|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | 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: «Тема. Ввод и вывод в Java. Пакет java.io его классы»** | |
|  | |
| Выполнил студент группы ИКБО-17-18 | Мисаилиди А.А. |
| Принял преподаватель | Баранова И.А. |

|  |  |  |
| --- | --- | --- |
| Лабораторная работа выполнена | «\_\_»\_\_\_\_\_\_\_2019 г. | *(подпись студента)* |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_2019 г. | *(подпись руководителя)* |

Москва 2019

# Задание 1. Управление файловой структурой Java. Класс File

### 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.

### 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.

Добавить в папку приложения еще одну папку.

Сформировать массив файлов, находящихся в папке приложения, используя метод list(). Отобразить содержимое массива.

Сформировать массив файлов, находящихся в папке приложения, используя метод listFiles(). Отобразить содержимое массива. Определить количество папок, содержащихся в файле приложения.

Удалить папки и файлы, созданные во всех трех упражнениях.

### Разработка тестов

#### File1 is a file

#### File2 is a file

#### File3 is a file

#### File4 is a directory

#### MyFile1.txt is in the root folder

#### TestDir - папка

#### Файл MyFile1.bin находится в папке приожения

#### Полный путь к файлу MyFile3.bin: C:\Users\IVC5-8\Downloads\Lab4\OOP\_lab04-ishida-master\T1\src\com\sanya\MyFile3.bin

#### C:\Education\University\OOP\Labs\Lab4\MyFile1.txt

#### C:\Education\University\OOP\Labs\Lab4\MyFile2.txt

#### C:\Education\University\OOP\Labs\Lab4\MyFile3.txt

#### C:\Education\University\OOP\Labs\Lab4\MyDir1

#### Size of the first file is 0 byte

#### Size of the second file is 0 byte

#### Size of the third file is 0 byte

#### Size of the folder is 0 byte

Root folder consists of:

.git

.idea

A.txt

B.txt

Lab4.iml

MyDir1

MyDir2

MyFile1.txt

MyFile2.txt

MyFile3.txt

out

Serialization.bin

SerializationCol.bin

src

T1.txt

T2.txt

TourList.txt

### Исходный код программы

### Реализация упражнения 1

File file1 = new File("MyFile1.txt");  
file1.createNewFile();  
File file2 = new File("MyFile2.txt");  
file2.createNewFile();  
File file3 = new File("C:\\Education\\University\\OOP\\Labs\\Lab4\\MyFile3.txt");  
file3.createNewFile();  
File file4 = new File("C:\\Education\\University\\OOP\\Labs\\Lab4\\MyDir1");  
file4.mkdirs();

### Реализация упражнения 2

if (file1.isFile()) {  
 System.*out*.println("File1 is a file");  
} else {  
 System.*out*.println("File1 is not a file");  
}  
if (file2.isFile()) {  
 System.*out*.println("File2 is a file");  
} else {  
 System.*out*.println("File2 is not a file");  
}  
if (file3.isFile()) {  
 System.*out*.println("File3 is a file");  
} else {  
 System.*out*.println("File3 is not a file");  
}  
  
if (file4.isDirectory()) {  
 System.*out*.println("File4 is a directory");  
} else {  
 System.*out*.println("File4 is not a directory");  
}  
System.*out*.println();  
  
if (file1.getPath() == "MyFile1.txt"){  
 System.*out*.println("MyFile1.txt is in the root folder");  
} else {  
 System.*out*.println("MyFile1.txt is not in the root folder");  
}  
System.*out*.println();  
  
System.*out*.println(file1.getAbsolutePath());  
System.*out*.println(file2.getAbsolutePath());  
System.*out*.println(file3.getAbsolutePath());  
System.*out*.println(file4.getAbsolutePath());  
System.*out*.println();  
  
System.*out*.println("Size of the first file is " + file1.length() + " byte");  
System.*out*.println("Size of the second file is " + file2.length() + " byte");  
System.*out*.println("Size of the third file is " + file3.length() + " byte");  
System.*out*.println("Size of the folder is " + file4.length() + " byte");  
System.*out*.println();

### Реализация упражнения 3

File file5 = new File("C:\\Education\\University\\OOP\\Labs\\Lab4\\MyDir2");  
 file5.mkdirs();  
 File root = new File("C:\\Education\\University\\OOP\\Labs\\Lab4");  
  
  
 String[] array1 = root.list();  
 System.*out*.println("Root folder consists of:");  
 for (String f:array1) {  
 System.*out*.println(f);  
 }  
 System.*out*.println();  
  
 File[] array2 = root.listFiles();  
 System.*out*.println("Root folder consists of:");  
 int dirCount = 0;  
 for (File f:array2) {  
 System.*out*.println(f.getName());  
 if (f.isDirectory()) {  
 dirCount++;  
 }  
 } System.*out*.println();  
  
