Project Overview: Circular Linked List

Objective:
Implement a circular linked list from scratch using object-oriented programming (OOP) principles. In a circular linked list, the last node's pointer (or reference) points back to the first node, creating a continuous loop. This project is designed to help both Java and Python students understand class design, encapsulation, and method implementation while also highlighting the differences in language syntax and behavior.

Key OOP Concepts Demonstrated:

- **Classes and Objects:**
  Two primary classes are used—a Node class to represent each element and a CircularLinkedList class to manage the list.
- **Encapsulation:**
  Internal attributes (head, tail, size) are hidden from direct external access. Interaction is provided via public methods.
- **Abstraction:**
  Users interact only with the list's public interface (e.g., add, remove, get), without needing to understand the underlying circular structure.
- **Modularity:**
  The project separates node functionality from list management, making it easier to maintain and extend.

Methods and Their Signatures:

1. **add(String s) / add(s):**
   - **Purpose:** Appends an element to the end of the list.
   - **Arguments:**
     `s` – the element to add (a String in Java; any type in Python).
   - **Return Type:**
     `void` (Java), `None` (Python)
2. **addFirst(String s) / add_first(s):**
   - **Purpose:** Inserts an element at the beginning of the list.
   - **Arguments:**
     `s` – the element to insert.
   - **Return Type:**
     `void / None`

3. **contains(String s) / contains(s):**
    - **Purpose:** Checks if the list contains the specified element.
    - **Arguments:**
      `s` – the element to search for.
    - **Return Type:**
      `boolean` (Java), `bool` (Python)
4. **getFirst() / get_first():**
    - **Purpose:** Retrieves the first element.
    - **Return Type:**
      `String` / element type (or `null` / `None` if empty)
5. **getLast() / get_last():**
    - **Purpose:** Retrieves the last element.
    - **Return Type:**
      `String` / element type
6. **size() / size():**
    - **Purpose:** Returns the number of elements.
    - **Return Type:**
      `int`
7. **remove() / remove():**
    - **Purpose:** Removes and returns the first element.
    - **Return Type:**
      `String` / element type
8. **removeLast() / remove_last():**
    - **Purpose:** Removes and returns the last element.
    - **Return Type:**
      `String` / element type
9. **get(int index) / get(index):**
    - **Purpose:** Retrieves the element at the specified index.
    - **Return Type:**
      `String` / element type

10. **clear() / clear():**
    - **Purpose:** Removes all elements.
    - **Return Type:**
      `void / None`
11. **toString() / str():**
    - **Purpose:** Returns a string representation of the list. For non-empty lists, the format is similar to `[elem1]<->[elem2]<->[elem3]`. If empty, returns a message like `"CircularLinkedList is empty"`.
    - **Return Type:**
      `String / str`
12. **isCircular() / is_circular():**
    - **Purpose:** Verifies the circular nature by checking that the tail's next pointer references the head.
    - **Return Type:**
      `boolean` (Java), `bool` (Python)