import java.util.ArrayList;  
  
public class Main {  
 public static void main(String[] args) {  
 ArrayList<String> list = new ArrayList<>(); //created new ArrayList with type String  
 list.add("First line"); //added first object (line) in our ArrayList  
 System.*out*.println(list); //show all ArrayList console: [First line]  
  
 list.add("Second line");  
 System.*out*.println(list); // console: [First line, Second line]  
  
 list.add(0, "new line"); // добавили новый объект по индексу; в данном случае на первое место  
 // весь список сдвинулся, уступив указанное место новому объекту  
 System.*out*.println(list); // console: [new line, First line, Second line]  
  
 // добавление всех элементов коллекции в список в порядке их расположения в collection  
 ArrayList<String> secondList = new ArrayList<>();  
 secondList.addAll(list); //копируем весь list в secondList  
 System.*out*.println(secondList); //console: [new line, First line, Second line]  
 ArrayList<String> thirdList = new ArrayList<>();  
 thirdList.add("ONE");  
 thirdList.add("TWO");  
 thirdList.add("THREE");  
 secondList.addAll(1, thirdList); //добавляем коллекцию в уже имеющуюся коллекцию по индексу  
 // таким образом хвост первой коллекции сдвинется в конец  
 System.*out*.println(secondList); //console: [new line, ONE, TWO, THREE, First line, Second line]  
  
 // .clear() - удаление всех элементов из списка  
 // .clone() - возвращает объект-копию массива  
  
 //метод clone() возвращает Object, так что после его вызова потребуется сделать приведение к необходимому классу  
 ArrayList<String> copyOfThirdList = (ArrayList<String>) thirdList.clone();  
 System.*out*.println(copyOfThirdList); //console: [ONE, TWO, THREE]  
  
 //.contains(Object o)  
 // проверка на наличие объекта в списке. Возвращает boolean-значение  
 System.*out*.println(list.contains("Fourth line")); //console: false  
 System.*out*.println(list.contains("Second line")); //console: true  
  
 //.remove(int index)  
 //удаление элемента в указанной позиции массива. Оставшиеся в хвосте элементы сдвигаются на один, соответственно.  
  
 secondList.remove(0);  
 System.*out*.println(secondList); //console: [ONE, TWO, THREE, First line, Second line]  
  
 //.remove(Object o)  
 // Метод удаляет из списка переданный элемент o.  
 // Если элемент присутствует в списке, он удаляется, а все элементы смещаются влево.  
 // Если элемент существует в списке и успешно удален, метод возвращает true, в обратном случае — false.  
  
 System.*out*.println(secondList.remove("THREE")); //console: true  
 System.*out*.println(secondList); //console: [ONE, TWO, First line, Second line]  
  
 //.removeAll(Collection<?> c)  
 // Если необходимо удалить несколько элементов, не стоит делать это в цикле по условию:  
 // гораздо удобнее и безопаснее воспользоваться методом removeAll().  
 // Он принимает коллекцию элементов, которая будет удалена из списка.  
 // Если операция прошла успешно - true, иначе - false.  
  
 list.addAll(thirdList);  
 System.*out*.println("Получился новый список под названием list: " + list);  
 // console: Получился новый список под названием list: [new line, First line, Second line, ONE, TWO, THREE]  
 System.*out*.println("Удаление коллекции thirdList из list: " + list.removeAll(thirdList));  
 // console: Удаление коллекции thirdList из list: true  
 System.*out*.println("Получился новый список list: " + list);  
 // console: Получился новый список list: [new line, First line, Second line]  
  
 //.set(index i, Element e)  
 // замена элемента в указанной по индексу позиции  
  
 //.size()  
 // лучший способ узнать размер массива  
  
 //.ensureCapacity(int minCapacity)  
 //увеличивает размер массива, чтобы в него поместилось нужное кол. элементов.  
 //т.е. если мы заранее знаем, что массив будет содержать большое кол. элементов, то можем задать сразу нужный объем  
  
 //Этот метод полезен, когда возникает потребность вместить большое количество элементов в несколько итераций.  
 // Например, при создании списка емкость его внутреннего массива — 10. При загрузке данных по сети  
 // они обрабатываются асинхронно порциями, и результаты помещаются в массив.  
 // Если ожидается доставка 10 000 элементов, может быть неэффективно просто добавлять эти данные каждый раз:  
 // достаточно будет в начале обработки вызвать метод ensureCapaciry(10000) и записывать туда данные  
 // по мере необходимости.  
  
 // secondList.ensureCapacity(100); - пример записи  
  
 //.get(int index)  
 //возвращает элемент, который расположен в указанной позиции списка.  
  
 //.indexOf(Object o)  
 //метод возвращает индекс первой встречи элемента в списке (в списке могут быть одинаковые элементы)  
 //если элемент не найден, результат: -1  
  
 //.lastIndexOf(Object o)  
 //принцип тот же, что и у .indexOf(), но выдает последнюю позицию встречи искомого элемента в массиве  
  
 //.isEmpty()  
 //возвращает true - если список пустой, false - если имеет объекты  
 //Если в списке содержатся только элементы null, метод вернет false.  
 // Иными словами, null элементы также учитываются этим методом.  
 }  
}