TP7 : Faites la queue

Exercice 1 - Fifo

1. Pour détecter si la file est pleine ou vide, on regarde la taille de la file.

2. ```
   public class Fifo<E> implements Iterable<E> {
       private final E[] array;
       private int head;
       private int tail;
       private int size;

    public Fifo(int capacity) {
       if (capacity < 1) {
           throw new IllegalArgumentException("capacity < 1");
       }
       @SuppressWarnings("unchecked")
       var array = (E[]) new Object[capacity];
       this.array = array;
       }
   }
   ```

3. Pour detecter si la file est pleine on regarde si le nombre d'élément est égal à la capacité de la file. Si la file est pleine, on throw un **IllegalStateException()**.

   ```
   public void offer(E element) {
       Objects.requireNonNull(element);
       if (size == array.length) {
           throw new IllegalStateException("No place");
       }
       array[tail] = element;
       tail = (tail + 1) % array.length;
       size++;
   }
   ```

4. Si la file est vide on renvoie un **IlegalStateException()**.

   ```
   public E poll() {
        if (size == 0) {
            throw new IllegalStateException("Any element found");
        }
        var element = array[head];
        array[head] = null;
        head = (head + 1) % array.length;
        size--;
        return element;
    }
   ```

5. ```
   @Override
   public String toString() {
     var stringJoiner = new StringJoiner(", ", "[", "]");
   ```

```
   var h = head;

   for (var i = 0; i < size; i++) {
      var element = array[h];
      h = (h + 1) % array.length;
      stringJoiner.add(element.toString());
   }
   return stringJoiner.toString();
}
```

6. Un **memory leak** correspond à une fuite de mémoire.

7. ```
   public int size() {
        return size;
   }

   public boolean isEmpty() {
        return size == 0;
   }
   ```

8. Un **itérateur** possède deux méthodes **hasNext()** qui renvoie un boolean pour savoir si il y a un prochain élément dans l'iterateur et une méthode **next()** qui renvoie le prochain élément et passe à l'élément suivant si il en existe un.

9. ```
   public Iterator<E> iterator() {
        return new Iterator<>() {
            private int h = head;
            private int i = 0;
            @Override
            public boolean hasNext() {
                return i < size;
            }
            @Override
            public E next() {
                if (!hasNext()) {
                    throw new NoSuchElementException("it has no next");
                }
                var element = array[h];
                h = (h + 1) % array.length;
                i++;
                return element;
            }
        };
   }
   ```

10. L'interface **Iterable** permet d'utiliser les méthodes tel que **forEach**.

Exercice 2 - ResizeableFifo

1. ```
   private void resize() {
        @SuppressWarnings("unchecked")
        var newArray = (E[]) new Object[array.length * 2];
   ```
   ```

```
    System.arraycopy(array, head, newArray, 0, size - head);
    System.arraycopy(array, 0, newArray, size - head, tail);
    head = 0;
    tail = size;
    array = newArray;
}
```

Steve Chen