МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”
КАФЕДРА ИИТ

ОТЧЁТ

по лабораторной работе №3

Выполнил:

Студент 3 курса
группы ПО-9
Кучко Ярослав Валерьевич

Проверил:

Крощенко А. А.

Брест 2024

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

Вариант 10

Задание 1. 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.

Класс множества.

```
public class CustomCharArray {

    private final ArrayList<Character> values;

    public CustomCharArray() {
        values = new ArrayList<>();
    }

    public CustomCharArray(List<Character> values) {
        this.values = new ArrayList<>(values);
    }

    public ArrayList<Character> getValues() {
        return values;
    }

    public boolean add (Character character) {
        return values.add(character);
    }

    public boolean remove (Character character) {
        return values.remove(character);
    }

    public boolean contains(Character character) {
        return values.contains(character);
    }

    public CustomCharArray intersection(CustomCharArray obj) {
        List<Character> valClone = new ArrayList<>(values);
        List<Character> objClone = new ArrayList<>(obj.getValues());

        ArrayList<Character> result = new ArrayList<>();
        for (int i = 0; i < valClone.size(); i++) {
            Character val = valClone.get(i);
            if (objClone.contains(val)) {
                result.add(val);
                if (valClone.remove(val)) {
                    i--;
                }
                objClone.remove(val);
            }
        }

        return new CustomCharArray(result);
    }

    @Override
    public String toString() {
```

```
        return "CustomCharArray{" +
                "values=" + values +
                '}';
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        CustomCharArray that = (CustomCharArray) o;
        return Objects.equals(values, that.values);
    }

    @Override
    public int hashCode() {
        return Objects.hash(values);
    }
}
```

Примеры использования.

```
public class Main {
    public static void main(String[] args) {
        CustomCharArray array = new CustomCharArray(List.of('a', 'b', 'b',
'c'));
        System.out.println("array:");
        System.out.println(array);
        System.out.println();

        System.out.println("array.add('f')");
        array.add('f');
        System.out.println(array);
        System.out.println();

        System.out.println("array.remove('c')");
        array.remove('c');
        System.out.println(array);
        System.out.println();

        System.out.println("array.contains('c')");
        System.out.println(array.contains('c'));
        System.out.println();

        System.out.println("array.contains('f')");
        System.out.println(array.contains('f'));
        System.out.println();

        CustomCharArray array1 = new CustomCharArray(List.of('b', 'b', 'b',
'a'));
        System.out.println("array1:");
        System.out.println(array1);
        System.out.println();

        System.out.println("array.intersection(array1)");
        System.out.println(array.intersection(array1));
        System.out.println();

        System.out.println("array.equals(array1)");
        System.out.println(array.equals(array1));
        System.out.println();

        CustomCharArray array2 = new CustomCharArray(List.of('a', 'b', 'b',
'f'));
        System.out.println("array2:");
        System.out.println(array2);
```

```
        System.out.println();

        System.out.println("array.equals(array2)");
        System.out.println(array.equals(array2));
        System.out.println();
    }
}
```

Результат работы.

```
array:
CustomCharArray{values=[a, b, b, c]}

array.add('f')
CustomCharArray{values=[a, b, b, c, f]}

array.remove('c')
CustomCharArray{values=[a, b, b, f]}

array.contains('c')
false

array.contains('f')
true

array1:
CustomCharArray{values=[b, b, b, a]}

array.intersection(array1)
CustomCharArray{values=[a, b, b]}

array.equals(array1)
false

array2:
CustomCharArray{values=[a, b, b, f]}

array.equals(array2)
true
```

Задание 2. 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.

