Project Description

Title:
Implementing a Priority Queue Using Ordered Arrays

Overview:
Students will implement a priority queue in which elements are stored in an array that is kept sorted at all times. The element with the highest priority (i.e. the smallest element when using natural ordering) is always located at the beginning of the array. This project reinforces concepts of array manipulation, algorithmic efficiency (trade-offs between insertion cost and fast removal/peek), and thorough testing of data structure methods.

Objectives:

- Understand how to maintain a sorted array during insertion.
- Learn the trade-offs of using an ordered array versus an unordered array for implementing a priority queue.
- Implement and test the following methods:
  - **add(E e)** / **offer(E e):** Insert an element while maintaining order.
  - **clear():** Remove all elements from the priority queue.
  - **contains(Object o):** Check whether a specific element exists.
  - **iterator():** Provide an iterator over the elements.
  - **peek():** Retrieve, but not remove, the highest-priority element.
  - **poll():** Retrieve and remove the highest-priority element.
  - **remove(Object o):** Remove a given element from the queue.
  - **size():** Return the number of elements.
- Develop a main method (or main block) using if/else conditions to test each method.

Requirements:

- Use an ordered array (or Python list) as the internal data storage.
- For the Java solution, implement dynamic array resizing.
- Ensure that all operations behave as expected, returning appropriate values when the queue is empty.
- Thoroughly test all methods using conditional logic to verify their correct behavior.