 System.*out*.println("There is " + dirCount + " folders in the root folder");  
  
 file1.delete();  
 file2.delete();  
 file3.delete();  
 file4.delete();  
 file5.delete();  
} catch (FileNotFoundException e) {  
 e.printStackTrace();  
} catch (IOException e) {  
 e.printStackTrace();  
} catch (Exception e) {  
 e.printStackTrace();  
}

# Задание 2. Использование классов байтовых потоков b потоков DataInputStream и DataOutputStream для записи в файл состояния объектов и файла RandomAccesFile

### Условия 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

### Определение класса варианта

Класс Tour имеет 9 полей типов int и String – 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. Методами класса являются:

* По наименованию тура определить и вернуть город и страну пребывания.
* Определить, имеют ли два тура одинаковую стоимость.
* Метод, возвращающий все туры одной туристической компании.
* Метод, увеличивающий стоимость тура на 10% у записи файла, номер которой указал пользователь.

### Класс tour

package com.company;  
  
public class Tour {  
 private String tourName;  
 private String country;  
 private String city;  
 private String hotelName;  
 private int hotelStar;  
 private int days;  
 private int excursionNum;  
 private int tourPrice;  
 private String tourCompany;  
  
 Tour(String tourName, String country, String city, String hotelName, int hotelStar, int days, int excursionNum,  
 int tourPrice, String tourCompany) {  
 this.setTourName(tourName);  
 this.setCountry(country);  
 this.setCity(city);  
 this.setHotelName(hotelName);  
 this.setHotelStar(hotelStar);  
 this.setDays(days);  
 this.setExcursionNum(excursionNum);  
 this.setTourPrice(tourPrice);  
 this.setTourCompany(tourCompany);  
 }  
  
 public int getDays() {  
 return days;  
 }  
  
 public int getExcursionNum() {  
 return excursionNum;  
 }  
  
 public int getHotelStar() {  
 return hotelStar;  
 }  
  
 public int getTourPrice() {  
 return tourPrice;  
 }  
  
 public String getCity() {  
 return city;  
 }  
  
 public String getCountry() {  
 return country;  
 }  
  
 public String getHotelName() {  
 return hotelName;  
 }  
  
 public String getTourCompany() {  
 return tourCompany;  
 }  
  
 public String getTourName() {  
 return tourName;  
 }  
  
 public void setCity(String city) {  
 this.city = city;  
 }  
  
 public void setCountry(String country) {  
 this.country = country;  
 }  
  
 public void setDays(int days) {  
 this.days = days;  
 }  
  
 public void setExcursionNum(int excursionNum) {  
 this.excursionNum = excursionNum;  
 }  
  
 public void setHotelName(String hotelName) {  
 this.hotelName = hotelName;  
 }  
  
 public void setHotelStar(int hotelStar) {  
 this.hotelStar = hotelStar;  
 }  
  
 public void setTourCompany(String tourCompany) {  
 this.tourCompany = tourCompany;  
 }  
  
 public void setTourName(String tourName) {  
 this.tourName = tourName;  
 }  
  
 public void setTourPrice(int tourPrice) {  
 this.tourPrice = tourPrice;  
 }  
}

### Определение класса для управления файлом

Класс FileControl имеет несколько методов, все они статические:

* Метод для добавления тура;
* Методы для вывода списка туров;
* Метод для удаления тура;
* Метод для определения страны и города тура;
* Метод для определения равенства цен двух туров;
* Метод для вывода туров одной тур компании;
* Метод для увеличения цены тура на 10%.

### Класс FileConstructor

package com.company;  
  
import java.io.\*;  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Scanner;  
  
public class FileControl {  
 public static void addTour() throws IOException {  
 File file1 = new File("TourList.txt");  
 DataOutputStream outT = new DataOutputStream(new FileOutputStream(file1.getAbsolutePath(), true));  
 Scanner sc = new Scanner(System.*in*, "cp1251");  
 try {  
 System.*out*.print("Enter tour name: ");  
 outT.writeUTF(sc.nextLine());  
 System.*out*.print("Enter country: ");  
 outT.writeUTF(sc.nextLine());  
 System.*out*.print("Enter city: ");  
 outT.writeUTF(sc.nextLine());  
 System.*out*.print("Enter hotel name: ");  
 outT.writeUTF(sc.nextLine());  
 System.*out*.print("Enter number of hotel stars: ");  
 outT.writeInt(sc.nextInt());  
 System.*out*.print("Enter length of the tour in days: ");  
 outT.writeInt(sc.nextInt());  
 System.*out*.print("Enter number of excursions: ");  
 outT.writeInt(sc.nextInt());  
 System.*out*.print("Enter price of the tour: ");  
 outT.writeInt(sc.nextInt());  
 System.*out*.print("Enter tour company name: ");  
 outT.writeUTF(sc.next());  
  
 outT.flush();  
 outT.close();  
// DataInputStream inT = new DataInputStream(new FileInputStream(file1.getAbsolutePath()));  
// System.out.println(inT.readUTF() + inT.readUTF() + inT.readUTF() + inT.readUTF() + inT.readInt() + inT.readInt() + inT.readInt() + inT.readInt() + inT.readUTF());  
  
