Министерство образования и науки РФ

Институт информационных технологий и управления в технических системах

Севастопольский государственный университет

Лабораторная работа №3

по дисциплине: «Платформа Java»

Выполнил:

ст.гр. ИСб/22о

Воронин И.Ю.

Проверил:

Кузнецов С.А.

г. Севастополь

2016 г.

1. Цель работы

В ходе выполнения данной лабораторной работы необходимо ознакомиться с организацией коллекций объектов на языке Java, приобрести практические навыки использования списков, очередей, хеш-таблиц при создании Java программ.

1. Вариант задания

2.1. В соответствии с вариантом задания (см. таблицу 2.1) реализовать класс для представления требуемой информации.

2.2. Реализовать коллекцию типа Т1(см. таблицу 2.1) объектов разработанного в п. 2.1. класса с возможностью ввода элементов из файла, вывода на консоль, проверки членства по введенному с консоли значению поля 1. Имя файла вводить параметром командной строки –i.

2.3. Реализовать коллекцию типа LinkedList объектов разработанного в п. 2.1. с возможностью: упорядочивания по полю 1 (использовать Collections.sort(list)); с возможностью упорядочивания по полю P(см. таблицу 2.1) в направлении U класса (использовать Collections.sort(list, myComp), где myComp – экземпляр разработанного класса, реализующего интерфейс Comparator); с возможностью ввода элементов из файла, вывода на консоль и сохранения в файл. Имена файлов вводить параметрами командной строки –i и –o.

2.4. Реализовать коллекцию типа Т2(см. таблицу 2.1) объектов разработанного в п. 2.1. класса c ключом по значению поля 1, с возможностью ввода элементов из файла, вывода на консоль в виде «Ключ -> Значения» (значения остальных полей), вывода значения полей по введенному с консоли значению поля 1. Имя файла вводить параметром командной строки –i.

2.5. Реализовать класс Lab3Java, в методе main которого реализовать работу с объектами классов из п. 2.1-2.4:

1. Ввести записи из файла заданного параметром командной строки –i в коллекцию T1.

2. Отобразить записи в консоли.

3. Предложить пользователю ввести значение поля 1.

4. Отобразить в консоли результат проверки наличия записи по введенному значению поля 1.

5. Ввести записи из файла заданного параметром командной строки –i в коллекцию LinkedList.

6. Отобразить записи в консоли. Отсортировать по полю 1. Отобразить записи в консоли. Отсортировать по полю P в направлении U. Отобразить записи в консоли.7. Вывести записи в файл, заданный параметром командной строки –o.

8. Ввести записи из файла заданного параметром командной строки –i в коллекцию T2.

9. Отобразить записи в консоли.

10. Предложить пользователю ввести значение поля 1.

11. Отобразить в консоли значения остальных полей по введенному значению поля 1.

Таблица 2.1 Варианты заданий

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Тип информации | Поле для сортировки (P) | Направление (U) | Тип коллекции (Т1) | Тип коллекции (Т2) |
| 2 | B | 2 | Возрастание | TreeSet | TreeMap |

Тип информации:

B: Автомобиль (Марка, Год выпуска, Объем двигателя, Максимальная скорость);

1. Код программы

Класс Car:

public class Car implements Comparable<Car>, SortedMap<String,Car> {

//Марка

private String mark;

//Год выпуска

private int year\_made;

//Объем двигателя

private double engine;

//Максимальная скорость

private double max\_speed;

public String getMark(){

return mark;

}

public int getYear\_made(){

return year\_made;

}

//чтение из файла

public TreeSet<Car> ReadFile(String FileName) throws IOException{

Scanner fin = new Scanner(new File(FileName));

TreeSet <Car> TSCar = new TreeSet<Car>();

String mark;

int year\_made;

double max\_speed,engine;

while(fin.hasNext())

{

if(TSCar.size()>0){

fin.nextLine();

}

mark = fin.nextLine();

year\_made = Integer.valueOf(fin.nextLine());

engine = Double.valueOf(fin.nextLine());

max\_speed = Double.valueOf(fin.nextLine());

TSCar.add(new Car(mark,year\_made,engine,max\_speed));

}

fin.close();

return TSCar;

}

//данные для сравнения

public String ToStringComp(){

return mark + year\_made + engine + max\_speed;

}

//вывод данных об объекте

public void getInfo(){

System.out.println("Mark: " + mark);

System.out.println("Year was made: " + year\_made);

System.out.println("Engine: " + engine);

System.out.println("Max speed: " + max\_speed + "\n");

}

//есть ли данные такого типа?

public boolean FindMatch(String mark){

if(mark.compareTo(this.mark)==0){

return true;

}

else return false;

}

Car(String mark, int year\_made, double engine, double max\_speed){

this.mark = mark;

this.year\_made = year\_made;

this.engine = engine;

this.max\_speed = max\_speed;

}

Car(){

mark = "";

engine = max\_speed = year\_made = 0;

}

public int compareTo(Car car){

return this.ToStringComp().compareTo(car.ToStringComp());

