Task-4

1. Write a test:

Create a test case that specifies the desired behavior of the code.

2. Run the test:

Execute the test case to see if it fails (which it should initially because the code hasn't been implemented yet).

3. Write the code:

Implement the code to make the test pass.

4. Run the test again:

Re-run the test case to see if it passes with the newly written code.

5. Refactor (optional):

If necessary, improve the code without changing its behavior and run the test again to ensure it still passes.

6. Repeat:

Continue the cycle for each new feature or enhancement.

Step 1: Write the test

Create a new test file, e.g., CarTest.php, and write the initial test for the Car class:

```
<?php
use PHPUnit\Framework\TestCase;
require_once 'your_code_file.php';
class CarTest extends TestCase {
   public function testCarModel() {
      $car = new Car("Calliope", "Capulet Engine", "Spindler Battery");
      $this->assertEquals("Calliope", $car->getCarModel());  }}
```

Step 2: Run the test

Run the test by executing the following command in the terminal:

```
phpunit CarTest.php
```

Since you haven't implemented the Car class yet, the test should fail with a message indicating that the class Car is not found.

Step 3: Write the code

Now, you'll implement the Car class along with its methods in the actual code file, e.g., your_code_file.php.

```
class Car {
    // ... (existing code for Car class)
}
```

Step 4: Run the test again

Run the test again using the same PHPUnit command:

```
phpunit CarTest.php
```

This time, the test should pass since you've created the Car class.

Step 5: Refactor (optional)

If you want to improve or optimize the code without changing its behavior, do so at this stage. However, since we have a simple implementation, refactoring may not be necessary in this case.

Step 6: Repeat

For each additional feature or enhancement, continue the TDD cycle by writing new tests, implementing the code, and running the tests to verify that everything works as expected.

Keep iterating through the TDD cycle to build a comprehensive test suite for your code, ensuring that it remains reliable and maintainable.