Collections - Part 2

The purpose of this exercise is reinforce understanding of the Map interface and how it's used.

Learning Objectives

After completing this exercise, students will understand:

- How to declare and create instances of Maps.
- How to iterate (loop through) key-value pairs in a Map.
- How to determine if a Map already contains a key.
- How to get and set a value with a key in a Map.
- How to remove items from a Map.
- How to navigate the Map API documentation.

Evaluation Criteria & Functional Requirements

- Maps should be used rather than Lists or arrays.
- The project must not have any build errors.
- Unit tests pass as expected.
- Appropriate variable names and data types are being used.
- Code is presented in a clean, organized format.

Getting Started

- Import the collections-part2-exercises project into Eclipse.
- Right-click on the project, and select the **Run As -> JUnit Test** menu option.
- Click on the **JUnit** tab to see the results of your tests and which passed/failed.
- Provide enough code to get a test passing.
- Repeat until all tests are passing.

Tips and Tricks

- **Note: If you find yourself stuck on a problem for longer than fifteen minutes, move on to the next, and try again later.**
- As a developer, you'll find documentation for classes and libraries to be invaluable resources when completing your work. Get into the habit of reading and understanding documentation now to level up more quickly on your journey as a developer. For instance, the Map interface (which you'll use for this exercise) is well-documented.
- Before each method, there is a description of the problem that needs to be solved, as well as examples with the expected output. Use these examples to figure out the values you need to write your code around. For example, in the comments above the `animalGroupName` method, there is a section that includes the method name, as well as the expected value that is returned for each method call. The following example signifies that when the method is called with "giraffe", it returns the word "Tower":

```
animalGroupName("giraffe") → "Tower"
```

- 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 Eclipse to help resolve issues.