Practice: Writing Methods and Tests

1 Problems

Try these problems, don't just look at the answers. If you do look at the answers and find you made mistakes, write the answers out again by hand, even if you are mostly copying the answers. People learn differently from writing versus reading details: you are much more likely to absorb the constructs for writing methods if you actually write them out yourself. If you type instead of write by hand, don't use cut and paste. You actually have to write out the characters yourself to help you absorb the patterns.

Make sure you also write test cases for each method. Testing is one of the key themes of the course. If you aren't sure what to write for test cases, come to office hours for help.

The following link provides starter files with class definitions for these problems.

`http://brown-cs18-master.github.io/content/practice/practice02.zip`

1. In the `Planet` class, add a method `convertWeight` that takes the weight of an object on earth (as a number of type `double`) and produces the corresponding weight on that planet. Since the class has the planet's gravity relative to earth's, you just need to multiply the given weight by the relative gravity to get the answer for this method.

2. In the `Planet` class, add a method `fartherFromSun` that takes a planet and produces a boolean indicating whether the planet is farther from the sun than the planet given as input.

3. In the `BankCustomer` class, write a method `openAccount` that takes an initial deposit and returns a new bank account that has the initial deposit as the amount and the customer object as the customer.

4. In the `BankAccount` class, write a method `statusMessage` that takes no arguments and returns a string. The string should be one of "Positive Balance", "No Money", or "Overdrawn", depending on whether the balance is positive, zero, or negative (respectively).

2 Answers

`http://brown-cs18-master.github.io/content/practice/Practice2.java`

This file puts all of the classes in a single file for readability. If you want to actually run these with IntelliJ, you'd have to split the classes into separate files (in a new Java package).

These solutions have one example test case, but a good solution would have several more.

Please let us know if you find any mistakes, inconsistencies, or confusing language in this or any other CS18 document by filling out the anonymous feedback form: `https://cs.brown.edu/courses/cs018/feedback`.