 }  
 catch (IOException e) {  
 System.*out*.println(e);  
 }  
 }  
  
 public static void outputTourList() throws IOException {  
 File file1 = new File("TourList.txt");  
 List<Tour> tours = new ArrayList<Tour>();  
 DataInputStream inT = new DataInputStream(new FileInputStream(file1.getAbsolutePath()));  
 try{  
 Tour current;  
 while(true){  
 current = new Tour(inT.readUTF(), inT.readUTF(), inT.readUTF(), inT.readUTF(), inT.readInt(), inT.readInt(), inT.readInt(), inT.readInt(), inT.readUTF());  
 tours.add(current);  
 }  
 }catch(IOException e){  
 }  
 finally {  
 if (tours.size() > 0) {  
 for (int i = 0; i < tours.size(); i++) {  
 System.*out*.println((i + 1) + ". " + tours.get(i).getTourName());  
 }  
 }  
 inT.close();  
 }  
 }  
  
 public static void deleteTour() throws IOException, IndexOutOfBoundsException {  
 File file1 = new File("TourList.txt");  
 List<Tour> tours = new ArrayList<Tour>();  
 Scanner sc = new Scanner(System.*in*, "cp1251");  
 DataInputStream inT = new DataInputStream(new FileInputStream(file1.getAbsolutePath()));  
 System.*out*.print("Enter the number of tour you want to delete: ");  
 int n = sc.nextInt();  
 try{  
 Tour current;  
 while(true){  
 current = new Tour(inT.readUTF(), inT.readUTF(), inT.readUTF(), inT.readUTF(), inT.readInt(), inT.readInt(), inT.readInt(), inT.readInt(), inT.readUTF());  
 tours.add(current);  
 }  
 }catch(IOException e){  
 }  
 finally {  
 DataOutputStream outT = new DataOutputStream(new FileOutputStream(file1.getAbsolutePath()));  
 tours.remove(n - 1);  
 for (int i = 0; i < tours.size(); i++) {  
 outT.writeUTF(tours.get(i).getTourName());  
 outT.writeUTF(tours.get(i).getCountry());  
 outT.writeUTF(tours.get(i).getCity());  
 outT.writeUTF(tours.get(i).getHotelName());  
 outT.writeInt(tours.get(i).getHotelStar());  
 outT.writeInt(tours.get(i).getDays());  
 outT.writeInt(tours.get(i).getExcursionNum());  
 outT.writeInt(tours.get(i).getTourPrice());  
 outT.writeUTF(tours.get(i).getTourCompany());  
 }  
 System.*out*.println("Tour was successfully deleted");  
 outT.flush();  
 outT.close();  
 inT.close();  
 }  
 }  
  
 public static void tourLocation() throws IOException {  
 File file1 = new File("TourList.txt");  
 List<Tour> tours = new ArrayList<Tour>();  
 DataInputStream inT = new DataInputStream(new FileInputStream(file1.getAbsolutePath()));  
 Scanner sc = new Scanner(System.*in*, "cp1251");  
 System.*out*.println("Enter the name of the tour to know the city and the country: ");  
 String target = sc.nextLine();  
 try{  
 Tour current;  
 while(true){  
 current = new Tour(inT.readUTF(), inT.readUTF(), inT.readUTF(), inT.readUTF(), inT.readInt(), inT.readInt(), inT.readInt(), inT.readInt(), inT.readUTF());  
 tours.add(current);  
 }  
 }catch(IOException e){  
 }  
 finally {  
 if (tours.size() > 0) {  
 for (int i = 0; i < tours.size(); i++) {  
 if (tours.get(i).getTourName().equals(target)) {  
 System.*out*.println("City: " + tours.get(i).getCity());  
 System.*out*.println("Country: " + tours.get(i).getCountry());  
 }  
 }  
 }  
 inT.close();  
 }  
 }  
  
