Welcome to CS2030S Lab 4!

17 September 2021 [16A]

Please login to your pe node once it's 4pm.

PA1

- The practical assessment will last 100 minutes
- You will be coding in the PE node (the server that you SSH into every week)
- Important: Please **plan** and **think** about your program's design before writing any code!
- Full marks if you finish the PE within the assessment period **and** pass all hidden test cases on CodeCrunch

PA1

- Moderation period of one week to modify your code for correctness; the less the modifications, the higher the final moderated score
- Style is **not** graded for the PA
- Topic coverage: PA1 will cover topics tested until lab 4 (today's lab)

PA1 - Invigilation

https://www.nus.edu.sg/celc/programmes/files/Zoom%20Invigilation.pdf

For students who are taking the practical assessment online, you may refer to the link above on how to set up your Zoom environment (ignore the Examplify portion)

Project

- The project has been released on CodeCrunch
- The entire project is worth 10% of your grade
- There are currently 2 levels for you to complete
- Same as previous labs, submit your files onto CodeCrunch!
- Style is graded! You will need to write Javadocs for public methods
- More to come :)

Submission Statistics

Labs	Submitted	Not Submitted	A
1	14	4	14
2	16	2	13
3	16	2	7

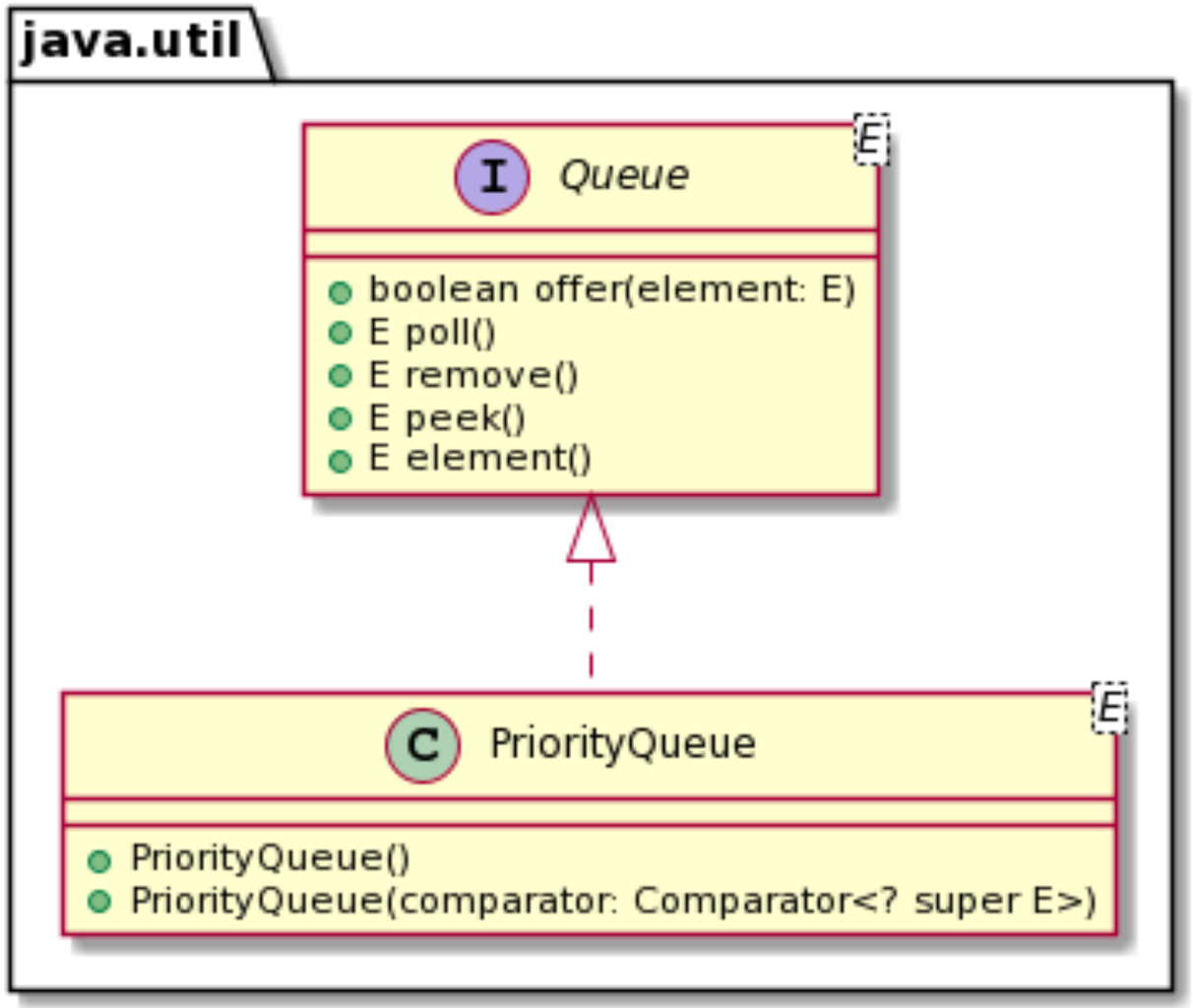
Mini-Lecture on Priority Queues, `Comparable` and `Comparator` interfaces

