Write a static method `subsets3` that uses recursive backtracking to find every possible subset of exactly size 3 of a given list. Your method should accept a `List` of strings as its parameter and print every subset of that could be created from 3 elements from the list, one per line. For example, suppose a variable called `list` stores the following elements:

```
[Janet, Robert, Morgan, Char]
```

The call of `subsets3(list);` would produce output such as the following:

```
[Janet, Robert, Morgan]
[Janet, Robert, Char]
[Janet, Morgan, Char]
[Robert, Morgan, Char]
```

The order in which you show the subsets does not matter, and the order of the elements of each subset also does not matter. The key thing is that your method should produce the correct overall set of subsets as its output. You may assume that the list passed to your method is not `null` and that the list contains no duplicates. In your solution, you should not recurse unnecessarily, as this would be inefficient for large lists.

As a hint, Section 14 handout had a problem called `subsets` that asked you to print all possible subsets of a list of any size. That is, all subsets containing 0 or more elements from the list. The solution for subset is as follows

```
public static void subsets(List<String> elements) {
    List<String> chosen = new ArrayList<String>();
    subsets(elements, chosen);
}

private static void subsets(List<String> elements, List<String> chosen) {
    if (elements.isEmpty()) {
        System.out.println(chosen);
    } else {
        String first = elements.remove(0);

        chosen.add(first);
        subsets(elements, chosen);
        chosen.remove(chosen.size() - 1);
        subsets(elements, chosen);

        elements.add(0, first);
    }
}
```

Solution

```
public static void subsets3(List<String> elements) {
    subset3(elements, new ArrayList<String>());
}

public static void subsets3(List<String> elements, List<String> chosen) {
    if (chosen.size() == 3) {
        System.out.println(chosen);
    } else if (!elements.isEmpty()) {
        String first = elements.remove(0);
        chosen.add(first);
        subsets3(elements, chosen);
        chosen.remove(chosen.size() - 1);
        subsets3(elements, chosen);
        elements.add(0, first);
    }
}
```