 public static void priceEq() throws IOException {  
 File file1 = new File("TourList.txt");  
 List<Tour> tours = new ArrayList<Tour>();  
 DataInputStream inT = new DataInputStream(new FileInputStream(file1.getAbsolutePath()));  
 Scanner sc = new Scanner(System.*in*, "cp1251");  
 System.*out*.print("Enter the number of the first tour which price you want to compare: ");  
 int target1 = sc.nextInt();  
 System.*out*.print("Enter the number of the second tour which price you want to compare: ");  
 int target2 = sc.nextInt();  
 try{  
 Tour current;  
 while(true){  
 current = new Tour(inT.readUTF(), inT.readUTF(), inT.readUTF(), inT.readUTF(), inT.readInt(), inT.readInt(), inT.readInt(), inT.readInt(), inT.readUTF());  
 tours.add(current);  
 }  
 }catch(IOException e){  
 }  
 finally {  
 if (target1 > 0 && target2 > 0 && target1 <= tours.size() && target2 <= tours.size()) {  
 if (tours.get(target1 - 1).getTourPrice() == tours.get(target2 - 1).getTourPrice()) {  
 System.*out*.println("The prices are equal");  
 } else {  
 System.*out*.println("Prices are not equal");  
 }  
 } else {  
 System.*out*.println("Invalid number");  
 }  
 }  
 }  
  
 public static void tourCompanyList() throws IOException {  
 File file1 = new File("TourList.txt");  
 List<Tour> tours = new ArrayList<Tour>();  
 DataInputStream inT = new DataInputStream(new FileInputStream(file1.getAbsolutePath()));  
 Scanner sc = new Scanner(System.*in*, "cp1251");  
 System.*out*.print("Enter tour company name: ");  
 String target = sc.nextLine();  
 try{  
 Tour current;  
 while(true){  
 current = new Tour(inT.readUTF(), inT.readUTF(), inT.readUTF(), inT.readUTF(), inT.readInt(), inT.readInt(), inT.readInt(), inT.readInt(), inT.readUTF());  
 tours.add(current);  
 }  
 }catch(IOException e){  
 }  
 finally {  
 List<Tour> company = new ArrayList<Tour>();  
 for (int i = 0; i < tours.size(); i++) {  
 if (tours.get(i).getTourCompany().equals(target)) {  
 company.add(tours.get(i));  
 }  
 }  
 for (int i = 0; i < company.size(); i++) {  
 System.*out*.println((i + 1) + ". " + company.get(i).getTourName());  
 }  
 inT.close();  
 }  
 }  
  
 public static void priceIncrease() throws IOException, IndexOutOfBoundsException {  
 File file1 = new File("TourList.txt");  
 List<Tour> tours = new ArrayList<Tour>();  
 Scanner sc = new Scanner(System.*in*, "cp1251");  
 DataInputStream inT = new DataInputStream(new FileInputStream(file1.getAbsolutePath()));  
 System.*out*.print("Enter the number of tour which price you want to increase by 10%: ");  
 int n = sc.nextInt();  
 try{  
 Tour current;  
 while(true){  
 current = new Tour(inT.readUTF(), inT.readUTF(), inT.readUTF(), inT.readUTF(), inT.readInt(), inT.readInt(), inT.readInt(), inT.readInt(), inT.readUTF());  
 tours.add(current);  
 }  
 }catch(IOException e){  
 }  
 finally {  
 DataOutputStream outT = new DataOutputStream(new FileOutputStream(file1.getAbsolutePath()));  
 for (int i = 0; i < tours.size(); i++) {  
 outT.writeUTF(tours.get(i).getTourName());  
 outT.writeUTF(tours.get(i).getCountry());  
 outT.writeUTF(tours.get(i).getCity());  
 outT.writeUTF(tours.get(i).getHotelName());  
 outT.writeInt(tours.get(i).getHotelStar());  
 outT.writeInt(tours.get(i).getDays());  
 outT.writeInt(tours.get(i).getExcursionNum());  
 if (i == n - 1) {  
 outT.writeInt((int)(tours.get(i).getTourPrice() \* 1.1));  
 }  
 else{  
 outT.writeInt(tours.get(i).getTourPrice());  
 }  
 outT.writeUTF(tours.get(i).getTourCompany());  
 }  
 outT.flush();  
 outT.close();  
 System.*out*.println("Tour was successfully deleted");  
 inT.close();  
 }  
 }  
  
