Міністерство освіти і науки України

Харківський національний університет ім. В. Н. Каразіна

Факультет комп’ютерних наук

Кафедра безпеки інформаційних систем і технологій

Лабораторна робота №9

*з навчальної дисципліни*

«Спеціалізовані мови програмування та проектування електронних елементів і систем»

Виконав:

Студент групи КБ-41

Кравченко Є.М.

Перевірив:

Олійников Р.В.

Харків – 2020 р.

1.Загрузить из файла множество слов, вывести все слова (каждое - только один раз) тремя способами: итераторами, for-each, toString() для Set<>. Применять HashSet<>.

**private static void** task1() {  
 Set<String> set = **new** HashSet<>();  
 **try** (Scanner scanner = **new** Scanner(**new** FileInputStream(**"lab9\_task1.txt"**));) {  
 **while** (scanner.hasNextLine()) {  
 set.add(scanner.nextLine());  
 }  
 } **catch** (FileNotFoundException e) {  
 e.printStackTrace();  
 }  
 System.***out***.println(**"toString()"**);  
 System.***out***.println(set);  
 System.***out***.println(**"for-each"**);  
 **for** (String s : set) {  
 System.***out***.println(s);  
 }  
 System.***out***.println(**"iterator"**);  
 Iterator<String> iterator = set.iterator();  
 **while** (iterator.hasNext()) {  
 System.***out***.println(iterator.next());  
 }  
}

2.Из двух файлов сформировать два множества слов, вывести их объединение, пересечение, разность.

**private static void** task2() {  
 Set<String> set1 = **new** HashSet<>();  
 Set<String> set2 = **new** HashSet<>();  
 **try** (Scanner scanner = **new** Scanner(**new** FileInputStream(**"lab9\_task2\_1.txt"**));) {  
 **while** (scanner.hasNextLine()) {  
 set1.add(scanner.nextLine());  
 }  
 } **catch** (FileNotFoundException e) {  
 e.printStackTrace();  
 }  
 **try** (Scanner scanner = **new** Scanner(**new** FileInputStream(**"lab9\_task2\_2.txt"**));) {  
 **while** (scanner.hasNextLine()) {  
 set2.add(scanner.nextLine());  
 }  
 } **catch** (FileNotFoundException e) {  
 e.printStackTrace();  
 }  
 Set<String> set3 = **new** HashSet<>();  
 System.***out***.println(**"Объединение:"**);  
 set3.addAll(set1);  
 set3.addAll(set2);  
 set3.forEach(System.***out***::println);  
 System.***out***.println(**"Пересечение"**);  
 set3.clear();  
 set3.addAll(set1);  
 set3.stream().filter(set2::contains).forEach(System.***out***::println);  
 System.***out***.println(**"Разность"**);  
 set3.clear();  
 set3.addAll(set1);  
 set3.addAll(set2);  
 set3.stream().filter(s -> !(set1.contains(s) && set2.contains(s))).forEach(System.***out***::println);  
}

3.Загрузить из файла множество слов, загрузить их в ArrayList<>, вывести частоту появления каждого слова (дополнительно использовать Set<> для коллекции уникальных элементов). Пользоваться только методами коллекций (не организовывать собственный подсчет через циклы и т.п.).

**private static void** task3() {  
 ArrayList<String> arrayList = **new** ArrayList<>();  
 Set<String> set = **new** HashSet<>();  
 **try** (Scanner scanner = **new** Scanner(**new** FileInputStream(**"lab9\_task3.txt"**))) {  
 **while** (scanner.hasNextLine()) {  
 String s = scanner.nextLine();  
 arrayList.add(s);  
 set.add(s);  
 }  
 } **catch** (FileNotFoundException e) {  
 e.printStackTrace();  
 }  
 **for** (String s : set) {  
 **int** size = arrayList.size();  
 arrayList.removeIf(s1 -> s1.equals(s));  
 System.***out***.println(s + **" "** + (size - arrayList.size()));  
 }  
}

4.Загрузить из файла множество слов, отсортировать (повторения допускаются), вывести на экран. Перемешать в случайном порядке, снова вывести.

