Lab Assignment: PHP Class Implementation – Car Inventory System with Pricing Validation

Objective:

In this lab, you will create a PHP class representing **Cars** in an inventory system. You will focus on implementing the following:

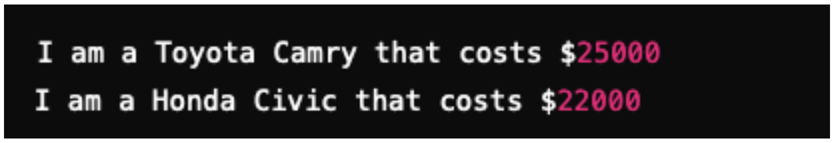
- A **static property** to assign unique IDs to each car.
- **Private and public properties** to represent car details (such as make, model, and price).
- A method to **validate and update** the car price, ensuring it doesn't change by more than 10%.
- A method to retrieve car details as an **associative array**.
- A toString() method to print out car details.

Part 1: Class Definition

1. **Create the Car Class:**
   - Define a class called Car.
   - Create a **static property** $id that assigns a unique ID to each car.
   - Add **public properties** for make and model.
   - Add a **private property** for price.
   - Add a **private property** for carID to store each car's unique ID.
2. **Constructor:**
   - The constructor should:
     - Accept make, model, and price as parameters.
     - Assign a unique carID to each car using the static property $id and increment $id for the next car.
     - Initialize the make, model, and price properties.
3. **setPrice() Method:**
   - Write a setPrice() method that:
     - Accepts a new price as a parameter.
     - Calculates 10% of the current price and defines the minimum and maximum allowable price.
     - Updates the price only if the new price is within ±10% of the current price. Otherwise, display an error message.
4. **getCar() Method:**
   - Write a getCar() method that returns an associative array containing the car's make, model, price, and carID.
5. **toString() Method:**
   - Write a toString() method that outputs a string in the format:
     "I am a [make] [model] that costs $[price]".

Part 2: Instantiate Car Objects

1. **Create Car Instances:**
   - Create two instances of the Car class with the following details:
     - Car 1: "Toyota", "Camry", Price 25000.00
     - Car 2: "Honda", "Civic", Price 22000.00
2. **Store Cars in an Array:**
   - Store both car objects in an array called $aryCars.
3. **Display Car Details Using toString():**
   - Loop through the array and display each car's details using the toString() method.



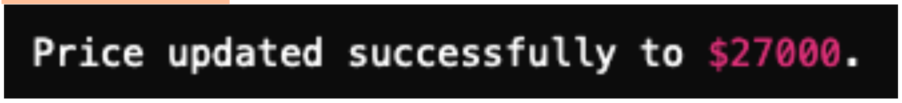
```
I am a Toyota Camry that costs $25000
I am a Honda Civic that costs $22000
```

Part 3: Test Cases and Expected Outputs

Test Case 1: Valid Price Update for Car 1

- Current Price: 25000.00
- New Price: 27000.00 (within the 10% range)

Expected Output:

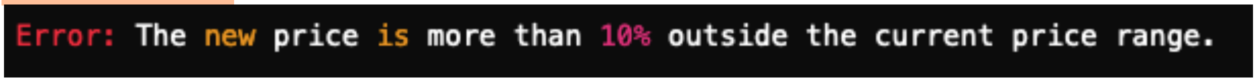


```
Price updated successfully to $27000.
```

Test Case 2: Invalid Price Update for Car 1

- **Current Price**: 27000.00
- **New Price**: 35000.00 (exceeds the 10% range)

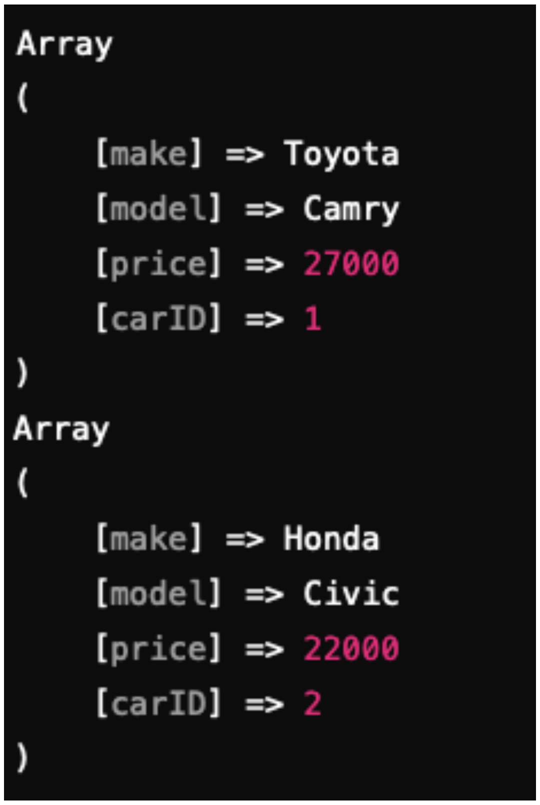
Expected Output:



```
Error: The new price is more than 10% outside the current price range.
```

Test Case 3: Retrieve Car Details Using getCar()

- **Car 1 Details:** "Toyota", "Camry", 27000.00, carID = 1
- **Car 2 Details:** "Honda", "Civic", 22000.00, carID = 2



```
Array
(
    [make] => Toyota
    [model] => Camry
    [price] => 27000
    [carID] => 1
)
Array
(
    [make] => Honda
    [model] => Civic
    [price] => 22000
    [carID] => 2
)
```