Класс словаря.

```
public class Dictionary {

    private static final Comparator<DictionaryItem> alphabetComparator = new
Comparator<DictionaryItem>() {
        @Override
        public int compare(DictionaryItem o1, DictionaryItem o2) {
            return o1.getValueRu().compareTo(o2.getValueRu());
        }
    };

    private static final Comparator<DictionaryItem> resuestsComparator = new
Comparator<DictionaryItem>() {
        @Override
        public int compare(DictionaryItem o1, DictionaryItem o2) {
            return o1.getRequestsAmount().compareTo(o2.getRequestsAmount());
        }
    };

    private final SortedSet<DictionaryItem> itemsByAlphabet;
    private final SortedSet<DictionaryItem> itemsByRequests;

    public Dictionary() {
        itemsByAlphabet = new TreeSet<>(alphabetComparator);
        itemsByRequests = new TreeSet<>(resuestsComparator);
    }

    public boolean add(DictionaryItem item) {
        return itemsByAlphabet.add(item) && itemsByRequests.add(item);
    }

    public boolean remove(DictionaryItem item) {
        return itemsByAlphabet.remove(item) && itemsByRequests.remove(item);
    }

    public DictionaryItem search(String value) {
        return             itemsByAlphabet.stream().filter(item             ->
item.getValueRu().equalsIgnoreCase(value)).findFirst()
                .orElse(itemsByAlphabet.stream().filter(item                ->
item.getValueEn().equalsIgnoreCase(value)).findFirst().orElse(null));
    }

    public void readFile(String fileName) {
        try {
            BufferedReader    bufferedReader    =    new    BufferedReader(new
FileReader(fileName));
            for (String line = bufferedReader.readLine(); line != null; line
= bufferedReader.readLine()) {
                String[] words = line.split("\\s+");
```

```
                DictionaryItem item = new DictionaryItem(words[0], words[1],
Integer.valueOf(words[2]));
                itemsByAlphabet.add(item);
                itemsByRequests.add(item);
            }
        } catch (IOException | IndexOutOfBoundsException e) {
            throw new FileOnReadException(fileName);
        }
    }

    public void writeFile(String fileName) {
        try {
            BufferedWriter writer = new BufferedWriter(new
FileWriter(fileName));
            itemsByAlphabet.forEach(item -> {
                try {
                    writer.write(item.getValueRu() + " " + item.getValueEn()
+ " " + item.getRequestsAmount());
                    writer.newLine();
                } catch (IOException e) {
                    throw new RuntimeException(e);
                }
            });

            writer.close();
        } catch (IOException e) {
            throw new FileOnWriteException(fileName);
        }
    }

    public SortedSet<DictionaryItem> getItemsByAlphabet() {
        return itemsByAlphabet;
    }

    public SortedSet<DictionaryItem> getItemsByRequests() {
        return itemsByRequests;
    }
}
```

Класс элемента словаря.

```
public class DictionaryItem {

    private String valueRu;
    private String valueEn;
    private Integer requestsAmount;

    public DictionaryItem() {
    }

    public DictionaryItem(String valueRu, String valueEn, Integer
requestsAmount) {
        this.valueRu = valueRu;
        this.valueEn = valueEn;
        this.requestsAmount = requestsAmount;
    }

    public String getValueRu() {
        return valueRu;
    }

    public void setValueRu(String valueRu) {
```

```
        this.valueRu = valueRu;
    }

    public String getValueEn() {
        return valueEn;
    }

    public void setValueEn(String valueEn) {
        this.valueEn = valueEn;
    }

    public Integer getRequestsAmount() {
        return requestsAmount;
    }

    public void setRequestsAmount(Integer requestsAmount) {
        this.requestsAmount = requestsAmount;
    }

    @Override
    public String toString() {
        return "\nDictionaryItem{" +
                "valueRu='" + valueRu + '\'' +
                ", valueEn='" + valueEn + '\'' +
                ", requestsAmount=" + requestsAmount +
                '}';
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        DictionaryItem that = (DictionaryItem) o;
        return        Objects.equals(valueRu,        that.valueRu)        &&
Objects.equals(valueEn,   that.valueEn)   &&   Objects.equals(requestsAmount,
that.requestsAmount);
    }

    @Override
    public int hashCode() {
        return Objects.hash(valueRu, valueEn, requestsAmount);
    }
}
```