  
}

### Разработка тестов

#### Choose variant:

#### 1. Add a tour to the list

#### 2. Show the list of the tours

#### 3. Delete tour from the list

#### 4. Get city and country of the tour

#### 5. Check equality of tours prices

#### 6. Get tour list of one of the companies

#### 7. Increase tour cost by 10%

#### 0. End program

#### Choose variant: 1

#### Enter tour name: r

#### Enter country: r

#### Enter city: r

#### Enter hotel name: r

#### Enter number of hotel stars: 10

#### Enter length of the tour in days: 10

#### Enter number of excursions: 10

#### Enter price of the tour: 10

#### Enter tour company name: q

#### Choose variant:

#### 1. Add a tour to the list

#### 2. Show the list of the tours

#### 3. Delete tour from the list

#### 4. Get city and country of the tour

#### 5. Check equality of tours prices

#### 6. Get tour list of one of the companies

#### 7. Increase tour cost by 10%

#### 0. End program

#### Choose variant: 2

#### 1. q

#### 2. w

#### 3. e

#### 4. r

#### Choose variant:

#### 1. Add a tour to the list

#### 2. Show the list of the tours

#### 3. Delete tour from the list

#### 4. Get city and country of the tour

#### 5. Check equality of tours prices

#### 6. Get tour list of one of the companies

#### 7. Increase tour cost by 10%

#### 0. End program

#### Choose variant: 4

#### Enter the name of the tour to know the city and the country:

#### q

#### City: q

#### Country: q

#### Choose variant:

#### 1. Add a tour to the list

#### 2. Show the list of the tours

#### 3. Delete tour from the list

#### 4. Get city and country of the tour

#### 5. Check equality of tours prices

#### 6. Get tour list of one of the companies

#### 7. Increase tour cost by 10%

#### 0. End program

#### Choose variant: 5

#### Enter the number of the first tour which price you want to compare: 1

#### Enter the number of the second tour which price you want to compare: 2

#### Prices are not equal

#### Choose variant:

#### 1. Add a tour to the list

#### 2. Show the list of the tours

#### 3. Delete tour from the list

#### 4. Get city and country of the tour

#### 5. Check equality of tours prices

#### 6. Get tour list of one of the companies

#### 7. Increase tour cost by 10%

#### 0. End program

#### Choose variant: 6

#### Enter tour company name: q

#### 1. q

#### 2. e

#### 3. r

#### Choose variant:

#### 1. Add a tour to the list

#### 2. Show the list of the tours

#### 3. Delete tour from the list

#### 4. Get city and country of the tour

#### 5. Check equality of tours prices

#### 6. Get tour list of one of the companies

#### 7. Increase tour cost by 10%

#### 0. End program

#### Choose variant: 2

#### 1. q

#### 2. w

#### 3. e

#### 4. r

#### Choose variant:

#### 1. Add a tour to the list

#### 2. Show the list of the tours

#### 3. Delete tour from the list

#### 4. Get city and country of the tour

#### 5. Check equality of tours prices

#### 6. Get tour list of one of the companies

#### 7. Increase tour cost by 10%

#### 0. End program

#### Choose variant: 7

#### Enter the number of tour which price you want to increase by 10%: 4

#### Tour was successfully deleted

#### Choose variant:

#### 1. Add a tour to the list

#### 2. Show the list of the tours

#### 3. Delete tour from the list

#### 4. Get city and country of the tour

#### 5. Check equality of tours prices

#### 6. Get tour list of one of the companies

#### 7. Increase tour cost by 10%

#### 0. End program

#### Choose variant: 2

#### 1. q

#### 2. w

#### 3. e

#### 4. r

#### Choose variant:

#### 1. Add a tour to the list

#### 2. Show the list of the tours

#### 3. Delete tour from the list

#### 4. Get city and country of the tour

#### 5. Check equality of tours prices

#### 6. Get tour list of one of the companies

#### 7. Increase tour cost by 10%

#### 0. End program

#### Choose variant: 0

#### Ending the program...

#### Process finished with exit code 0

### Исходный код программы

### Реализация приложения

int comm0 = -1;  
Scanner sc = new Scanner(System.*in*);  
while (comm0 != 0) {  
 *outputMenu*();  
 System.*out*.print("Choose variant: ");  
 comm0 = sc.nextInt();  
 System.*out*.println();  
 switch (comm0) {  
 case 1:  
 FileControl.*addTour*();  
 break;  
 case 2:  
 FileControl.*outputTourList*();  
 break;  
 case 3:  
 FileControl.*deleteTour*();  
 break;  
 case 4:  
 FileControl.*tourLocation*();  
 break;  
 case 5:  
 FileControl.*priceEq*();  
 break;  
 case 6:  
 FileControl.*tourCompanyList*();  
 break;  
 case 7:  
 FileControl.*priceIncrease*();  
 break;  
 case 0:  
 System.*out*.println("Ending the program...");  
 break;  
 default:  
 System.*out*.println("Invalid input");  
 }  
}

# Задание 3. Исследование возможностей символьных потоков ввода и вывода

### Условие задачи

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

### 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 src 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 поток in для 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.

Определите новый поток для чтения, установив кодировку «UTF-8».

