```
import java.util.Arrays;

// Custom exception for handling index out-of-bounds
class IndexOutOfBoundsException extends RuntimeException {
  public IndexOutOfBoundsException(String message) {
    super(message);
  }
}

// Array class with various array operations
class CustomArray {
  private int[] array;
  private int size;

  public CustomArray() {
    this.array = new int[10]; // Initial size of the array
    this.size = 0;
  }

  // Method to insert an element at the end of the array
  public void insert(int element) {
    ensureCapacity();
    array[size++] = element;
  }

  // Method to delete an element at a given index
  public void delete(int index) {
    if (index < 0 || index >= size) {
      throw new IndexOutOfBoundsException("Index out of bounds");
    }
    System.arraycopy(array, index + 1, array, index, size - index - 1);
    size--;
  }

  // Method to search for an element in the array using linear search
  public int linearSearch(int target) {
    for (int i = 0; i < size; i++) {
      if (array[i] == target) {
        return i; // Return the index if found
      }
    }
    return -1; // Return -1 if not found
  }

  // Method to search for an element in the array using binary search
  public int binarySearch(int target) {
    Arrays.sort(array, 0, size); // Ensure the array is sorted
    return Arrays.binarySearch(array, 0, size, target);
  }

  // Method to search for an element in the array using interpolation search
  public int interpolationSearch(int target) {
    Arrays.sort(array, 0, size); // Ensure the array is sorted
```

```
    int low = 0;
    int high = size - 1;

    while (low <= high && target >= array[low] && target <= array[high]) {
      int pos = low + ((target - array[low]) * (high - low)) / (array[high] - array[low]);

      if (array[pos] == target) {
        return pos; // Return the index if found
      }

      if (array[pos] < target) {
        low = pos + 1;
      } else {
        high = pos - 1;
      }
    }

    return -1; // Return -1 if not found
  }

  // Method to update an element at a given index
  public void update(int index, int newValue) {
    if (index < 0 || index >= size) {
      throw new IndexOutOfBoundsException("Index out of bounds");
    }
    array[index] = newValue;
  }

  // Method to print the elements of the array
  public void printArray() {
    System.out.print("Array: [");
    for (int i = 0; i < size; i++) {
      System.out.print(array[i]);
      if (i < size - 1) {
        System.out.print(", ");
      }
    }
    System.out.println("]");
  }

  // Private method to ensure capacity for dynamic array resizing
  private void ensureCapacity() {
    if (size == array.length) {
      array = Arrays.copyOf(array, size * 2);
    }
  }
}

public class majorProject {
  public static void main(String[] args) {
    CustomArray customArray = new CustomArray();

    // Array operations
    customArray.insert(5);
    customArray.insert(10);
```

```
    customArray.insert(15);
    customArray.printArray();

    customArray.delete(1);
    customArray.printArray();

    customArray.update(0, 8);
    customArray.printArray();

    // Searching algorithms
    int linearSearchIndex = customArray.linearSearch(15);
    System.out.println("Linear Search Index: " + linearSearchIndex);

    int binarySearchIndex = customArray.binarySearch(8);
    System.out.println("Binary Search Index: " + binarySearchIndex);

    int interpolationSearchIndex = customArray.interpolationSearch(8);
    System.out.println("Interpolation Search Index: " + interpolationSearchIndex);
  }
}
```