**EXPERIMENT 3.3**

**AIM:**

Person Class Hierarchy with Student and Teacher Subclasses

**CODE:**

<!doctype html>

<html lang="en">

<head>

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1" />

<title>Person → Student & Teacher (HTML + JS)</title>

<style>

body{font-family:system-ui,-apple-system,Segoe UI,Roboto,Helvetica,Arial;max-width:900px;margin:40px auto;padding:20px}

h1{font-size:1.4rem;margin-bottom:0.2rem}

.card{border:1px solid #ddd;border-radius:8px;padding:12px;margin:10px 0;background:#fafafa}

label{display:block;margin:8px 0 4px}

input,select,button{padding:8px;border-radius:6px;border:1px solid #ccc}

.row{display:flex;gap:8px}

.row > \*{flex:1}

.actions{margin-top:10px}

pre{background:#111;color:#eee;padding:12px;border-radius:6px;overflow:auto}

</style>

</head>

<body>

<h1>Person class hierarchy — Person, Student, Teacher (ES6 classes)</h1>

<p>This page demonstrates JavaScript classes for <strong>Person</strong> and two subclasses: <strong>Student</strong> and <strong>Teacher</strong>. Add examples using the form below and see them rendered.</p>

<div class="card">

<form id="entityForm">

<label for="type">Type</label>

<select id="type">

<option value="student">Student</option>

<option value="teacher">Teacher</option>

</select>

<label for="name">Name</label>

<input id="name" placeholder="e.g. Alice" required />

<div class="row">

<div>

<label for="age">Age</label>

<input id="age" type="number" min="0" placeholder="20" required />

</div>

<div id="extraField">

<!-- student or teacher specific field injected here -->

</div>

</div>

<div class="actions">

<button type="submit">Add</button>

<button type="button" id="clearBtn">Clear All</button>

</div>

</form>

</div>

<div id="list" aria-live="polite"></div>

<h2>Example code (short)</h2>

<pre id="snippet"></pre>

<script>

// Base class

class Person {

constructor(name, age) {

this.name = name;

this.age = Number(age);

}

displayInfo() {

return `Name: ${this.name}, Age: ${this.age}`;

}

}

// Student subclass

class Student extends Person {

constructor(name, age, studentId, course) {

super(name, age);

this.studentId = studentId;

this.course = course;

}

displayInfo() {

return `${super.displayInfo()}, Student ID: ${this.studentId}, Course: ${this.course}`;

}

}

// Teacher subclass

class Teacher extends Person {

constructor(name, age, employeeId, subject) {

super(name, age);

this.employeeId = employeeId;

this.subject = subject;

}

displayInfo() {

return `${super.displayInfo()}, Employee ID: ${this.employeeId}, Subject: ${this.subject}`;

}

}

// Simple in-memory list to store instances

const items = [];

// DOM elements

const typeEl = document.getElementById('type');

const extraField = document.getElementById('extraField');

const form = document.getElementById('entityForm');

const list = document.getElementById('list');

const snippet = document.getElementById('snippet');

const clearBtn = document.getElementById('clearBtn');

// Build the student-specific inputs by default

function renderExtraInputs() {

const t = typeEl.value;

if (t === 'student') {

extraField.innerHTML = `

<label for="studentId">Student ID</label>

<input id="studentId" placeholder="S101" required />

<label for="course">Course</label>

<input id="course" placeholder="Computer Science" required />

`;

} else {

extraField.innerHTML = `

<label for="employeeId">Employee ID</label>

<input id="employeeId" placeholder="T202" required />

<label for="subject">Subject</label>

<input id="subject" placeholder="Mathematics" required />

`;

}

}

typeEl.addEventListener('change', renderExtraInputs);

renderExtraInputs();

function renderList() {

if (!items.length) {

list.innerHTML = '<p><em>No entries yet.</em></p>';

return;

}

list.innerHTML = items.map((it, idx) => {

return `

<div class="card">

<strong>${it.\_\_type\_\_.toUpperCase()}</strong>

<p>${it.displayInfo()}</p>

<button data-index="${idx}" class="deleteBtn">Delete</button>

</div>

`;

}).join('');

// attach delete handlers

document.querySelectorAll('.deleteBtn').forEach(btn => {

btn.addEventListener('click', e => {

const i = Number(e.currentTarget.dataset.index);

items.splice(i,1);

renderList();

})

})

}

form.addEventListener('submit', e => {

e.preventDefault();

const name = document.getElementById('name').value.trim();

const age = document.getElementById('age').value;

if (typeEl.value === 'student') {

const studentId = document.getElementById('studentId').value.trim();

const course = document.getElementById('course').value.trim();

const s = new Student(name, age, studentId, course);

s.\_\_type\_\_ = 'student';

items.push(s);

} else {

const employeeId = document.getElementById('employeeId').value.trim();

const subject = document.getElementById('subject').value.trim();

const t = new Teacher(name, age, employeeId, subject);

t.\_\_type\_\_ = 'teacher';

items.push(t);

}

form.reset();

renderExtraInputs();

renderList();

updateSnippet();

});

clearBtn.addEventListener('click', () => {

if (!confirm('Clear all entries?')) return;

items.length = 0;

renderList();

updateSnippet();

});

function updateSnippet() {

snippet.textContent = `// Example usage:\nconst s = new Student('Alice', 20, 'S101', 'CS');\nconsole.log(s.displayInfo());\n\nconst t = new Teacher('Mr. Smith', 45, 'T202', 'Math');\nconsole.log(t.displayInfo());`;

}

// initial render

renderList();

updateSnippet();

</script>

</body>

</html>

**OUTPUT:**