Priority Queues

In a priority queue (PQ), the element with the **highest priority*** is removed first.

- A **queue** is a first-in, first-out data structure.
- Elements are appended to the end of the queue and removed from the head of the queue.
- In a PQ, elements are assigned priorities. When accessing the elements, the element with the highest priority is removed first.

`java.util.PriorityQueue<E>`



- `offer`: inserts an element into the queue.
- `poll`: Retrieves and removes the head of this queue, or `null` if this queue is empty.
- `remove`: Retrieves and removes the head of this queue, or throws an exception if this queue is empty.
- `peek`: Retrieves, but does not remove, the head of this queue, returning `null` if this queue is empty.
- `element`: Retrieves, but does not remove, the head of this queue, throws an exception if this queue is empty.

How to define priority?

The `Comparable<T>` Interface

```
public interface Comparable<T> {
    public int compareTo(T o);
}
```

Many classes in the Java library such as `Integer`, `Double` and `String` already implement it to define a natural order.

```
jshell> new Integer(3).compareTo(new Integer(5));
$1 ===> -1
jshell> "ABC".compareTo("ABC"); // lexicographical order
$2 ===> 0
jshell> new Double(2.1).compareTo(new Double(2));
$3 ===> 1
```

```java
// Compare ages, then names if ages are the same
class Person implements Comparable<Person> {
    int age;
    String name;

    // Other code here

    @Override
    public int compareTo(Person other) {
        if (this.age != other.age) {
            return this.age - other.age;
        }
        return this.name.compareTo(other.name);
    }
}
```

In the above example, we compare names if the ages are the same

java.util.Comparator<T>

`Comparator` can be used to compare the objects of a class that **doesn't implement** `Comparable` or **define a new criteria for comparing objects**.

```
import java.util.Comparator;

class AgeComparator implements Comparator<Person> {
    // Sort by age in ascending order
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

```java
import java.util.PriorityQueue;
import java.util.Comparator;

class PriorityQueueDemo {
    public static void main(String[] args) {
        PriorityQueue<String> queue1 = new PriorityQueue<>();
        queue1.offer("Oklahoma");
        queue1.offer("Indiana");
        queue1.offer("Georgia");
        queue1.offer("Texas");
        System.out.println("PQ using Comparable:");
        while (!queue1.isEmpty()) {
            System.out.print(queue1.remove() + " ");
        }

        PriorityQueue<String> queue2 = new PriorityQueue<>(Comparator.reverseOrder());
        queue2.offer("Oklahoma");
        queue2.offer("Indiana");
        queue2.offer("Georgia");
        queue2.offer("Texas");
        System.out.println("\nPQ using Comparator:");
        while (!queue2.isEmpty()) {
            System.out.print(queue2.remove() + " ");
        }
    }
}
```

```
PQ using Comparable:
Georgia Indiana Oklahoma Texas
PQ using Comparator:
Texas Oklahoma Indiana Georgia
```

Mid-Semester Teaching Feedback Survey

- A survey has been released on LumiNUS for you to provide teaching feedback about the module so far
- Please fill it in so that we can continue to improve on the teaching materials

elinks index.html