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.
- Code uses appropriate variable names and data types.
- Code is in a clean, organized format.
- Loops and arrays used appropriately.

Getting started

1. Import the project into IntelliJ IDEA.
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

Read the problem description carefully

Before each method, there's a description of the problem you need to solve and examples with expected output. Use these examples to get an idea of 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 through your code.

For example, in the comments before the `getCalzoneSales` method, there is a section that includes the method name, as well and the expected value that's returned for each method call. The following example shows that when the method gets called with `[40, 30, 32, 40, 10]`, it returns 2, when it's called with `[30, 32, 10]`, it returns 0, and when it's called with `[]` (an empty array), it returns 0:

```
getCalzoneSales([40, 30, 32, 40, 10]) → 2
getCalzoneSales([30, 32, 10]) → 0
getCalzoneSales([]) → 0
```

Check test output if your tests are failing

If your tests fail, check the output of the test run. The test results display the input and expected output for each failing tests. In software development, unit tests can provide helpful clues and 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 stops the code at certain points in the editor. You can then look at the values of variables while the test runs, and step through code line-by-line as it runs. 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 for 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.