемещаем курсор в начало записи  
 //randFile.writeBytes(gaf.toString()); эта строка нужна, если мы хотим удалять последнюю запись* **randFile**.seek(position - 141); *//берем последнюю запись* tempStr = **randFile**.readLine(); *//считываем до конца строки* **randFile**.seek(position - 141);  
 **randFile**.writeBytes(gaf.toString()); *//заменяем последнюю запись на пустую* **randFile**.seek(position2 - 40);  
 **randFile**.writeBytes(tempStr); *//заменяем нужную запись* position2 = position; *//создаем условия для выхода из цикла* }  
 position2 += 141;  
 }  
 }**catch** (IOException e){  
 System.***out***.println(e.toString());  
 }  
 }  
}

Класс Main задания 2

**package** com.company;  
**import** java.io.\*;  
**import** java.util.ArrayList;  
**import** java.util.Scanner;  
  
**public class** Main {  
  
 **public static void** main(String[] args)**throws** IOException,Exception {  
 Student a=**new** Student(**"A"**,**"A2"**,15,**"12.12.2000"**,**"8-800-555-35-35"**,**"28.08.2016"**,**"IKBO-08"**);  
 Student b=**new** Student(**"B"**,**"B2"**,16,**"12.12.3123"**,**"8-805-35-35"**,**"28.08.2016"**,**"IKBO-08"**);  
 Student c=**new** Student(**"C"**,**"C2"**,17,**"12.12.2005"**,**"8"**,**"28.08.2015"**,**"IKBO-07"**);  
 Student d=**new** Student(**"D"**,**"D2"**,20,**"12.11.2003"**,**"35"**,**"28.08.2016"**,**"IKBO-08"**);  
 File f = **new** File(**"tempFile.txt"**); f.delete();  
 StudStream t=**new** StudStream();  
 Student temp;  
 ArrayList<Student> tempList= **new** ArrayList<>();  
 System.***out***.println(**"Введите имя файла: "**);  
 Scanner sc = **new** Scanner(System.***in***);  
 **while** (!t.FileCreate(sc.nextLine())){  
 System.***out***.println(**"Такой файл уже существует, введите другое имя: "**);  
 }  
 System.***out***.println(**"Сколько вы хотите записать объектов? "**);  
 **int** count = sc.nextInt(); sc.nextLine();  
 System.***out***.println(**"Последовательно введите через enter: Имя-Фамилию-Номер зачетной книжки-Дату рождения-Номер телефона-Дату зачисления-Группу"**);  
 **for** (**int** i = 0; i<count; i++){  
 System.***out***.println(**"Вводите: "**);  
 String fn = sc.nextLine(); String ln = sc.nextLine(); **int** rbn = sc.nextInt(); sc.nextLine(); String bd = sc.nextLine(); String pn = sc.nextLine();String ed = sc.nextLine(); String gn = sc.nextLine();  
 temp = **new** Student(fn, ln, rbn, bd, pn, ed, gn);  
 t.OutputInfo(temp);  
  
 }  
  
 *//Здесь начинается пункт 4* System.***out***.println(**"Введите номер группы:"**);  
 tempList=t.FullInput(sc.nextLine());  
 System.***out***.println(**"Список студентов этой группы из файла: "**);  
 **for** (Student s: tempList  
 ) {  
 System.***out***.println(s.getFirstName()+**" "**+s.getLastName()+**" "**+s.getRecBookNum()+**" "**+s.getBirthDate()+**" "**+s.getEnrollmentDate()+**" "**+s.getPhoneNumber()+**" "**+s.getGroupNumber());  
 }  
  
 *//Пункт 5* File tempFile=**new** File(**"tempFile.txt"**);  
 tempFile.createNewFile();  
 System.***out***.println(**"Введите номер зачетной книжки студента, которого хотите убрать из файла"**);  
 t.Task5(tempList,tempFile,sc.nextInt());  
  
 }  
  
}

Класс Main задания 3

**package** com.company;  
**import** java.io.\*;  
**import** java.nio.charset.Charset;  
  
**public class** Main {  
 **public static void** main(String[] args)**throws** IOException {  
 **try**{  
 *//Упражнение 1* File f = **new** File(**"T2.txt"**); f.createNewFile();  
 FileReader read = **new** FileReader(**"T1.txt"**);  
 FileWriter write = **new** FileWriter(**"T2.txt"**);  
 **int** x;  
 **while**((x = read.read()) != -1)  
 {  
 write.write(x);  
 }  
 read.close();  
 write.close();  
  