Вновь выполните программу, удостоверьтесь, что текст был выведен корректно.

### Тестирование приложения

Data has been written from T1.txt to T2.txt successfully

Data has been written from A.txt to B.txt successfully

Предложение на русском языке

Process finished with exit code 0

### Исходный код программы

### Реализация упражнения 1

public static void first\_step() throws IOException {  
 File file1 = new File("T1.txt");  
 File file2 = new File("T2.txt");  
 Reader R = new InputStreamReader(new FileInputStream(file1.getAbsolutePath()));  
 Writer W = new OutputStreamWriter(new FileOutputStream(file2.getAbsolutePath(), true));  
 int x;  
 try {  
 while((x = R.read()) != -1){  
 W.write((char)x);  
 }  
 } catch(IOException e) {  
 } finally {  
 System.*out*.println("Data has been written from T1.txt to T2.txt successfully");  
 }  
}

### Реализация упражнения 2

public static void second\_step() throws IOException {  
 File file1 = new File("A.txt");  
 file1.createNewFile();  
 Writer W = new OutputStreamWriter(new FileOutputStream(file1.getAbsolutePath(), true));  
 BufferedReader inb = new BufferedReader(new FileReader(file1.getAbsolutePath()), 128);  
 for (int i = 0; i < 512; i++) {  
 W.write("k");  
 }  
 File file2 = new File("B.txt");  
 file2.createNewFile();  
 BufferedWriter outb = new BufferedWriter(new FileWriter(file2.getAbsolutePath()), 128);  
 char[] buf = new char[128];  
 for (int i = 0; i < 4; i++) {  
 inb.read(buf);  
 outb.write(buf);  
 if (i != 3) {  
 outb.write("\n");  
 }  
 }  
 inb.close();  
 outb.close();  
 System.*out*.println("Data has been written from A.txt to B.txt successfully");  
}

### Реализация упражнения 3

private static void third\_step() throws IOException {  
 File file1 = new File("A.txt");  
 file1.createNewFile();  
 BufferedReader in = new BufferedReader(new InputStreamReader(new FileInputStream(file1.getAbsolutePath()),"Cp1251"));  
 System.*out*.println(Charset.*defaultCharset*().name());  
 String line = in.readLine();  
 in.close();  
  
 in = new BufferedReader(new InputStreamReader(new FileInputStream(file1.getAbsolutePath()), StandardCharsets.*UTF\_8*));  
 line = in.readLine();  
 in.close();  
 System.*out*.println(line);  
}

### Реализация приложения

// Упражнение 1  
 *first\_step*();  
  
// Упражнение 2  
 *second\_step*();  
  
// Упражнение 3  
 *third\_step*();

# Задание 4. Применение классов ObjectOutputStream и ObjectInputStream для сериализации и десериализации объектов

### 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.

### Постановка задачи

1. В задании лабораторной работы 2 выберите свой вариант. 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:
     1. Конструктор с параметром – путь к файлу.
     2. 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.

Занесение данных в объект.

Метод, возвращающий объект, вызвавший метод.

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 сериализованых 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.

# Диаграмма классов

Рис. 1 UML-диаграмма классов

### Разработка тестов

#### File already exists!

#### Info about tour: Name country city hotelName 5 7 3 20000 tourCompany

#### Enter tour name: q

#### Enter tour city: q

#### Enter country of the tour: q

#### Enter the length of the tour: 1

#### Enter excursions number: 1

#### Enter hotel name: q

#### Enter number of hotel stars: 1

#### Enter tour price: 1

#### Enter tour company name: q

#### Info about a test: q q q q 1 1 1 1 q

#### How many tours you want to add?

#### 1

#### Enter tour name: w

#### Enter country of the tour: w

#### Enter tour city: w

#### Enter hotel name: w

#### Enter number of hotel stars: 2

#### Enter the length of the tour: 2

#### Enter excursions number: 2

#### Enter tour price: 2

#### Enter tour company name: w

#### Collection looks like this after addition of objects:

#### w w w w 2 2 2 2 w

#### Collection looks like this after write/read:

#### w w w w 2 2 2 2 w

#### Process finished with exit code 0

### Исходный код программы

### Класс Object

package com.company;  
  
import java.util.Scanner;  
  
public class Object {  
 private String tourName;  
 private String country;  
 private String city;  
 private String hotelName;  
 private int hotelStar;  
 private int days;  
 private int excursionNum;  
 private int tourPrice;  
 private String tourCompany;  
 private Scanner sc = new Scanner(System.*in*);  
  
