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 Лукьяненко Д.В.

Принял Степанов П.В.

Практическая работа выполнена «___»_____2021 г.

«_____» «___»_____2021 г.
Отметка о выполнении

Москва – 2021 г.

СОДЕРЖАНИЕ

Цель работы

Цель данной практической работы – Изучить различные виды 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.

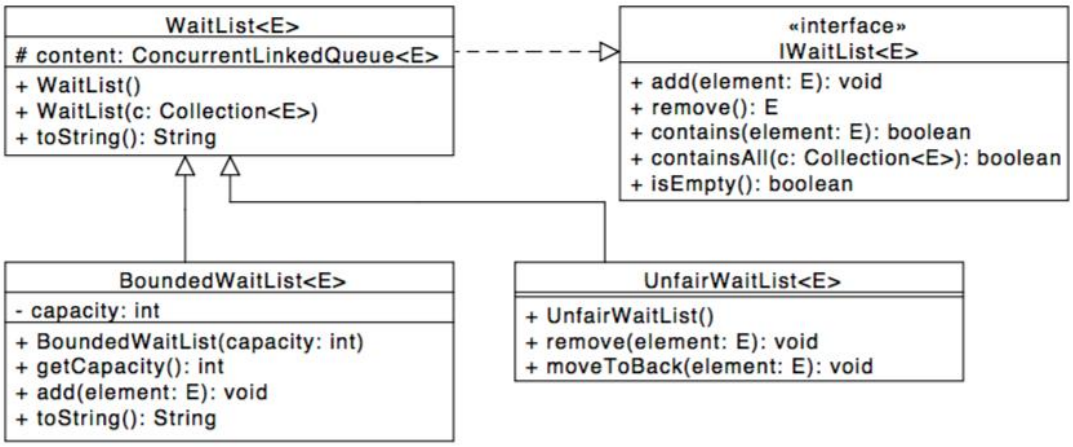

Рисунок 1 – Заданная UML Диаграмма

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.

Репозиторий

Ссылка: https://github.com/neluckoff/mirea-java-lessons/tree/master/src/ru/luckoff/mirea/practice_8

Выполнение работы

Первым делом я стал рассматривать диаграмму, которую нам дали в задании. После того, как я понял примерный смысл программы и того, что от меня хотят, я приступил создавать эти классы и заполнять их “пустыми” методами.

Затем, после полного переписывания диаграммы в таблицу у меня началась импровизация, ведь в самом задании нам сказали модифицировать код, чтобы полностью реализовать схему UML, что я и старался сделать.

Что получилось по итогу Вы можете посмотреть ниже на рисунке 2.