}

public String toString(){

return "" + mark + " " + year\_made + " " + engine + " " + max\_speed + "\r\n";

}

@Override

public int size() {

// TODO Auto-generated method stub

return 0;

}

@Override

public boolean isEmpty() {

// TODO Auto-generated method stub

return false;

}

@Override

public boolean containsKey(Object key) {

// TODO Auto-generated method stub

return false;

}

@Override

public boolean containsValue(Object value) {

// TODO Auto-generated method stub

return false;

}

@Override

public Car get(Object key) {

// TODO Auto-generated method stub

return null;

}

@Override

public Car put(String key, Car value) {

// TODO Auto-generated method stub

return null;

}

@Override

public Car remove(Object key) {

// TODO Auto-generated method stub

return null;

}

@Override

public void putAll(Map<? extends String, ? extends Car> m) {

// TODO Auto-generated method stub

}

@Override

public void clear() {

// TODO Auto-generated method stub

}

@Override

public Comparator<? super String> comparator() {

// TODO Auto-generated method stub

return null;

}

@Override

public SortedMap<String, Car> subMap(String fromKey, String toKey) {

// TODO Auto-generated method stub

return null;

}

@Override

public SortedMap<String, Car> headMap(String toKey) {

// TODO Auto-generated method stub

return null;

}

@Override

public SortedMap<String, Car> tailMap(String fromKey) {

// TODO Auto-generated method stub

return null;

}

@Override

public String firstKey() {

// TODO Auto-generated method stub

return null;

}

@Override

public String lastKey() {

// TODO Auto-generated method stub

return null;

}

@Override

public Set<String> keySet() {

// TODO Auto-generated method stub

return null;

}

@Override

public Collection<Car> values() {

// TODO Auto-generated method stub

return null;

}

@Override

public Set<java.util.Map.Entry<String, Car>> entrySet() {

// TODO Auto-generated method stub

return null;

}

}

Класс Main:

package javaLab3;

import java.io.BufferedReader;

import java.io.FileWriter;

import java.io.IOException;

import java.security.KeyStore.Entry;

import java.util.\*;

import Sort.SortFirstField;

import Sort.SortSecondField;

public class Main {

public static void main(String[] args) throws IOException {

//работа с файлами

Scanner scanner = new Scanner(System.in);

BufferedReader fin = null;

FileWriter fout = null;

String IFileName = new String();

String OFileName = new String();

if(args.length>=2){

if(args[0].compareTo("-i") == 0) IFileName = args[1];

if(args[0].compareTo("-o") == 0) OFileName = args[1];

}

if(args.length>=4){

if(args[2].compareTo("-o") == 0) OFileName = args[3];

if(args[2].compareTo("-i") == 0) IFileName = args[3];

}

//меню

while(true){

System.out.println("1 - TreeSet");

System.out.println("2 - LinkedList");

System.out.println("3 - TreeMap");

System.out.println("0 - Exit");

switch(scanner.nextInt()){

case 1:{

TreeSet <Car> TSCar = new TreeSet<Car>();

String str = new String();

//чтение из файла

System.out.println("Загрузить данные с файла?");

String temp = scanner.next();

int c = (int)temp.toCharArray()[0];

if(c == (int)'Y' | c == (int)'y'){

TSCar.addAll(new Car().ReadFile(IFileName));

}

TSCar.add(new Car("BMW", 1990, 3, 170));

TSCar.add(new Car("Ferari", 2010, 4, 170));

TSCar.add(new Car("Porsche", 1992, 6, 200));

TSCar.add(new Car("Jugylini", 1980, 7, 300));

//поиск совпадений по 1 полю

System.out.print("Enter string to find matches: ");

int flag = 0;

str = scanner.next();

for(Car car: TSCar){

if(car.FindMatch(str) == true){

System.out.println("Matche was found!");

flag = 1;

break;

}

}

if(flag == 0) System.out.println("Matche was not found!");

//печать всех значений

for(Car car: TSCar){

car.getInfo();

}

}break;

///////////////////////////////2.3/////////////////////////////

case 2:{

LinkedList <Car> LLCar = new LinkedList<Car>();

//загрузка с файла

System.out.println("Загрузить данные с файла?");

String temp = scanner.next();

int c = (int)temp.toCharArray()[0];

if(c == (int)'Y' | c == (int)'y'){

LLCar.addAll(new Car().ReadFile(IFileName));

}

LLCar.add(new Car("Ferari", 2010, 4, 170));

LLCar.add(new Car("Porsche", 1992, 6, 200));

LLCar.add(new Car("BMW", 1990, 3, 170));

LLCar.add(new Car("Jugylini", 1980, 7, 300));

System.out.println("Выберите номер поля, по которому будет произведена сортировка(1,2): ");

int key = scanner.nextInt();

switch(key){

case 1: Collections.sort(LLCar, new SortFirstField());break;

case 2: Collections.sort(LLCar, new SortSecondField());break;

}

for(Car C: LLCar){

C.getInfo();