 Object() {  
 }  
  
 Object(String tourName, String country, String city, String hotelName, int hotelStar, int days, int excursionNum, int tourPrice, String tourCompany) {  
 this.tourName = tourName;  
 this.country = country;  
 this.city = city;  
 this.hotelName = hotelName;  
 this.hotelStar = hotelStar;  
 this.days = days;  
 this.excursionNum = excursionNum;  
 this.tourPrice = tourPrice;  
 this.tourCompany = tourCompany;  
 }  
  
 public void setTourPrice() {  
 System.*out*.print("Enter tour price: ");  
 this.tourPrice = sc.nextInt();  
 sc.nextLine();  
 }  
  
 public void setTourName() {  
 System.*out*.print("Enter tour name: ");  
 this.tourName = sc.nextLine();  
 }  
  
 public void setTourCompany() {  
 System.*out*.print("Enter tour company name: ");  
 this.tourCompany = sc.nextLine();  
 }  
  
 public void setHotelStar() {  
 System.*out*.print("Enter number of hotel stars: ");  
 this.hotelStar = sc.nextInt();  
 }  
  
 public void setHotelName() {  
 System.*out*.print("Enter hotel name: ");  
 this.hotelName = sc.nextLine();  
 }  
  
 public void setExcursionNum() {  
 System.*out*.print("Enter excursions number: ");  
 this.excursionNum = sc.nextInt();  
 sc.nextLine();  
 }  
  
 public void setDays() {  
 System.*out*.print("Enter the length of the tour: ");  
 this.days = sc.nextInt();  
 }  
  
 public void setCountry() {  
 System.*out*.print("Enter country of the tour: ");  
 this.country = sc.nextLine();  
 }  
  
 public void setCity() {  
 System.*out*.print("Enter tour city: ");  
 this.city = sc.nextLine();  
 }  
  
 public String getTourName() {  
 return tourName;  
 }  
  
 public String getTourCompany() {  
 return tourCompany;  
 }  
  
 public String getHotelName() {  
 return hotelName;  
 }  
  
 public String getCountry() {  
 return country;  
 }  
  
 public String getCity() {  
 return city;  
 }  
  
 public int getTourPrice() {  
 return tourPrice;  
 }  
  
 public int getHotelStar() {  
 return hotelStar;  
 }  
  
 public int getExcursionNum() {  
 return excursionNum;  
 }  
  
 public int getDays() {  
 return days;  
 }  
  
}

### Класс Object serialization

package com.company;  
  
import java.io.\*;  
import java.util.ArrayList;  
import java.util.Scanner;  
  
public class ObjectSerialization {  
 String file1 = "C:\\Education\\University\\OOP\\Labs\\Lab4\\Serialization.bin";  
 String file2 = "C:\\Education\\University\\OOP\\Labs\\Lab4\\SerializationCol.bin";  
 private ArrayList<Object> tours = new ArrayList<Object>();  
  
 ObjectSerialization(String file1) throws IOException {  
 try {  
 if (!new File(file1).createNewFile()) {  
 System.*out*.println("File already exists!");  
 } else {  
 this.file1 = file1;  
 }  
 } catch(IOException e) {}  
 }  
  
 public void outputTour() throws IOException {  
 Object output = new Object();  
 output.setTourName();  
 output.setCity();  
 output.setCountry();  
 output.setDays();  
 output.setExcursionNum();  
 output.setHotelName();  
 output.setHotelStar();  
 output.setTourPrice();  
 output.setTourCompany();  
  
 ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream(file1));  
 out.writeUTF(output.getTourName());  
 out.writeUTF(output.getCountry());  
 out.writeUTF(output.getCity());  
 out.writeUTF(output.getHotelName());  
 out.writeInt(output.getHotelStar());  
 out.writeInt(output.getDays());  
 out.writeInt(output.getExcursionNum());  
 out.writeInt(output.getTourPrice());  
 out.writeUTF(output.getTourCompany());  
  
 out.close();  
 }  
  
 void inputTour () throws IOException {  
 ObjectInputStream in = new ObjectInputStream(new FileInputStream(file1));  
  
 Object input = new Object(in.readUTF(), in.readUTF(), in.readUTF(), in.readUTF(), in.readInt(), in.readInt(), in.readInt(), in.readInt(), in.readUTF());  
  
