

## Лабораторная работа №3

### Сабо Е.О. Вариант-10

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

**Задание 1** Реализовать простой класс. Требования к выполнению

- Реализовать пользовательский класс по варианту.
- Создать другой класс с методом main, в котором будут находиться примеры использования пользовательского класса. Для каждого класса
- Создать поля классов
- Создать методы классов
- Добавьте необходимые get и set методы (по необходимости)
- Укажите соответствующие модификаторы видимости
- Добавьте конструкторы
- Переопределить методы toString() и equals()

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:

```
package SSP.Lab3.Task1;

public class Solution {

    public static void main(String[] args) {
        CharMultitude fistSet = new CharMultitude(new
char[]{'a','a','b','c'});
        CharMultitude secondSet = new CharMultitude(new char[]{'a','d','f'});
        System.out.println(fistSet);
        System.out.println(secondSet);
        System.out.println("Пересечение множеств");
        System.out.println(CharMultitude.intersection(fistSet, secondSet));

        System.out.println("Добавление");
        fistSet.add('l');
        System.out.println(fistSet);

        System.out.println("Удаление");
        secondSet.remove('f');
        System.out.println(secondSet);

        CharMultitude fistEq = new CharMultitude(new char[]{'a','d','f'});
        CharMultitude secondEq = new CharMultitude(new char[]{'a','d','f'});
        CharMultitude thirdEq = new CharMultitude(new
char[]{'a','d','f','t'});
        System.out.println(fistEq.equals(secondEq));
        System.out.println(fistEq.equals(thirdEq));
    }
}
```

```
}  
}
```

### Класс:

```
package SSP.Lab3.Task1;  
  
import java.util.*;  
  
public class CharMultitude {  
    private int power;  
    private List<Character> list = new ArrayList<>();  
  
    public CharMultitude(CharSequence array) {  
        for(Character ch : array.toString().toCharArray())  
            list.add(ch);  
        power = list.size();  
        sortArray();  
    }  
  
    public CharMultitude(char[] chars) {  
        for(char ch : chars)  
            list.add(ch);  
        power = list.size();  
        sortArray();  
    }  
  
    public CharMultitude(List<Character> list) {  
        for(Character ch : list)  
            this.list.add(ch);  
        this.power = list.size();  
        sortArray();  
    }  
  
    private void sortArray(){  
        Collections.sort(this.list);  
    }  
  
    public void add(char a){  
        this.list.add(a);  
        this.sortArray();  
    }  
  
    public void remove(char a){  
        if(this.list.contains((Character)a)){  
            int id = list.indexOf((Character)a);  
            list.remove(id);  
        }  
    }  
  
    @Override  
    public int hashCode() {  
        int hash = 1;  
        for (int i = 0; i < power; i++) {  
            hash+=31*this.list.get(i);  
        }  
        return hashCode();  
    }  
  
    @Override  
    public boolean equals(Object o) {  
        if (this == o) return true;  
        if (o == null || getClass() != o.getClass()) return false;  
        CharMultitude that = (CharMultitude) o;  
  
        if(getPower() != that.getPower())  
            return false;  
        int i=0;  
        for (; i < getPower(); i++) {
```

```

        if(this.list.get(i) != that.list.get(i))
            return false;
    }
    if(i == that.list.size())
        return true;
    return false;
}

@Override
public String toString() {
    StringBuilder stringBuilder = new StringBuilder("{}");
    for(Character character : this.list)
        stringBuilder.append(" "+character+ " ");
    stringBuilder.append("}");
    return stringBuilder.toString();
}

public static String intersection(CharMultitude a, CharMultitude b){
    Map<Character, Integer> map= new HashMap<>();
    List<Character> list = new ArrayList<>();
    for(Character ch : a.list){
        if(!map.containsKey(ch)) {
            int frequencyA = Collections.frequency(a.list,ch);
            int frequencyB =Collections.frequency(b.list,ch);
            map.put(ch,Math.min(frequencyA,frequencyB));
        }
    }
    for(Map.Entry<Character, Integer> entry : map.entrySet()){
        for (int i = 0; i < entry.getValue(); i++) {
            list.add(entry.getKey());
        }
    }
    return new CharMultitude(list).toString();
}

public int getPower() {
    return list.size();
}
}

```

### Результат выполнения:

```

{ a a b c }
{ a d f }
Пересечение множеств
{ a }
Добавление
{ 1 a a b c }
Удаление
{ a d }
{ a d f } == { a d f } true
{ a d f } == { a d f t } false

```

## Задание 2:

Разработать автоматизированную систему на основе некоторой структуры данных, манипулирующей объектами пользовательского класса. Реализовать требуемые функции обработки данных Требования к выполнению

- Задание посвящено написанию классов, решающих определенную задачу автоматизации;
- Данные для программы загружаются из файла (формат произволен). Файл создать и написать вручную