 *//Упражнение 2* f = **new** File(**"A.txt"**);  
 f = **new** File(**"B.txt"**);  
 write = **new** FileWriter(**"A.txt"**);  
 **for** (**int** i = 0; i<512;i++)  
 write.write(**"a"**);  
 write.close();  
 BufferedReader inb = **new** BufferedReader(**new** FileReader(**"A.txt"**),128);  
 BufferedWriter outb = **new** BufferedWriter(**new** FileWriter(**"B.txt"**),128);  
 **char** []buf = **new char**[128];  
 **while**((inb.read(buf))!=-1){  
 outb.write(buf);  
 System.***out***.println(1);}  
 inb.close(); outb.close();  
  
 *//Упражнение 3(если что, второй файл А - с русской буквой)* BufferedReader in = **new** BufferedReader(**new** InputStreamReader(**new** FileInputStream(**"А.txt"**), **"Cp1251"**));  
 System.***out***.println(Charset.*defaultCharset*().name());  
 String str;  
 str=in.readLine();  
 System.***out***.println(str);  
 in = **new** BufferedReader(**new** InputStreamReader(**new** FileInputStream(**"А.txt"**), **"UTF8"**));  
 str=in.readLine();  
 System.***out***.println(str);  
 }**catch** (IOException e){  
 System.***out***.println(e.toString());  
 }  
 }  
}

Класс User задания 4

**package** com.company;  
**import** java.io.\*;  
  
**public class** User **implements** Serializable{  
 **private** String **name**;  
 **private** String **login**;  
 **private** String **password**;  
 User(){  
 **name**=**null**;**login**=**null**;**password**=**null**;  
 }  
 User(String name, String login, String password){  
 **this**.**name**=name; **this**.**login**=login; **this**.**password**=password;  
 }  
 **public boolean** enter(String login, String password){  
 **return** (**this**.**login**==login & **this**.**password**==password);  
 }  
 **public** String getLogin() {  
 **return login**;  
 }  
 **public** String getPassword() {  
 **return password**;  
 }  
 **public** String getPlace(){ **return null**; }  
  
 **public void** setLogin(String login) {  
 **this**.**login** = login;  
 }  
  
 **public void** setName(String name) {  
 **this**.**name** = name;  
 }  
  
 **public void** setPassword(String password) {  
 **this**.**password** = password;  
 }  
}  
  
**class** Client **extends** User{  
 **private** String **place**;  
 Client(String n, String l, String p, String place){  
 **super**(n,l,p);  
 **this**.**place**=place;  
 }  
 @Override  
 **public** String getPlace(){  
 **return place**;  
 }  
 **public void** setPlace(String s){  
 **place**=s;  
 }  
}

класс ClassTextFile задания 4

**package** com.company;  
  
**import** java.io.File;  
**import** java.io.FileNotFoundException;  
**import** java.io.FileReader;  
**import** java.util.Scanner;  
  
**public class** ClassTextFile {  
 **private** String **path**;  
 ClassTextFile(String path){  
 **if** (!**new** File(path).exists())  
 System.***out***.println(**"Этого файла не существует"**);  
 **else  
 this**.**path** = path;  
 }  
 **public** Client InputObject()**throws** FileNotFoundException{  
 Scanner sc = **new** Scanner(**new** File(**path**));  
 String name = sc.nextLine();  
 String login = sc.nextLine();  
 String password = sc.nextLine();  
 String place = sc.nextLine();  
 sc.close();  
 **return new** Client(name,login,password,place);  
 }  
}

класс ClassTask2 задания 4

**package** com.company;  
  
**import** java.util.Scanner;  
  
**public class** ClassTask2 {  
 **private** String **name**;  
 **private** String **login**;  
 **private** String **password**;  
 **private** String **place**;  
  
 ClassTask2(String n, String l, String pw, String pl){  
 **name** = n;  
 **login** = l;  
 **password** = pw;  
 **place** = pl;  
 }  
 **public void** InputInfo(){  
 Scanner sc = **new** Scanner(System.***in***);  
 **name** = sc.nextLine(); sc.nextLine();  
 **login** = sc.nextLine(); sc.nextLine();  
 **password** = sc.nextLine(); sc.nextLine();  
 **place** = sc.nextLine();  
 }  
 **public** Client getClient(){  
 **return new** Client(**name**,**login**,**password**,**place**);  
 }  
 **public void** setClient(Client client){  
 client.setLogin(**login**);  
 client.setName(**name**);  
 client.setPassword(**password**);  
 client.setPlace(**place**);  
 }  
}

