```
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.Iterator;
import java.util.List;
import java.util.ListIterator;
import java.util.Spliterator;
import java.util.function.Consumer;
import java.util.function.UnaryOperator;

public class ArrayListAllMethodsDemo {
    public static void main(String[] args) {

        ArrayList<String> fruits = new ArrayList<>();

        // 1. add(E e) - Adds element to the list
        fruits.add("Apple");
        fruits.add("Banana");
        fruits.add("Cherry");
        // Output: [Apple, Banana, Cherry]

        // 2. addAll(Collection<? extends E> c) - Adds all elements from another collection
        ArrayList<String> moreFruits = new ArrayList<>();
        moreFruits.add("Date");
        moreFruits.add("Elderberry");
        fruits.addAll(moreFruits);

        // Output: [Apple, Banana, Cherry, Date, Elderberry]

        // 3. clear() - Removes all elements
        fruits.clear();
        // Output: []

        // 4. contains(Object o) - Checks if list contains an element
        fruits.add("Apple");
        boolean containsApple = fruits.contains("Apple");
        // Output: true

        // 5. containsAll(Collection<?> c) - Checks if list contains all elements from another
collection
        fruits.addAll(moreFruits);
        boolean containsAll = fruits.containsAll(moreFruits);
        // Output: true

        // 6. isEmpty() - Checks if list is empty
        boolean isEmpty = fruits.isEmpty();
        // Output: false
```

```
// 7. iterator() - Returns an iterator
Iterator<String> iterator = fruits.iterator();
while (iterator.hasNext()) {
    String fruit = iterator.next();
}
// Output during iteration: Apple Date Elderberry

// 8. remove(Object o) - Removes the first occurrence of the specified element
fruits.remove("Apple");
// Output: [Date, Elderberry]

// 9. removeAll(Collection<?> c) - Removes all matching elements
fruits.removeAll(moreFruits);
// Output: []

// 10. retainAll(Collection<?> c) - Retains only matching elements
fruits.add("Apple");
fruits.add("Banana");
fruits.retainAll(moreFruits);
// Output: []

// 11. size() - Returns number of elements
int size = fruits.size();
// Output: 0

// 12. toArray() - Converts list to Object[]
Object[] fruitArray = fruits.toArray();
// Output: []

// 13. toArray(T[] a) - Converts to typed array
String[] fruitArray2 = new String[fruits.size()];
fruitArray2 = fruits.toArray(fruitArray2);
// Output: []

// 14. add(int index, E element) - Inserts element at specified index
fruits.add(0, "Apple");
// Output: [Apple]

// 15. addAll(int index, Collection<? extends E> c) - Inserts collection at index
fruits.addAll(1, moreFruits);
// Output: [Apple, Date, Elderberry]

// 16. get(int index) - Gets element at index
String elementAt1 = fruits.get(1);
// Output: Date

// 17. indexOf(Object o) - Returns index of first occurrence
int index = fruits.indexOf("Date");
```

```
// Output: 1

// 18. lastIndexOf(Object o) - Returns last index of occurrence
int lastIndex = fruits.lastIndexOf("Date");
// Output: 1

// 19. listIterator() - Returns ListIterator
ListIterator<String> listIterator = fruits.listIterator();
while (listIterator.hasNext()) {
    String value = listIterator.next();
}

// 20. listIterator(int index) - Iterator starting at index
ListIterator<String> listIterator2 = fruits.listIterator(1);
while (listIterator2.hasNext()) {
    String value = listIterator2.next();
}

// 21. remove(int index) - Removes element at index
fruits.remove(0);
// Output: [Date, Elderberry]

// 22. set(int index, E element) - Updates element at index
fruits.set(0, "Apple");
// Output: [Apple, Elderberry]

// 23. subList(int fromIndex, int toIndex) - Returns view of range
List<String> subList = fruits.subList(0, 2);
// Output: [Apple, Elderberry]

// 24. ensureCapacity(int minCapacity) - Ensures internal array can hold elements
fruits.ensureCapacity(10);

// 25. trimToSize() - Trims capacity to current size
fruits.trimToSize();

// 26. replaceAll(UnaryOperator<E> operator) - Replaces each element using function
fruits.replaceAll(String::toUpperCase);
// Output: [APPLE, ELDERBERRY]

// 27. sort(Comparator<? super E> c) - Sorts list
fruits.sort(Comparator.naturalOrder());
// Output: [APPLE, ELDERBERRY]

// 28. forEach(Consumer<? super E> action) - Loops through list using lambda
fruits.forEach(f -> {
    // Output: APPLE ELDERBERRY
});
```

```
        // 29. spliterator() - Returns Spliterator for parallel processing
        Spliterator<String> spliterator = fruits.spliterator();
        spliterator.forEachRemaining(f -> {
            // Output: APPLE ELDERBERRY
        });

        // 30. equals(Object o) - Checks equality with another list
        ArrayList<String> fruitsCopy = new ArrayList<>(fruits);
        boolean isEqual = fruits.equals(fruitsCopy);
        // Output: true

        // 31. hashCode() - Returns hash code for list
        int hashCode = fruits.hashCode();

        // 32. toString() - Returns string representation
        String stringOutput = fruits.toString();
        // Output: [APPLE, ELDERBERRY]

        // 33. clone() - Returns shallow copy
        ArrayList<String> clonedFruits = (ArrayList<String>) fruits.clone();
        // Output: [APPLE, ELDERBERRY]

        // 34. removeRange(int fromIndex, int toIndex) - Removes elements in range (Protected
method)
        class CustomArrayList<E> extends ArrayList<E> {
            public void removeRange(int fromIndex, int toIndex) {
                super.removeRange(fromIndex, toIndex);
            }
        }

        CustomArrayList<String> customFruits = new CustomArrayList<>();
        customFruits.addAll(fruits);
        customFruits.removeRange(0, 1);
        // Output: [ELDERBERRY]
    }
}
```