Program 1: Array Traversal with a for loop

```
import java.util.Arrays;
public class ArrayTraversal {
    public static void main(String[] args) {

        // Declare and initialize an array of integers
        int[] numbers = {10, 20, 30, 40, 50};

        System.out.println("Printing elements of the array using a for loop:");

        // Loop through the array from the first element (index 0) to the last
        // 'i < numbers.length' ensures we don't go out of bounds
        for (int i = 0; i < numbers.length; i++) {
            // Access and print each element using its index 'i'
            System.out.println("Element at index " + i + ": " + numbers[i]);
        }

        // A shortcut to print the entire array's contents
        System.out.println("\nPrinting the entire array: " + Arrays.toString(numbers));
    }
}
```

Program 2: Array Traversal with a for-each loop (Enhanced for loop)

```
import java.util.Arrays;

public class EnhancedForLoop {

    public static void main(String[] args) {

        // Declare and initialize an array of strings
        String[] fruits = {"Apple", "Banana", "Orange", "Grape"};

        System.out.println("Printing fruits using an enhanced for loop:");

        // The for-each loop iterates over each element in the 'fruits' array
        // 'String fruit' declares a variable to hold the current element
        for (String fruit : fruits) {
            // 'fruit' now holds the value of the current element
            System.out.println("Fruit: " + fruit);
        }
        System.out.println("\nPrinting the entire array: " + Arrays.toString(fruits));
    }
}
```

Program 3: Using an Iterator for a List

```
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

public class ListIteratorExample {

    public static void main(String[] args) {

        // Create an ArrayList, which is a dynamic list that implements the List interface
        List<String> cities = new ArrayList<>();

        // Add elements to the list
        cities.add("New York");
        cities.add("London");
        cities.add("Tokyo");
        cities.add("Paris");

        System.out.println("Printing cities using an Iterator:");

        // Get an Iterator for the list. This allows us to traverse the collection.
        Iterator<String> iterator = cities.iterator();

        // The hasNext() method checks if there is another element available
        while (iterator.hasNext()) {
            // The next() method returns the next element in the iteration
            String city = iterator.next();
            System.out.println("City: " + city);
        }
    }
}
```

Program 4: Modifying an Array and finding the sum

```
import java.util.Arrays;

public class ArraySumAndModify {

    public static void main(String[] args) {

        // An array of double values
        double[] prices = {12.50, 9.99, 5.00, 15.75, 8.25};
        double totalSum = 0.0; // Initialize a variable to hold the sum

        System.out.println("Original prices: " + Arrays.toString(prices));

        // Iterate through the array to calculate the sum and modify elements
        for (int i = 0; i < prices.length; i++) {
            // Add the current element to the total sum
            totalSum += prices[i];

            // Let's add a 10% tax to each price
            prices[i] = prices[i] * 1.10;
        }

        System.out.println("\nCalculated total sum before tax: " + String.format("%.2f", totalSum));
        System.out.println("Prices after adding 10% tax: " + Arrays.toString(prices));
    }
}
```

Program 5: Using Iterator with the remove() method

```
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

public class IteratorRemoveExample {

    public static void main(String[] args) {

        List<Integer> numbers = new ArrayList<>();

        numbers.add(1);
        numbers.add(2);
        numbers.add(3); // This will be removed
        numbers.add(4);
        numbers.add(5); // This will be removed

        System.out.println("Original list: " + numbers);

        // Get an iterator for the list
        Iterator<Integer> iterator = numbers.iterator();

        // Iterate through the list
        while (iterator.hasNext()) {
            Integer currentNumber = iterator.next();

            // Check if the current number is odd
            if (currentNumber % 2 != 0) {
                // If it's odd, safely remove it from the list using the iterator's remove() method
                iterator.remove();
                System.out.println("Removed odd number: " + currentNumber);
```

```
            }
        }

        System.out.println("List after removing odd numbers: " + numbers);
    }
}
```