Assignment 2

Classes and Objects

Introduction

In this assignment, you will write and test non-static methods in the `Bike` and `Brand` classes for Part 1, and then implement and test static methods in the `A2Exercises` class for Part 2.

The automated grading of your assignment will include some different and additional tests to those found in the `A2Tester.java` file, as it does not include a comprehensive set of sets for each method. You are expected to write additional tests until you are convinced each method has full test coverage. The `displayResults` and test coverage videos provide more information about code testing.

Objectives

Upon finishing this assignment, you should be able to:

- Write methods that operate on objects and arrays of objects in Java;
- Describe the difference between a static and non-static method;
- Use the `displayResults` method for code testing in Java.

Submission and Grading

Attach `Bike.java`, `Brand.java` and `A2Exercises.java` to the BrightSpace assignment page. Remember to click **submit**. You should receive notification that your assignment was successfully submitted.

If you chose not to complete some of the methods required, you must provide a stub for the incomplete method(s) in order for our tester to compile. There are stubs for each method right now, which is why it compiles without issue. Notice that all the provided methods have a correct signature (name, return type, and parameter list), allowing the tester to call the methods.

If you submit files that do not compile with our tester, you will receive a zero grade for the assignment. It is your responsibility to ensure you follow the specification and submit the correct files. Additionally, your code must not be written to specifically pass the test cases in the tester, instead, it must work on all valid inputs. We may change the input values during grading and we will inspect your code for hard-coded solutions. This video explains stubs.

Be sure you submit your assignment, not just save a draft. All late and incorrect submissions will be given a zero grade. A reminder that it is OK to talk about your assignment with your classmates, but not to share code electronically or visually (on a display screen or paper). Plagiarism detection software will be run on all submissions.

Instructions

1. Download all of the `.java` files found in the **Assignments -> Assignment 2** page on BrightSpace.
2. Read through the `Bike.java` and `Brand.java` files provided for you. The fields and methods found in the two classes are overviewed in the UML diagram shown below.
3. Compile and run `A2Tester.java`. Note the new way we perform tests, using the `displayResults` method. For more information about how this method is used watch the `displayResults` video.
4. Open the `A2Tester.java` file. Read through the tests for `testBikeConstructor` to see what the expected behaviour is for the `Bike` constructor. Implement the `Bike` constructor, save, and then

recompile and run `A2Tester.java`. If any tests fail, fix the errors until all of the tests pass for that method. Add any tests that you feel are necessary.

5. After finishing your implementation of the `Bike` constructor, uncomment the second test method in `A2Tester.java` (`testEquals`). Follow the same process that you did in the previous step until you are certain the `equals` method has been implemented correctly in the `Bike` class.
6. Continue moving down one test method at a time until all of the required methods have been implemented. Remember to write additional tests until you are sure that your implementation will work correctly under all different possible input scenarios. It may be worthwhile reviewing the video on test coverage when considering which additional tests to write.

CRITICAL: Any compile or runtime errors will result in a **zero grade** (if the tester crashes, it will not be able to award you any points for any previous tests that may have passed). Make sure to compile and run your program **before** submitting it!

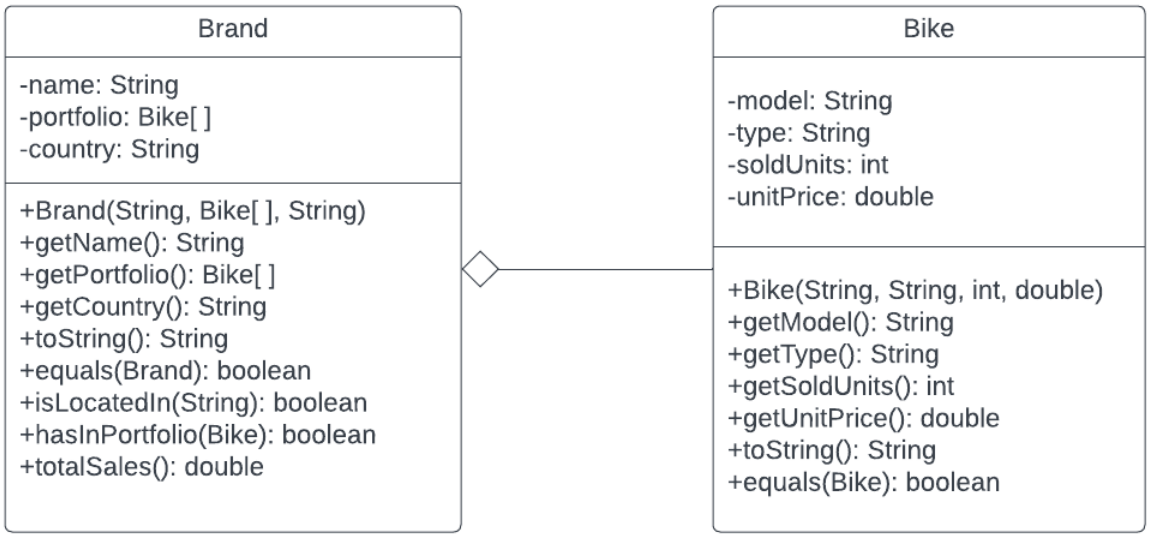

Figure 1: A UML diagram showing the fields and methods found in Bike.java and Brand.java.