ArrayLists

Daniel Yee
CIS 2168

Purpose

In this exercise, you will learn to write static methods that use `Lists` and `Strings`. Each of these is around the difficulty of some exam questions. If you need to make a new List as part of the method, implement it using an `ArrayList`.

Static Generic Methods

Java makes writing generic methods look intimidating, but, it's not so bad. I'll walk you through how to do it by showing what we want to do, encountering an exception, and then showing you how to resolve the exception.

Suppose we want to write a method called `in` that checks if the `List` contains a specified item. We don't know, in advance, what type the item will be nor do we know the type of the items in the List. We also want this method to be able to be used in any kind of context. That means we want it to be generic. However, if we write

```
public static boolean in(List<E> list, E item) {

}
```

we get a compile time exception: "E cannot be resolved to a type" because Java doesn't know that you want to use E as the symbol for generics in this method. We can inform the compiler of our intention by including the generic symbol before the method's return type.

```
public static <E> Boolean in(List<E> list, E item) {

}
```

The big difference between a class that uses a generic, like `ArrayList`, and the static methods you write for this assignment is that the generic type only exists for the method.

Specifications

For each of the following, create a static method with the appropriate inputs and outputs. Call each of them in the main method.

Uniqueness

Write a method `unique()` which returns true if all the items in the supplied `List` are unique. All the items are unique if none of them are the same. Return false otherwise.

All Multiples

Write a method `allMultiples()` which returns a new `List` of integers that contains all of the integers in the `List` argument that are multiples of the given `int`. For example, if the `List` is [1; 25; 2; 5; 30; 19; 57; 2; 25] and 5 was provided, the new list should contain [25; 5; 30; 25].

All Strings of Size

Write a method named `allStringsOfSize()` which returns a new `List<String>` which contains all the Strings from the `List` argument that are `length` characters long. For example, if the inputs are ["I", "like", "to", "eat", "eat", "eat", "apples", "and", "bananas"] and 3, the new list is ["eat", "eat", "eat", "and"].

Permutations

Write a method `isPermutation()` which return true if the two `List` arguments are permutations of each other, otherwise it returns false. Two lists are permutations if they contain the same elements, including the same number of duplicates, but they don't have to contain the elements in the same order. For example, [1,2,4] and [2,1,4] are permutations of each other.

String Tokenization[1]

Write a method `tokenize()` returns a `List` of words from the input string. We assume that each word in the input string is separated by whitespace.[2] If our input String is "Hello, world!", then the output should be ["Hello,", "world!"]. For extra credit, sanitize the string - cleaning it up so that punctuation and other extraneous characters are removed. The final sanitized string would become ["Hello", "world"].

Remove All

Write a method `removeAll()` which removes all of the specified items from the `List` and returns the new List. For example, if the method is passed the List<Integer> [1, 4, 5, 6,5, 5, 2] and the Integer 5, the method removes all 5's from the List. The List then becomes [1, 4, 6, 2]. It should return nothing, since the changes the List it was provided.

---

[1] *Tokenization* is the process of demarcating and possibly classifying sections of a string of input characters. The resulting tokens are then passed on to some other form of processing. Tokenization is often used to parse inputs in lexical processing for natural and computer programming processing. Real-world lexical processing has much more complex rule than our simple method.

[2] Check the String class Javadoc to find a method that might help.

Rubric

90 points

Each properly working method is worth 10 points. An additional 5 points for each method that is using generics correctly.

10 points

Code is neat and properly indented.

5 points (extra credit)

See the String Tokenization section.

Presentation

Be prepared to explain how you solved `removeAll()` and that it works on the test cases, as well as one other method of our choice.