Graphical user interface, text

Description automatically generated with medium confidence

Or compare object attributes

Text

Description automatically generated

Eg child class of Driver  
Text

Description automatically generatedGraphical user interface, text

Description automatically generated

import java.util.Comparator;

import java.util.PriorityQueue;

/\*\*

 \* An immutable and unbounded priority queue based on a priority heap.

 \* The elements of the priority queue are ordered according to their

 \* a {@link Comparator} provided at queue construction time. A priority

 \* queue does not permit {@code null} elements.

 \* <p>The <em>head</em> of this queue is the <em>least</em> element

 \* with respect to the specified ordering.  If multiple elements are

 \* tied for least value, the head is one of those elements -- ties are

 \* broken arbitrarily.  The queue retrieval operation {@code poll},

 \* access the element at the head of the queue.

 \*

 \* <p>A priority queue is unbounded, but has an internal

 \* <i>capacity</i> governing the size of an array used to store the

 \* elements on the queue.  It is always at least as large as the queue

 \* size.  As elements are added to a priority queue, its capacity

 \* grows automatically.  The details of the growth policy are not specified

 \* @param <E> the type of elements held in this queue

 \*/

public class PQ<E> {

    private final PriorityQueue<E> pq;

    /\*\*

     \* Creates a {@code PQ} with the default initial capacity and

     \* whose elements are ordered according to the specified comparator.

     \*

     \* @param  comparator the comparator that will be used to order this

     \*         priority queue.

     \*/

    public PQ(Comparator<? super E> comparator) {

        this.pq = new PriorityQueue<E>(comparator);

    }

    private PQ(PQ<E> source) {

        this.pq = new PriorityQueue<E>(source.pq);

    }

    /\*\*

     \* Returns {@code true} if this collection contains no elements.

     \*

     \* @return {@code true} if this collection contains no elements

     \*/

    public boolean isEmpty() {

        return this.pq.isEmpty();

    }

    /\*\*

     \* Inserts the specified element into this priority queue.

     \*

     \* @param  element the element to be added to the priority queue

     \* @return the priority queue with the element added

     \*/

    public PQ<E> add(E element) {

        PQ<E> copy = new PQ<E>(this);

        copy.pq.add(element);

        return copy;

    }

    /\*\*

     \* Retrieves and removes the head of this queue,

     \* or returns {@code null} if this queue is empty.

     \*

     \* @return the head of this queue, and the resulting priority queue

     \*         after removal as a {@code Pair}

     \*/

    public Pair<E, PQ<E>> poll() { // may need Pair class

        PQ<E> copy = new PQ<E>(this);

        E t = copy.pq.poll();

        return new Pair<E,PQ<E>>(t, copy);

    }

    /\*\*

     \* Returns a string representation of this priority queue.  The string

     \* representation consists of a list of elements in the order they are

     \* returned by its iterator, enclosed in square brackets ({@code "[]"}).

     \* Adjacent elements are separated by the characters {@code ", "} (comma and space).

     \* Elements are converted to strings as by {@link String#valueOf(Object)}.

     \*

     \* @return a string representation of this list

     \*/

    @Override

    public String toString() {

        return this.pq.toString();

    }

}