**Министерство образования Республики Беларусь**

**Учреждение образования**

**«Брестский государственный технический университет»**

**Кафедра ИИТ**

**Лабораторная работа №3**

По дисциплине: «ССП»

Вариант 10

**Выполнил:** Студент 3 курса

Группы ПО-8 Дымша А.Г.

**Проверил:**

Крощенко А.А

**Брест 2024**

**Лабораторная работа №3**

**Цель работы:** научиться создавать и использовать классы в программах на языке программирования Java

**Задание 1:** 10) Множество символов переменной мощности – Предусмотреть возможность пересечения двух множеств, вывода на печать элементов множества, а так же метод, определяющий, принадлежит ли указанное значение множеству. Класс должен содержать методы, позволяющие добавлять и удалять элемент в/из множества. Конструктор должен позволить создавать объекты с начальной инициализацией. Реализацию множества осуществить на базе структуры ArrayList. Реализовать метод equals, выполняющий сравнение объектов данного типа.

**Код программы:**

import java.util.ArrayList;

import java.util.Objects;

class CharacterSet {

    private ArrayList<Character> set;

    public CharacterSet() {

        set = new ArrayList<>();

    }

    public CharacterSet(Character... characters) {

        set = new ArrayList<>();

        for (Character c : characters) {

            add(c);

        }

    }

    public void add(Character c) {

        if (!set.contains(c)) {

            set.add(c);

        }

    }

    public void remove(Character c) {

        set.remove(c);

    }

    public boolean contains(Character c) {

        return set.contains(c);

    }

    public CharacterSet intersection(CharacterSet other) {

        CharacterSet result = new CharacterSet();

        for (Character c : set) {

            if (other.contains(c)) {

                result.add(c);

            }

        }

        return result;

    }

    @Override

    public String toString() {

        StringBuilder sb = new StringBuilder();

        sb.append("{");

        for (int i = 0; i < set.size(); i++) {

            sb.append(set.get(i));

            if (i != set.size() - 1) {

                sb.append(", ");

            }

        }

        sb.append("}");

        return sb.toString();

    }

    @Override

    public boolean equals(Object o) {

        if (this == o)

            return true;

        if (o == null || getClass() != o.getClass())

            return false;

        CharacterSet that = (CharacterSet) o;

        return Objects.equals(set, that.set);

    }

    @Override

    public int hashCode() {

        return Objects.hash(set);

    }

}

class Main {

    public static void main(String[] args) {

        // Создание пустого множества

        CharacterSet set1 = new CharacterSet();

        System.out.println("Пустое множество: " + set1);

        // Создание множества с элементами

        CharacterSet set2 = new CharacterSet('a', 'b', 'c');

        System.out.println("Множество с элементами: " + set2);

        // Добавление элемента

        set2.add('d');

        System.out.println("После добавления 'd': " + set2);

        // Удаление элемента

        set2.remove('b');

        System.out.println("После удаления 'b': " + set2);

        // Проверка принадлежности элемента

        System.out.println("Содержит 'a'? " + set2.contains('a'));

        System.out.println("Содержит 'b'? " + set2.contains('b'));

        // Пересечение множеств

        CharacterSet set3 = new CharacterSet('c', 'd', 'e');

        CharacterSet intersection = set2.intersection(set3);

        System.out.println("Пересечение " + set2 + " и " + set3 + ": " + intersection);

        // Сравнение множеств

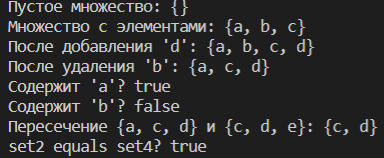
        CharacterSet set4 = new CharacterSet('a', 'c', 'd');

        System.out.println("set2 equals set4? " + set2.equals(set4));

    }

}

**Результат работы программы:**



**Задание 2:** 10) Частотный словарь Составить программу, которая формирует англо-русский словарь. Словарь должен содержать английское слово, русское слово и количество обращений к слову. Программа должна:

• обеспечить начальный ввод словаря (по алфавиту) с конкретными значениями счетчиков обращений;

• формирует новое дерево, в котором слова отсортированы не по алфавиту, а по количеству обращений.

