МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт информационных технологий (ИТ) Кафедра инструментального и прикладного программного обеспечения (ИиППО)

Дисциплина «Программирование на языке Джава»

ОТЧЕТ ПО ПРАКТИЧЕСКОМУ ЗАНЯТИЮ №8

Выполнил студент группы ИНБО-02-20 Деревянкин Н.А.

Принял Степанов П.В.

Практическая работа выполнена «___»_____2021 г.

«_____» «___»_____2021 г.

Отметка о выполнении

Москва – 2021 г.

СОДЕРЖАНИЕ

Цель работы

Изучение списков ожидания.

Задание

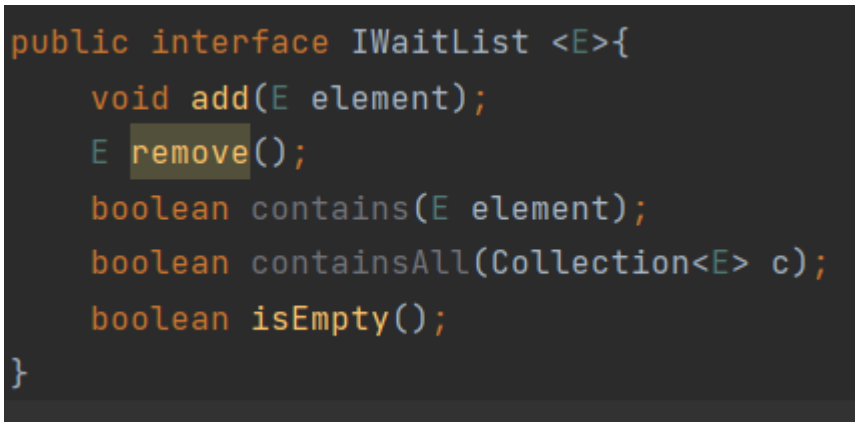
1. Исследуйте UML диаграмму классов на рисунке 1 и понаблюдайте, как она выражает то, что мы говорили выше в словах. Убедитесь, что вы понимаете все аспекты диаграммы.
2. Расширить и модифицировать исходный код WaitList, как необходимо, чтобы полностью реализовать всю схему UML. Включить комментарии Javadoc. Обратите внимание на переключение ролей после реализации каждого интерфейса / класса!
3. Изучение работу метода main(), которая использует ваши новые классы и интерфейс.

Выполнение работы

Приступив к выполнению, я повторил UML диаграмму, соблюдая все отношения классов. Прописав и переопределив все методы, а также, создав конструкторы, я создал класс Main для тестирования всех списков.

Код выполненной работы

Здесь в нескольких скриншотах можно увидеть, как выглядит код полученного задания и его вывод.



```
public interface IWaitList <E>{
    void add(E element);
    E remove();
    boolean contains(E element);
    boolean containsAll(Collection<E> c);
    boolean isEmpty();
}
```

Рисунок 1 – Интерфейс IWiatList<E>

```java
public class WaitList <E> implements IWaitList<E>{
    protected ConcurrentLinkedQueue<E> content;

    public WaitList() { content = new ConcurrentLinkedQueue<>(); }

    @Override
    public String toString() {
        return "WaitList{" +
                "content=" + content +
                '}';
    }


    @Override
    public void add(E element) { content.add(element); }

    @Override
    public E remove() { return content.remove(); }

    @Override
    public boolean contains(E element) { return content.contains(element); }

    @Override
    public boolean containsAll(Collection<E> c) { return content.containsAll(c); }

    @Override
    public boolean isEmpty() { return content.isEmpty(); }

}
```

Рисунок 2 – Класс WaitList

```java
public class BoundedWaitList <E> extends WaitList<E>{
    private int capacity = 5;

    public BoundedWaitList(int capacity) { this.capacity = capacity; }


    public int getCapacity() { return capacity; }

    public void add(E element) {
        if (content.size() < capacity) {
            content.add(element);
        }
    }

    @Override
    public String toString() {
        return "BoundedWaitList{" +
                "capacity=" + capacity +
                ", content=" + content +
                '}';
    }
}
```

Рисунок 3 – Класс BoundedWaitList

```java
public class UnfairWaitList <E> extends WaitList<E>{
    public UnfairWaitList() {
    }
    public void remove(E element) {
        content.remove(element);
    }

    public void moveToBack(E element) {
        content.remove(element);
        content.add(element);
    }
}
```

Рисунок 4 – Класс UnfairWaitList

```java
public class Main {
    public static void main(String[] args) {
        System.out.println("WaitList");
        WaitList<Integer> list1 = new WaitList<>();
        System.out.println("List: " + list1);
        System.out.println("Checking for 'isEmpty': " + list1.isEmpty());
        for (int i = 0; i < 5; i++) {
            list1.add(i);
        }
        System.out.println("Numbers were added to the list");
        System.out.println("List: " + list1);
        list1.remove();
        System.out.println("The first number was deleted");
        System.out.println("List: " + list1);
        System.out.println("isEmpty: " + list1.isEmpty() + "\n\n");
```

Рисунок 5 - Main

Вывод

В результате выполнения данной практической работы я научился работать со списками.

GitHub - https://github.com/dronikosha/JavaPractice