| **Experiment No. – 3** | | | | |
| --- | --- | --- | --- | --- |
| **Date of Performance:** | **22/1/25** | | | |
| **Date of Submission:** | **29/1/25** | | | |
| Program Execution/  formation/  correction/  ethical practices  (06) | Timely  Submission  (01) | Viva  (03) | Experiment  Total (10) | Sign with Date |
|  |  |  |  |  |

**Experiment No. 3**

**3.1 Aim:** Implement Inheritance using TypeScript and HTML

**3.2 Course Outcome:** CO2: Understand how TypeScript and AngularJS framework can build dynamic, responsive single-page web applications

**3.3 Learning Objectives:** To understand how inheritance works in TypeScript by creating base and derived classes, compiling and running the code in the browser, and displaying the output using HTML.

**3.4 Requirement:**

* Visual Studio Code / Any IDE
* TypeScript Compiler (tsc)
* Node.js (for compilation)
* Browser (Chrome/Edge/Firefox)

3.5 Related Theory:

Inheritance is a fundamental concept in Object-Oriented Programming (OOP) that allows one class (called the *child* or *derived* class) to inherit the properties and behaviors (methods) of another class (called the *parent* or *base* class). This helps in code reusability, readability, and maintaining a cleaner and modular codebase.

Key Concepts in TypeScript Inheritance:

1. Class Declaration: A class in TypeScript is declared using the class keyword. It can contain properties (variables) and methods (functions).
2. Base (Parent) Class: The base class defines common features that can be inherited by derived classes.
3. Derived (Child) Class: The derived class uses the extends keyword to inherit the parent class.
4. Constructor and super(): If a derived class has a constructor, it must call the parent class constructor using the super() method before using this.
5. Method Overriding: A derived class can override methods of the base class to provide its own implementation.
6. Access Modifiers: TypeScript uses access modifiers like:
   * public: Accessible anywhere.
   * private: Accessible only within the class.
   * protected: Accessible within the class and its subclasses.

3.6 Procedure:

Step 1: Create a TypeScript file (inheritance.ts) containing base and derived classes.

Step 2: Use the tsc inheritance.ts command to compile it into JavaScript.

Step 3: Link the compiled JS file in an HTML file.

Step 4: Use innerHTML to show the output.

Step 5: Run the HTML file in the browser.

**3.7 Program and Output:**

**HTML:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Multilevel Inheritance in TypeScript</title>

<!-- Bootstrap CSS -->

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">

</head>

<body>

  <div class="container mt-5">

        <div class="card shadow-lg">

            <div class="card-header bg-primary text-white text-center">

                <h2>Vehicle Information</h2>

            </div>

            <div class="card-body">

                <form id="vehicleForm">

                    <div class="mb-3">

                        <label for="brand" class="form-label">Brand:</label>

                        <input type="text" id="brand" class="form-control" required>

                    </div>

                    <div class="mb-3">

                        <label for="model" class="form-label">Model:</label>

                        <input type="text" id="model" class="form-control" required>

                    </div>

                    <div class="mb-3">

                        <label for="batteryLife" class="form-label">Battery Life (hrs):</label>

                        <input type="number" id="batteryLife" class="form-control" required>

                    </div>

                    <button type="submit" class="btn btn-success w-100">Add Vehicle</button>

                </form>

            </div>

        </div>

        <div id="output" class="mt-4 p-3 border rounded bg-light"></div>

    </div>

<!-- Bootstrap JS -->

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>

    <script src="main.js"></script>

</body>

</html>

TypeScript:

// Base Class

class Vehicle {

    brand: string;

    constructor(brand: string) {

        this.brand = brand;

    }

    displayBrand(): string {

        return `<strong>Brand:</strong> ${this.brand}`;

    }

}

// Derived Class (inherits from Vehicle)

class Car extends Vehicle {

    model: string;

    constructor(brand: string, model: string) {

        super(brand);

        this.model = model;

    }

    displayModel(): string {

        return `<strong>Model:</strong> ${this.model}`;

    }

}

// Another Derived Class (inherits from Car)

class ElectricCar extends Car {

    batteryLife: number;

    constructor(brand: string, model: string, batteryLife: number) {

        super(brand, model);

        this.batteryLife = batteryLife;

    }

    displayDetails(): string {

        return `

            <div class="alert alert-info">

                ${this.displayBrand()} <br>

                ${this.displayModel()} <br>

                <strong>Battery Life:</strong> ${this.batteryLife} hours

            </div>`;

