Collections (Part 1) exercises

The purpose of this exercise is to gain experience using the `List` object to better understand how it solves problems.

Evaluation criteria and functional requirements

- You must use the `List` object, not arrays.
- The project must not have any build errors.
- Unit tests pass as expected.
- Appropriate variable names and data types are used.
- Code is presented in a clean, organized format.

Getting started

1. Import the collections part one exercises project into IntelliJ.
2. Run all tests to see the results of your tests and which ones passed or failed.
3. Write your code in the provided classes in the project, starting with `array2List`.
4. Provide enough code to get a test passing.
5. Repeat until all tests are passing.

Tips and tricks

Here are some tips that may help you as you work on your exercises.

Read the problem description carefully

Before each method, there's a description of the problem you need to solve and examples with the expected output. Use these examples to figure out the values you need to write your code around. It may help to keep track of the state of variables on a piece of paper as you work on your exercises.

For example, in the comments before the `array2List` method, there's a section that includes the method name and the expected value that's returned for each method call. The following example shows that when the method is called with `["Apple", "Orange", "Banana"]`, it returns a `List<String>` that contains three elements: "Apple", "Orange", and "Banana."

```
array2List( {"Apple", "Orange", "Banana"} )  ->  ["Apple", "Orange", "Banana"]
```

Check test output if your tests are failing

If your tests fail, check the output of the test run. It provides helpful clues and information that could be valuable when troubleshooting. You can look at the unit tests to see what input is being tested.

You can also run the tests in debug mode when executing the tests. This allows you to set a "breakpoint", which stops the code at certain points in the editor. You can then look at the values of variables while the test

is executing, and can also see what code is being executed. Don't hesitate to use the debugging capabilities in IntelliJ to help resolve issues.

Don't linger too long on one problem

If you find yourself stuck on a problem more than fifteen minutes, move on to the next, and try again later. You may figure out the solution after working through another problem or two.

Read the documentation

As a developer, you'll find documentation for classes and libraries to be invaluable resources when completing your work. Reading and understanding documentation now will help you in the long run.

List ArrayList