

```

Tests failed: 2, passed: 30 of 32 tests - 975 ms
"C:\Program Files\Java\jdk-19\bin\java.exe" ...
Java HotSpot(TM) 64-Bit Server VM warning: Options -Xverify:none and -noverify were deprecated in JDK 13 and will likely be removed in a future release.
Test iterator and wraparound behavior with 200 random values with seed = 69 and fixed array size = 20
=====
Hint: This test is likely failing because you don't have correct wrap-around behavior.
Remember that everything should advance; so, it's possible that your start wraps around.

org.opentest4j.AssertionFailedError: Reference and implemented queues are not equal ==> iterable contents differ at index [0], expected: <1695069013> but was: <-1182032251>
  at edu.caltech.cs2.datastructures.CircularArrayFixedSizeQueueTests$ImplementationTests.testWrapAround(CircularArrayFixedSizeQueueTests.java:210) <81 internal lines>
  at java.base/java.util.ArrayList.forEach(ArrayList.java:1511) <9 internal lines>
  at java.base/java.util.ArrayList.forEach(ArrayList.java:1511) <9 internal lines>
  at java.base/java.util.ArrayList.forEach(ArrayList.java:1511) <23 internal lines>

Test iterator and wraparound behavior with 300 random values with seed = 21 and fixed array size = 200
=====
Hint: This test is likely failing because you don't have correct wrap-around behavior.
Remember that everything should advance; so, it's possible that your start wraps around.

org.opentest4j.AssertionFailedError: Reference and implemented queues are not equal ==> iterable contents differ at index [0], expected: <-549383847> but was: <-1151252339>
  at edu.caltech.cs2.datastructures.CircularArrayFixedSizeQueueTests$ImplementationTests.testWrapAround(CircularArrayFixedSizeQueueTests.java:210) <81 internal lines>

```

```

private class ArraySetIterator implements Iterator<E> {
    4 usages
    private int idx;

    1 usage  👤 Julian Millan
    public ArraySetIterator() { this.idx = 0; }

    👤 Julian Millan
    @Override
    public boolean hasNext() { return this.idx < capacity; }

    👤 Julian Millan
    @Override
    public E next() {
        E toReturn = data[this.idx];
        this.idx++;
        return toReturn;
    }
}

```

```

public boolean enqueue(E e) {
    if (!isFull()) {
        if (this.positions[0] == -1) { //empty queue
            this.positions[0] = 0; //set front and back to same positions
            this.positions[1] = 0;
            data[0] = e;
        } else if (this.positions[1] != this.data.length - 1) {
            this.positions[1] += 1;
            data[this.positions[1]] = e;
        } else {
            this.positions[1] = 0;
            data[0] = e;
        }

        return true;
    }
    return false;
}

```

Julian Millan +1

@Override

```

public E dequeue() {
    E elem = null;
    if (this.positions[0] == -1){
        return null;
    }
    elem = data[this.positions[0]];
    this.data[this.positions[0]] = null;
    if (this.positions[0] == this.positions[1]){

```