Примеры использования.

```
public class Main {
    public static void main(String[] args) {
        Dictionary dictionary = new Dictionary();
        dictionary.readFile(args[0]);

        System.out.println("dictionary by alphabet");
        System.out.println(dictionary.getItemsByAlphabet());
        System.out.println();
        System.out.println("dictionary by requests amount");
        System.out.println(dictionary.getItemsByRequests());
        System.out.println();

        System.out.println("dictionary.add()");
        dictionary.add(new DictionaryItem("собака", "dog", 67));

        System.out.println("dictionary by alphabet");
```

```
        System.out.println(dictionary.getItemsByAlphabet());
        System.out.println();
        System.out.println("dictionary by requests amount");
        System.out.println(dictionary.getItemsByRequests());
        System.out.println();

        System.out.println("dictionary.remove()");
        dictionary.remove(dictionary.search("кот"));

        System.out.println("dictionary by alphabet");
        System.out.println(dictionary.getItemsByAlphabet());
        System.out.println();
        System.out.println("dictionary by requests amount");
        System.out.println(dictionary.getItemsByRequests());
        System.out.println();

        System.out.println("dictionary.search(\"alphabet\")");
        System.out.println(dictionary.search("alphabet"));
    }
}
```

Результат работы.

```
> java Main D:\dict.txt
dictionary by alphabet
[
DictionaryItem{valueRu='алфавит', valueEn='alphabet', requestsAmount=21},
DictionaryItem{valueRu='кот', valueEn='cat', requestsAmount=89},
DictionaryItem{valueRu='словарь', valueEn='dictionary', requestsAmount=100},
DictionaryItem{valueRu='чтение', valueEn='read', requestsAmount=76}]

dictionary by requests amount
[
DictionaryItem{valueRu='алфавит', valueEn='alphabet', requestsAmount=21},
DictionaryItem{valueRu='чтение', valueEn='read', requestsAmount=76},
DictionaryItem{valueRu='кот', valueEn='cat', requestsAmount=89},
DictionaryItem{valueRu='словарь', valueEn='dictionary', requestsAmount=100}]

dictionary.add()
dictionary by alphabet
[
DictionaryItem{valueRu='алфавит', valueEn='alphabet', requestsAmount=21},
DictionaryItem{valueRu='кот', valueEn='cat', requestsAmount=89},
DictionaryItem{valueRu='словарь', valueEn='dictionary', requestsAmount=100},
DictionaryItem{valueRu='собака', valueEn='dog', requestsAmount=67},
DictionaryItem{valueRu='чтение', valueEn='read', requestsAmount=76}]

dictionary by requests amount
[
DictionaryItem{valueRu='алфавит', valueEn='alphabet', requestsAmount=21},
DictionaryItem{valueRu='собака', valueEn='dog', requestsAmount=67},
DictionaryItem{valueRu='чтение', valueEn='read', requestsAmount=76},
DictionaryItem{valueRu='кот', valueEn='cat', requestsAmount=89},
DictionaryItem{valueRu='словарь', valueEn='dictionary', requestsAmount=100}]

dictionary.remove()
dictionary by alphabet
[
DictionaryItem{valueRu='алфавит', valueEn='alphabet', requestsAmount=21},
DictionaryItem{valueRu='словарь', valueEn='dictionary', requestsAmount=100},
DictionaryItem{valueRu='собака', valueEn='dog', requestsAmount=67},
DictionaryItem{valueRu='чтение', valueEn='read', requestsAmount=76}]
```

```
dictionary by requests amount
[
DictionaryItem{valueRu='алфавит', valueEn='alphabet', requestsAmount=21},
DictionaryItem{valueRu='собака', valueEn='dog', requestsAmount=67},
DictionaryItem{valueRu='чтение', valueEn='read', requestsAmount=76},
DictionaryItem{valueRu='словарь', valueEn='dictionary', requestsAmount=100}]

dictionary.search("alphabet")

DictionaryItem{valueRu='алфавит', valueEn='alphabet', requestsAmount=21}
```

Вывод: в ходе выполнения лабораторной работы получен опыт в создании использовании классов на языке программирования Java.