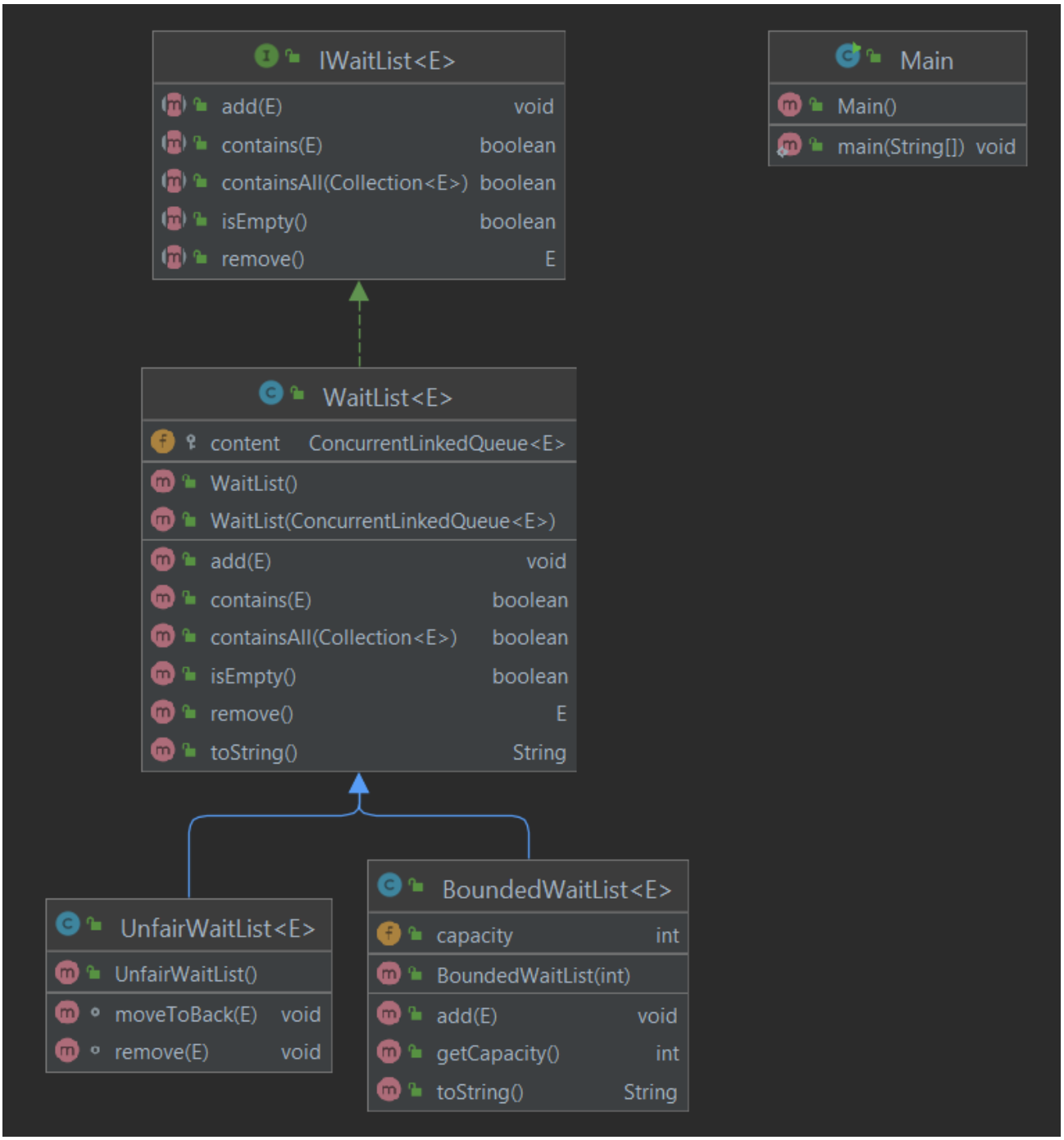

Рисунок 2 – Полученная UML Диаграмма

Код выполненной работы

Здесь в нескольких скриншотах можно увидеть как выглядит код выполненного задания и результат его работы.

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

Рисунок 3 – Интерфейс IWaitList

```
public class BoundedWaitList<E> extends WaitList<E> {
    public int capacity;

    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 +
                '}';
    }
}
```

Рисунок 4 – Класс BoundedWaitList

```java
public class UnfairWaitList<E> extends WaitList<E> {
    public UnfairWaitList() {
    }

    void remove(E element) {
        content.remove(element);
    }

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

Рисунок 5 – Класс UnfairWaitList

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

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

    @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 6 – Класс WaitList

```java
public class Main {
    public static void main(String[] args) {
        WaitList<Integer> list1 = new WaitList<>();
        System.out.println("add");
        for (int i = 0; i <= 5; i++) {
            list1.add(i);
        }
        System.out.println(list1);

        System.out.println("\nisEmpty");
        System.out.println(list1.isEmpty());


    }
}
```

Рисунок 7 – Класс Main

```
add
WaitList{content=[0, 1, 2, 3, 4, 5]}

isEmpty
false

Process finished with exit code 0
```

Рисунок 8 – Результат запуска программы

Вывод

В результате выполнения данной практической работы я изучил на практике приемы работы с различными видами списков ожидания и научился применять их в программе.