Loops and arrays

This exercise consists of various problems to give you the opportunity to practice what you learned about loops and arrays.

Learning objectives

After completing this exercise, you'll understand:

- How to iterate, or loop through, arrays to solve complex problems.
- How to access items in an array with an index.
- How to create arrays.
- How to use loops to execute blocks of code multiple times.

Evaluation criteria and functional requirements

- 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.
- Loops and arrays are used appropriately.

Getting started

1. Import the loops and arrays exercises project into IntelliJ.
2. Run all tests to see the results of your tests and which ones passed or failed.
3. Provide enough code to get a test passing.
4. Repeat until all tests are passing.

Tips and tricks

- **Note: If you find yourself stuck on a problem for longer than fifteen minutes, move onto the next, and try again later.**
- Before each method, there is a description of the problem you need to solve, as well as examples with expected output. Use these examples to get an idea of the values you need to write your code around. For example, in the comments before the `sum2` method, there is a section that includes the method name, as well as the expected value that will be returned for each method call. The following example shows that when the method is called with `[1, 2, 3]`, it returns 3, when it's called with `[1, 1]`, it returns 2, and when it's called with `[1, 1, 1, 1]`, it returns 2:

```
sum2([1, 2, 3]) → 3
sum2([1, 1]) → 2
sum2([1, 1, 1, 1]) → 2
```

- When you are trying to solve these sorts of problems, it's helpful to keep track of the state of variables on a piece of paper as you are working through your code.

- The output of the test run can provide helpful clues as to why the tests are failing. Try reading the output of a failing test for more information that could be valuable when troubleshooting.
- You can also run the tests in debug mode when executing the tests. This allows you to set a "breakpoint", which halts 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 currently being executed. Don't hesitate to use the debugging capabilities in IntelliJ to help resolve issues.