Project Description

Title:
Implementing a Priority Queue Using Unordered Arrays

Overview:
In this project, students will implement a priority queue data structure using an unordered array. A priority queue stores elements so that the element with the highest priority (the smallest element when using natural ordering) is always the one that can be accessed or removed first. Because the underlying array is not kept sorted, methods like `peek()` and `poll()` must scan the entire array to determine the highest-priority element.

Objectives:

- Gain experience implementing dynamic data structures using arrays.
- Practice working with generics in Java and lists in Python.
- Implement common priority queue operations:
  - **add(E e)** / **offer(E e):** Insert elements.
  - **clear():** Remove all elements.
  - **contains(Object o):** Check if an element exists.
  - **iterator():** Return an iterator for traversing the queue.
  - **peek():** Retrieve (without removing) the highest-priority element.
  - **poll():** Retrieve and remove the highest-priority element.
  - **remove(Object o):** Remove a specified element.
  - **size():** Return the number of elements in the queue.
- Write a main method (or main block) that uses if/else conditions to test each method.

Requirements:

- Use an unordered array (or Python list) for the internal storage.
- Include dynamic resizing in the Java implementation if the array becomes full.
- Ensure all methods behave as specified (for example, returning `null` or `None` when the queue is empty for `peek()` and `poll()`).
- Test the functionality thoroughly with conditional checks in the main method or main block.