}

System.out.println("Сохранить данные в файл?");

temp = scanner.next();

c = (int)temp.toCharArray()[0];

if (c == (int)'Y' | c == (int)'y'){

FileSaveCollection.LinkerListSave(LLCar, OFileName);

}

}break;

///////////////////////////////////2.4/////////////////////////////////////////

case 3:{

TreeMap<String, Car> TMCar = new TreeMap<String, Car>();

//загрузка из файла

TMCar.putAll(Adapter.TreeToMap(IFileName));

//поиск элемента по ключу

System.out.println("Введите ключ элемента: ");

String str = scanner.next();

if(TMCar.get(str) != null){

System.out.println("Совпадение было найдено. ");

TMCar.get(str).getInfo();

}

for(java.util.Map.Entry<String, Car> entry: TMCar.entrySet()) {

String K = entry.getKey();

Car value = entry.getValue();

System.out.println(K + " ---> " + value.getYear\_made() );

}

break;

}

case 0: return;

}

}

}

}

Класс FileSaveCollection:

public class FileSaveCollection {

public static void LinkerListSave(LinkedList<Car> LL, String IFilename) throws IOException{

FileWriter f = new FileWriter(IFilename);

for(Car c : LL){

f.write(c.toString());

}

f.close();

}

}

Класс Adapter:

public abstract class Adapter {

public static TreeMap <String, Car> TreeToMap(String IFileName) throws IOException{

TreeSet <Car> TSCar = new TreeSet<Car>();

TreeMap <String, Car> TMCar = new TreeMap <String, Car>();

TSCar.addAll(new Car().ReadFile(IFileName));

for(Car c: TSCar){

TMCar.put(c.getMark(), c);

}

return TMCar;

}

}

Класс SortFirstField:

public class SortFirstField implements Comparator<Car>{

public int compare(Car obj1, Car obj2) {

String str1 = obj1.getMark();

String str2 = obj2.getMark();

return str1.compareTo(str2);

}

}

Класс SortSecondField:

public class SortSecondField implements Comparator<Car>{

public int compare(Car obj1, Car obj2) {

if(obj2.getYear\_made() == obj1.getYear\_made()) return 0;

else

if(obj2.getYear\_made() > obj1.getYear\_made()) return -1;

else return 1;

}

}

1. Выполнение программы

Данная лабораторная работа состоит из класса Programm, в котором запускается функция main. Сама функция представлена блоком switch, который выбирает, с каким видом контейнера будет проведена работа.

Для работы к контейнером LinkedList были написаны классы SortFirstField и SortSecondField, которые представляют собой реализация класса Comparator, который будет позволять сортировать объекта по их конкретным полям. Также был написан класс FileSaveCollection, который позволял записывать данные списка в файл.

Класс Adapter был написан с целью использования функции считывания с файла, которая была реализована для контейнера TreeSet. Данные сначала загружались в TreeSet, а после делились на ключевые поля и объект класса Car и записывались в TreeMap.

Работа с TreeSet:

Загрузить данные с файла?

yes

Enter string to find matches: BMW

Matche was found!

Mark: Audi

Year was made: 1980

Engine: 3.4

Max speed: 112.0

Mark: BMW

Year was made: 1990

Engine: 3.0

Max speed: 170.0

Mark: Ferari

Year was made: 2010

Engine: 4.0

Max speed: 170.0

Mark: Jugylini

Year was made: 1980

Engine: 7.0

Max speed: 300.0

Mark: Lada

Year was made: 2017

Engine: 1.0

Max speed: 666.0

Mark: Mers

Year was made: 1990

Engine: 7.7

Max speed: 90.0

Mark: Porsche

Year was made: 1992

Engine: 6.0

Max speed: 200.0

Работа с LinkedList:

Загрузить данные с файла?

yes

Выберите номер поля, по которому будет произведена сортировка(1,2):

2

Mark: Audi

Year was made: 1980

Engine: 3.4

Max speed: 112.0

Mark: Jugylini

Year was made: 1980

Engine: 7.0

Max speed: 300.0

Mark: Mers

Year was made: 1990

Engine: 7.7

Max speed: 90.0

Mark: BMW

Year was made: 1990

Engine: 3.0

Max speed: 170.0

Mark: Porsche

Year was made: 1992

Engine: 6.0

Max speed: 200.0

Mark: Ferari

Year was made: 2010

Engine: 4.0

Max speed: 170.0

Mark: Lada

Year was made: 2017

Engine: 1.0

Max speed: 666.0

Сохранить данные в файл?

yes

Работа с TreeMap:

Введите ключ элемента:

Audi

Совпадение было найдено.

Mark: Audi

Year was made: 1980

Engine: 3.4

Max speed: 112.0

Audi ---> 1980

Lada ---> 2017

Mers ---> 1990

ВЫВОДЫ

Была написана программа на языке Java в среде программирования Eclipse Mars. В данной работе были исследованы коллекции различных типов. Была реализована работа с файлами, переопределены основные функции сортировки объектов внутри коллекции. Были реализованы вспомогательные классы, которые предоставляли коллекциям тот или иной функционал.