 System.*out*.println("Info about a test: "+input.getTourName()+" "+input.getCountry()+" "+input.getCity()  
 +" "+input.getHotelName()+" "+input.getHotelStar()+" "+input.getDays()+" "+input.getExcursionNum()  
 +" "+input.getTourPrice()+" "+input.getTourCompany());  
 in.close();  
 }  
  
 public void addToCollection() {  
 Scanner in = new Scanner(System.*in*);  
 System.*out*.println("How many tours you want to add?");  
 int num = in.nextInt();  
 for (int i = 0; i < num; i++) {  
 Object t = new Object();  
 t.setTourName();  
 t.setCountry();  
 t.setCity();  
 t.setHotelName();  
 t.setHotelStar();  
 t.setDays();  
 t.setExcursionNum();  
 t.setTourPrice();  
 t.setTourCompany();  
 tours.add(t);  
 }  
 }  
  
 public void outputCollection() throws IOException {  
 ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream(file2));  
 for (Object t: tours) {  
 out.writeUTF(t.getTourName());  
 out.writeUTF(t.getCountry());  
 out.writeUTF(t.getCity());  
 out.writeUTF(t.getHotelName());  
 out.writeInt(t.getHotelStar());  
 out.writeInt(t.getDays());  
 out.writeInt(t.getExcursionNum());  
 out.writeInt(t.getTourPrice());  
 out.writeUTF(t.getTourCompany());  
 }  
  
 out.close();  
 }  
  
 void inputCollection() throws IOException {  
 ObjectInputStream in = new ObjectInputStream(new FileInputStream(file2));  
  
 tours.clear();  
 while(true) {  
 try {  
 tours.add(new Object(in.readUTF(), in.readUTF(), in.readUTF(), in.readUTF(), in.readInt(), in.readInt(), in.readInt(), in.readInt(), in.readUTF()));  
 } catch (EOFException e) {  
 in.close();  
 break;  
 }  
 }  
 }  
  
 void printCollection() {  
 for (Object t : tours) {  
 System.*out*.println(t.getTourName()+" "+t.getCountry()+" "+t.getCity()  
 +" "+t.getHotelName()+" "+t.getHotelStar()+" "+t.getDays()+" "+t.getExcursionNum()  
 +" "+t.getTourPrice()+" "+t.getTourCompany());  
 }  
 }  
  
 ObjectSerialization getObject() {  
 return this;  
 }  
  
 ArrayList<Object> getCollection() {  
 return tours;  
 }  
}

### Класс ClassTextFile

package com.company;  
  
import java.io.File;  
import java.io.FileNotFoundException;  
import java.io.FileReader;  
import java.io.IOException;  
import java.util.Scanner;  
  
public class ClassTextFile {  
 private String file1 = "Input.txt";  
 ClassTextFile (String file1) {  
 try {  
 if (!new File(file1).createNewFile())  
 System.*out*.println("File already exists!");  
 else  
 this.file1 = file1;  
 } catch (IOException e) {  
 }  
 }  
  
 Object setTourInfo() throws FileNotFoundException {  
 Scanner inT = new Scanner(new FileReader(file1));  
 String tourName = inT.nextLine();  
 String country = inT.nextLine();  
 String city = inT.nextLine();  
 String hotelName = inT.nextLine();  
 int hotelStar = inT.nextInt();  
 int days = inT.nextInt();  
 int excursionNum = inT.nextInt();  
 int tourPrice = inT.nextInt();  
 inT.nextLine();  
 String tourCompany = inT.nextLine();  
 return new Object(tourName, country, city, hotelName, hotelStar, days, excursionNum, tourPrice, tourCompany);  
 }  
  
}

### Реализация приложения

try {  
 String file1path = "C:\\Education\\University\\OOP\\Labs\\Lab4\\Serialization.bin";  
 File fileS = new File(file1path);  
 fileS.createNewFile();  
 Scanner fileIn = new Scanner(new FileReader(fileS));  
 ClassTextFile input = new ClassTextFile("Input.txt");  
 Object one = input.setTourInfo();  
 System.*out*.println("Info about tour: "+one.getTourName()+" "+one.getCountry()+" "+one.getCity()  
 +" "+one.getHotelName()+" "+one.getHotelStar()+" "+one.getDays()+" "+one.getExcursionNum()  
 +" "+one.getTourPrice()+" "+one.getTourCompany());  
 ObjectSerialization second = new ObjectSerialization("C:\\Education\\University\\OOP\\Labs\\Lab4\\Serialization.bin");  
 second.outputTour();  
 second.inputTour();  
 ObjectSerialization third = new ObjectSerialization("C:\\Education\\University\\OOP\\Labs\\Lab4\\SerializationCol.bin");  
 third.addToCollection();  
 System.*out*.println("Collection looks like this after addition of objects:");  
 third.printCollection();  
 third.outputCollection();  
 third.inputCollection();  
 System.*out*.println("Collection looks like this after write/read:");  
 third.printCollection();  
} catch (IOException e) {  
}

# Работа с GitHub

Готовый проект загружен и размещен на GitHub, доступен по ссылке:

<https://github.com/rtu-mirea/OOP_lab04-Ish1da>

Рис. 2 Файлы загружены и находятся в репозитории