Project Description

Title:
Implementing a Priority Queue Using Binary Heaps

Overview:
Students will design and implement a priority queue where the underlying data structure is a binary heap. The binary heap is stored in an array and supports two main operations:

- **Insertion (add/offer):** When a new element is added, it is placed at the end of the array and "bubbled up" until the heap property is restored.
- **Removal (poll):** The highest-priority element (the smallest in a min-heap) is removed. The last element in the array is moved to the root, then "bubbled down" to restore the heap property.

Objectives:

- Understand the binary heap structure and its properties.
- Learn how to implement dynamic array resizing (in Java) for a binary heap.
- Implement core priority queue methods:
  - **add(E e)/offer(E e):** Insert elements into the binary heap.
  - **peek():** Retrieve (without removing) the highest-priority element.
  - **poll():** Remove and return the highest-priority element.
  - **remove(Object o):** Remove a specific element.
  - **clear():** Remove all elements.
  - **contains(Object o):** Check if an element exists in the heap.
  - **size():** Return the number of elements.
  - **iterator():** Provide an iterator over the elements.
- Test each method using a main method (or main block) with if/else conditions to verify correct behavior.

Requirements:

- Implement the binary heap using an array (or Python list) and include helper methods for "bubbling up" and "bubbling down."
- Ensure that all methods behave as specified, particularly when the priority queue is empty.
- Thoroughly test all functionality using conditional statements.