Класс ClassSeriyazbleFile задания 4

**package** com.company;  
  
**import** java.io.\*;  
**import** java.util.ArrayList;  
  
**public class** ClassSeriyazableFile {  
 **private** String **path**;  
 **private** ArrayList<Client> **clients**;  
 ClassSeriyazableFile(String p){  
 **clients** = **new** ArrayList<>();  
 **path** = p;  
 }  
 **public** ArrayList<Client> getClients() {  
 **return clients**;  
 }  
 **public void** writeOne(Client client)**throws** IOException {  
 **try** {  
 ObjectOutputStream out = **new** ObjectOutputStream(**new** FileOutputStream(**path**));  
 out.writeObject(client);  
 out.close();  
 }**catch** (IOException e) {  
 System.***out***.println(e.toString());  
 }  
 }  
 **public** Client readOne()**throws** IOException, ClassNotFoundException {  
 **try** {  
 ObjectInputStream in = **new** ObjectInputStream(**new** FileInputStream(path));  
 Client client = (Client) in.readObject();  
 in.close();  
 **return** client;  
 } **catch** (IOException | ClassNotFoundException e) {  
 System.out.println(e.toString());  
 **return null**;  
 }  
 }  
 **public** Object get(){  
 **return this**;  
 }  
 **public void** add(Client client){  
 clients.add(client);  
 }  
 **public void** CollectionSerialization(ArrayList<Client> c)**throws** IOException {  
 **try**{  
 ObjectOutputStream out = **new** ObjectOutputStream(**new** FileOutputStream(path));  
 out.writeObject(c);  
 out.close();  
 }**catch** (IOException e){  
 System.out.println(e.toString());  
 }  
 }  
 **public void** CollectionDeserialization()**throws** IOException,ClassNotFoundException{  
 **try**(ObjectInputStream in = **new** ObjectInputStream(**new** FileInputStream(path))){  
 clients = (ArrayList<Client>) in.readObject();  
 }**catch** (IOException | ClassNotFoundException e){  
 System.out.println(e.toString());  
 }  
 }  
 **public void** CollectionSet(ArrayList<Client>c){  
 clients = c;  
 }  
 **public** ArrayList<Client> CollectionGet(){  
 **return** clients;  
 }  
 **public void** goodOOP1(){  
 **for** (Client client : clients  
 ) {  
 System.out.println(client.getLogin()+**" "**+client.getPassword()+**" "**+client.getPlace());  
 }  
 }  
 **public void** goodOOP2()**throws** IOException{  
 **try**(BufferedReader in = **new** BufferedReader(**new** FileReader(path))){  
 String line;  
 **while** ((line=in.readLine()) != **null**)  
 System.out.println(line);  
 }  
 }  
  
}

Класс Main задания 4

**package** com.company;  
  
**import** java.io.FileNotFoundException;  
**import** java.io.IOException;  
**import** java.util.ArrayList;  
  
**public class** Main {  
  
 **public static void** main(String[] args)**throws** IOException,ClassNotFoundException {  
 *//Первое задание* ClassTextFile ctf = **new** ClassTextFile(**"text.txt"**);  
 Client c = ctf.InputObject();  
 System.***out***.println(c.getPlace() + **" "** + c.getLogin() + **" "** + c.getPassword()+**"\n"**);  
  
 *//Второе задание  
 //Client c= new Client("","","","");* ClassTask2 t2 = **new** ClassTask2(**"i"**,**"f"**,**"g"**,**"h"**);  
 t2.setClient(c);  
 System.***out***.println(c.getPlace() + **" "** + c.getLogin() + **" "** + c.getPassword()+**'\n'**);  
  
 *//третье задание* ClassSeriyazableFile sf = **new** ClassSeriyazableFile(**"Clients.txt"**);  
 Client a = **new** Client(**"1"**,**"2"**,**"3"**,**"4"**);  
 Client b = **new** Client(**"5"**, **"6"**, **"7"**, **"8"**);  
 Client d = **new** Client(**"9"**, **"o"**,**"0"**,**"l"**);  
 ArrayList<Client> testList = **new** ArrayList<>();  
 testList.add(a); testList.add(b); testList.add(d);  
 sf.CollectionSerialization(testList); *//добавили коллекцию файл* sf.CollectionDeserialization(); *//добавили коллекцию из файла в коллекцию в классе* sf.goodOOP1(); *//вывели коллекцию класса* sf.writeOne(c); *//записали "c" в файл* a = sf.readOne(); *//прочитали "c" из файла в "a"* System.***out***.println(a.getPlace() + **" "** + a.getLogin() + **" "** + a.getPassword()+**'\n'**);  
 }  
}