• Реализовать возможность добавления новых слов, удаления существующих, поиска нужного слова, выполнять просмотр обоих вариантов словаря

**Код программы:**

import java.io.BufferedReader;

import java.io.File;

import java.io.FileReader;

import java.io.IOException;

import java.util.Map;

import java.util.Scanner;

import java.util.TreeMap;

import java.nio.file.Path;

import java.nio.file.Paths;

import java.util.List;

import java.nio.file.Files;

import java.nio.charset.StandardCharsets;

import java.io.IOException;

import java.nio.charset.StandardCharsets;

import java.nio.file.Files;

import java.nio.file.Paths;

import java.util.Arrays;

import java.util.List;

class CreateDictionaryFile {

    public static void main(String[] args) {

        List<String> lines = Arrays.asList(

                "hello,привет,10",

                "world,мир,20",

                "java,ява,30");

        try {

            Files.write(Paths.get("dictionary.txt"), lines, StandardCharsets.UTF\_8);

            System.out.println("File created successfully.");

        } catch (IOException e) {

            System.out.println("Error writing file: " + e.getMessage());

        }

    }

}

class WordEntry {

    String russianWord;

    int count;

    WordEntry(String russianWord, int count) {

        this.russianWord = russianWord;

        this.count = count;

    }

}

class DictionaryApp {

    private static TreeMap<String, WordEntry> dictionaryByWord = new TreeMap<>();

    private static TreeMap<WordEntry, String> dictionaryByCount = new TreeMap<>((a, b) -> b.count - a.count);

    public static void main(String[] args) {

        loadDictionary("dictionary.txt");

        printDictionaryByWord();

        printDictionaryByCount();

        Scanner scanner = new Scanner(System.in);

        while (true) {

            System.out.println("Enter command (add, remove, search, exit):");

            String command = scanner.nextLine();

            switch (command) {

                case "add":

                    addWord(scanner);

                    break;

                case "remove":

                    removeWord(scanner);

                    break;

                case "search":

                    searchWord(scanner);

                    break;

                case "exit":

                    return;

                default:

                    System.out.println("Invalid command");

            }

        }

    }

    private static void loadDictionary(String filename) {

        try {

            Path path = Paths.get(filename);

            List<String> lines = Files.readAllLines(path, StandardCharsets.UTF\_8);

            for (String line : lines) {

                String[] parts = line.split(",");

                String englishWord = parts[0];

                String russianWord = parts[1];

                int count = Integer.parseInt(parts[2]);

                dictionaryByWord.put(englishWord, new WordEntry(russianWord, count));

                dictionaryByCount.put(new WordEntry(russianWord, count), englishWord);

            }

        } catch (IOException e) {

            System.out.println("Error reading file: " + e.getMessage());

        }

    }

    private static void printDictionaryByWord() {

        System.out.println("Dictionary by word:");

        for (Map.Entry<String, WordEntry> entry : dictionaryByWord.entrySet()) {

            System.out.println(

                    entry.getKey() + " - " + entry.getValue().russianWord + " (" + entry.getValue().count + ")");

        }

    }

    private static void printDictionaryByCount() {

        System.out.println("Dictionary by count:");

        for (Map.Entry<WordEntry, String> entry : dictionaryByCount.entrySet()) {

            System.out

                    .println(entry.getValue() + " - " + entry.getKey().russianWord + " (" + entry.getKey().count + ")");

        }

    }

    private static void addWord(Scanner scanner) {

        System.out.print("Enter English word: ");

        String englishWord = scanner.nextLine();

        System.out.print("Enter Russian word: ");

        String russianWord = scanner.nextLine();

        System.out.print("Enter count: ");

        int count = scanner.nextInt();

        scanner.nextLine(); // consume newline character

        WordEntry wordEntry = new WordEntry(russianWord, count);

        dictionaryByWord.put(englishWord, wordEntry);

        dictionaryByCount.put(wordEntry, englishWord);

    }

    private static void removeWord(Scanner scanner) {

        System.out.print("Enter English word to remove: ");

        String englishWord = scanner.nextLine();

        if (dictionaryByWord.containsKey(englishWord)) {

            WordEntry wordEntry = dictionaryByWord.get(englishWord);

            dictionaryByWord.remove(englishWord);

            dictionaryByCount.remove(wordEntry);

        } else {

            System.out.println("Word not found");

        }

    }

    private static void searchWord(Scanner scanner) {

        System.out.println("Enter English word:");

        String englishWord = scanner.nextLine();

        if (dictionaryByWord.containsKey(englishWord)) {

            WordEntry entry = dictionaryByWord.get(englishWord);

            System.out.println("Russian word: " + new String(entry.russianWord.getBytes(), StandardCharsets.UTF\_8));

            System.out.println("Count: " + entry.count);

        } else {

            System.out.println("Word not found");

        }

    }

}

**Результаты работы программы:**