**private static void** task4() {  
 ArrayList<String> arrayList = **new** ArrayList<>();  
 **try** (Scanner scanner = **new** Scanner(**new** FileInputStream(**"lab9\_task3.txt"**))) {  
 **while** (scanner.hasNextLine()) {  
 String s = scanner.nextLine();  
 arrayList.add(s);  
 }  
 } **catch** (FileNotFoundException e) {  
 e.printStackTrace();  
 }  
 arrayList.sort(String::compareTo);  
 System.***out***.println(arrayList);  
 Collections.*shuffle*(arrayList);  
 System.***out***.println(arrayList);  
}

5.В файле в текстовом формате занесены данные о студентах (формат жестко фиксирован), в каждой строке содержится:

фамилия, группа, название учебной дисциплины (одним словом), успеваемость по дисциплине за семестр.

Фамилии студентов, дисциплин и пр. могут повторяться (студент изучает несколько предметов. Данные находятся в произвольном порядке, например:

Петров КБ-41 java 99

Семенов КС-42 ПЗС 92

Петров КБ-41 КСС 95

Петров КБ-41 КСМ 95

Иванов КУ-33 экономика 80

Семенов КС-42 ТКС 94

Прочесть данные из файла в LinkedList, состоящий из элементов класса Student, содержащего необходимые поля (фамилия, группа, дисциплина, успеваемость). Класс Student реализует интерфейс Comparable<Student> (единственный метод интерфейса - compareTo(Student other); по умолчанию реализуется сортировка по фамилии).

**private static void** task5() {  
 List<Student> students = **new** LinkedList<>();  
 **try** (Scanner scanner = **new** Scanner(**new** FileInputStream(**"lab9\_task5.txt"**))) {  
 **while** (scanner.hasNextLine()) {  
 String s = scanner.nextLine();  
 String[] params = s.split(**" "**);  
 students.add(**new** Student(params[0], params[1], params[2], Integer.*parseInt*(params[3])));  
 }  
 } **catch** (FileNotFoundException e) {  
 e.printStackTrace();  
 }  
 System.***out***.println(**"Список дисциплин:"**);  
 Set<Student> courses = **new** TreeSet<>(Comparator.*comparing*(Student::getCourse));  
 courses.addAll(students);  
 courses.stream().map(Student::getCourse).forEach(System.***out***::println);  
 System.***out***.println(**"Список групп:"**);  
 Set<Student> groups = **new** TreeSet<>(Comparator.*comparing*(Student::getGroup));  
 groups.addAll(students);  
 groups.stream().map(Student::getGroup).forEach(System.***out***::println);  
}

**public class** Student **implements** Comparable<Student> {  
 **private** String **surname**;  
 **private** String **group**;  
 **private** String **course**;  
 **private int mark**;  
  
 **public** Student(String surname, String group, String course, **int** mark) {  
 **this**.**surname** = surname;  
 **this**.**group** = group;  
 **this**.**course** = course;  
 **this**.**mark** = mark;  
 }  
  
 **public** String getSurname() {  
 **return surname**;  
 }  
  
 **public void** setSurname(String surname) {  
 **this**.**surname** = surname;  
 }  
  
 **public** String getGroup() {  
 **return group**;  
 }  
  
 **public void** setGroup(String group) {  
 **this**.**group** = group;  
 }  
  
 **public** String getCourse() {  
 **return course**;  
 }  
  
 **public void** setCourse(String course) {  
 **this**.**course** = course;  
 }  
  
 **public int** getMark() {  
 **return mark**;  
 }  
  
 **public void** setMark(**int** mark) {  
 **this**.**mark** = mark;  
 }  
  
 @Override  
 **public int** compareTo(Student o) {  
 **return surname**.compareTo(o.**course**);  
 }  
}