### 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:

```
package SSP.Lab3.Task2;

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.*;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class Solution {
    final static String file = "D:\\SSP\\src\\SSP\\Lab3\\Task2\\input.txt";
    public static void main(String[] args) throws IOException {
        Dictionary eng = new Dictionary(file);
        //добавление элементов
        eng.add("cute", "милый", 10);
        eng.add("cute", "милый");
        eng.add("house", "дом", 7);
        eng.add("good", "хорошо", 9);
        System.out.println("Словарь");
        eng.show();
        System.out.println("Сортировка по количеству обращений");
        eng.showSort();
        System.out.println("Просмотр обоих вариантов словаря\nEnglish -
Русский\n");
        eng.show();
        System.out.println("Русский - English\n");
        new Dictionary(eng.reverse()).show();

        System.out.println("Удаление дерева");
        System.out.println("Удаление cute");
        eng.remove("дерево");
        eng.remove("cute");
    }
}
```

```

        System.out.println("Словарь");
        eng.show();
        System.out.println("Поиск read");
        eng.find("read");
        eng.show();
    }

    public static class Dictionary{

        public Dictionary() {
            this.dict = new HashMap<>();
        }
        public Dictionary(String file) throws IOException {
            this.dict = new HashMap<>();
            BufferedReader reader = new BufferedReader(new FileReader(file));
            Pattern pattern = Pattern.compile("^(.+) (.+)$");
            String str = " ";

            while((str = reader.readLine()) != null){
                Matcher matcher = pattern.matcher(str);
                while(matcher.find()){
                    dict.put(new Word(matcher.group(1)), new
Word(matcher.group(2)));
                }
            }

            public Dictionary(Map<Word, Word> dict) {
                this.dict = dict;
            }

            Map<Word, Word> dict;
            public void add(String word1, String word2){
                Word temp = new Word(word1);
                if(dict.containsKey(temp)){
                    dict.forEach((k,v)->{if (k.equals(temp)) k.count +=1;});
                }
                else {
                    dict.put(temp, new Word(word2));
                }
            }
            public void add(String word1, String word2, int count){
                Word temp = new Word(word1, count);
                if(dict.containsKey(temp)){
                    dict.forEach((k,v)->{if (k.equals(temp)) k.count +=1;});
                }
                else {
                    dict.put(temp, new Word(word2));
                }
            }
            public void remove(String word1){
                dict.entrySet().removeIf((k) -> (k.getKey().word.equals(word1) ||
k.getValue().word.equals(word1)));
            }
            public void show(){
                for(Map.Entry<Word, Word> entry: dict.entrySet()){
                    System.out.println(entry.getKey().word + " " +
entry.getValue().word + " " + entry.getKey().count);
                }
                System.out.println();
            }

            public void showSort(){
                Map<Word, Word> treeDict = new TreeMap<>(dict);
                treeDict.forEach((k,v)-> System.out.println(k.count + " " +
k.word + " " + v.word));
                System.out.println();
            }
        }
    }

```

```

    public Map<Word, Word> reverse() {
        Map<Word, Word> rev = new HashMap<>();
        dict.forEach((k, v) -> { rev.put(v, k); });
        return rev;
    }

    public void find(String str) {
        dict.forEach((k, v) -> {
            if(k.word.equals(str))
                System.out.println("Перевод слова " + str + " - " + v.word);
            k.count++;
        });
    }
}

public static class Word implements Comparable<Word> {
    public Word(String word) {
        this.word = word;
        this.count = 0;
    }

    public Word(String word, int count) {
        this.word = word;
        this.count = count;
    }

    String word;
    int count;

    @Override
    public int hashCode() {
        int magic = 31;
        int res = 0;
        res = res * magic + word.hashCode();
        return res;
    }

    @Override
    public boolean equals(Object obj) {
        return this.word.equals(((Word) obj).word);
    }

    @Override
    public int compareTo(Word o) {
        if(this.count == o.count)
            return 0;
        else if (this.count > o.count)
            return 1;
        else
            return -1;
    }
}
}

```

Результат работы программы:

Словарь

read читать 0

tree дерево 0

cute милый 11

house дом 7

good хорошо 9

Сортировка по количеству обращений

0 read дерево

7 house дом

9 good хорошо

11 cute милый

Просмотр обоих вариантов словаря

English - Русский

read читать 0

tree дерево 0

cute милый 11

house дом 7

good хорошо 9

Русский - English

дом house 0

читать read 0

дерево tree 0

хорошо good 0

милый cute 0

Удаление дерево

Удаление cute

Словарь

read читать 0

house дом 7

good хорошо 9

Поиск read


Перевод слова read - читать

read читать 1

house дом 8

good хорошо 10

Исходные данные:

 input.txt – Блокнот

Файл Правка Формат Вид Справка

|tree дерево

read читать

Вывод: в ходе работы научился создавать и использовать классы в программах на языке программирования Java.