    }

}

// Event Listener for Form Submission

document.getElementById("vehicleForm")!.addEventListener("submit", function(event) {

    event.preventDefault();

    const brand = (document.getElementById("brand") as HTMLInputElement).value;

    const model = (document.getElementById("model") as HTMLInputElement).value;

    const batteryLife = parseInt((document.getElementById("batteryLife") as HTMLInputElement).value);

    // Create an instance of ElectricCar

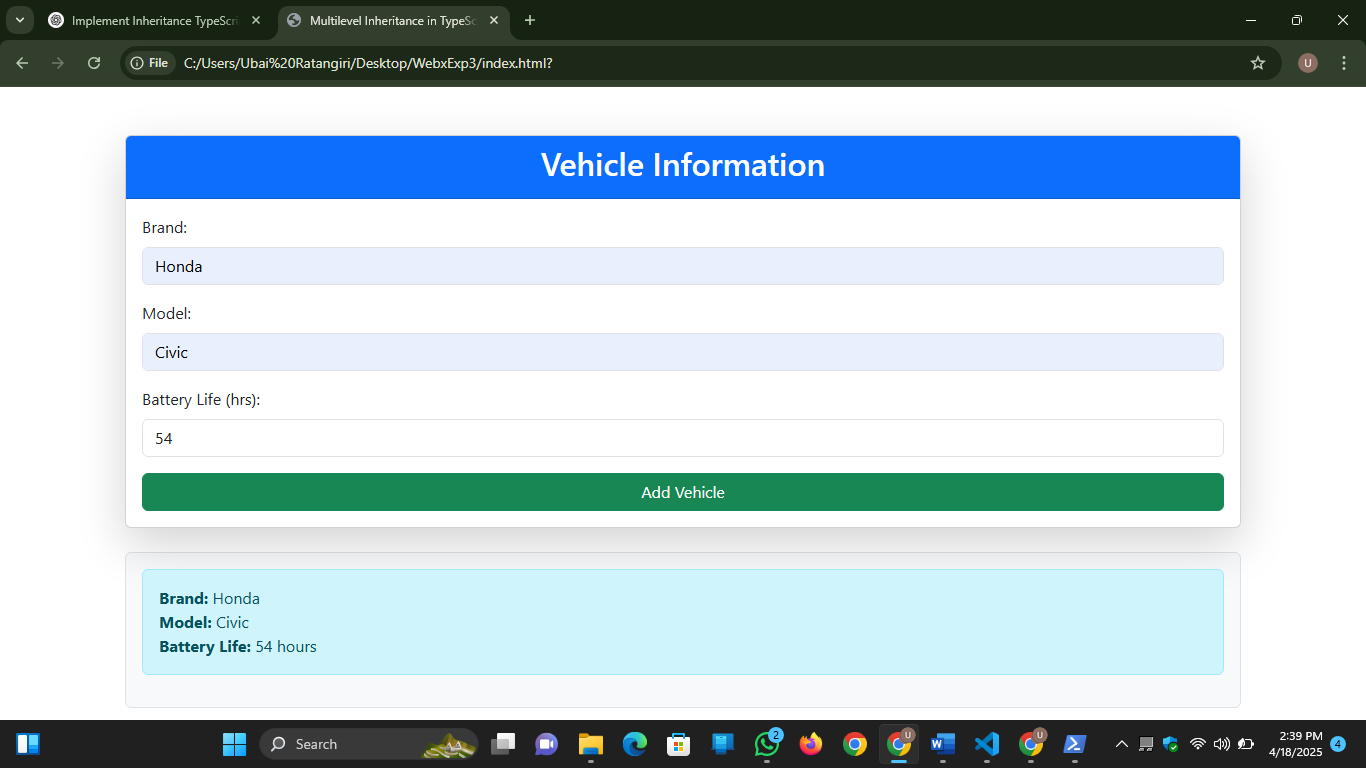
    const myCar = new ElectricCar(brand, model, batteryLife);

    // Display the result

    document.getElementById("output")!.innerHTML = myCar.displayDetails();

});

Output:



**3.8 Conclusion:** The experiment successfully demonstrated how inheritance works in TypeScript using classes and how the output can be shown using an HTML page. It reinforced key OOP concepts in a web development context.

**3.9 Questions:**

1. What is the role of the super() keyword in TypeScript?

**super()** is used to call the constructor or methods of a parent class from a derived class.

1. How is inheritance different in Typescript compared to JavaScript?

TypeScript adds static typing and better support for class-based inheritance compared to JavaScript, making it easier to define and enforce class hierarchies.

1. Can TypeScript support multiple inheritance? If not, how is it achieved?

TypeScript does not support multiple inheritance directly, but it can be achieved using interfaces and mixins.