PROCESO DE GESTIÓN DE LA FORMACIÓN PROFESIONAL INTEGRAL
FORMATO ENTREGA DE EVIDENCIAS

PROGRAMACIÓN ORIENTADA A OBJETOS EN JAVASCRIPT

Presentado a:	Instructor César Marino Cuéllar Chacón
Por Aprendiz:	**Melva Cerón Buitrón**
Ficha:	3064975
Competencia:	Diseñar la solución de software de acuerdo con procedimientos y requisitos técnicos
Resultado de Aprendizaje:	Verificar los entregables de la fase de diseño del software de acuerdo con lo establecido en el informe de análisis

Tecnólogo en Análisis y Desarrollo de Software
Servicio Nacional de Aprendizaje SENA
Centro de Teleinformática y Producción Industrial
Regional Cauca

Popayán, día **31** de **agosto** del año **2025**

PROCESO DE GESTIÓN DE LA FORMACIÓN PROFESIONAL INTEGRAL FORMATO ENTREGA DE EVIDENCIAS

Tabla de Contenido

PROCESO DE GESTIÓN DE LA FORMACIÓN PROFESIONAL INTEGRAL
FORMATO ENTREGA DE EVIDENCIAS

UNIDAD 1: Introducción a la POO en JS

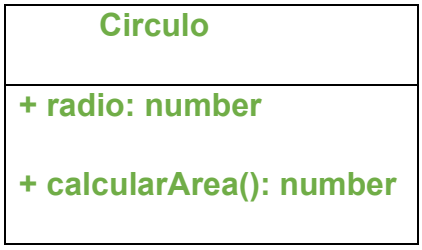
1. Actividad 1, Ejercicio1

1.1 Enunciado

Ejercicio 1 – Clase Círculo
Crear una clase Círculo con radio y método calcularÁrea().

Solución

Diagrama de clases



1.2 Solución

Código en JavaScript

```
class Circulo {
   constructor(radio) {
      this.radio = radio;
   }
```

```
    calcularArea() {
        return Math.PI * this.radio * this.radio;
    }
}

// Crear un objeto círculo
let c1 = new Circulo(5);
console.log("Área del círculo:", c1.calcularArea());
```

1.Actividad1 o Ejercicio2

1.3 Enunciado

Crear una clase Estudiante con nombre, materias (array), y método listarMaterias().

1.4 Solución

Diagrama de clases:

Estudiante
+nombre: string
+ materias: string[]
+ listarMaterias(): void

Código en JavaScript:

```
class Estudiante {
    constructor(nombre, materias) {
        this.nombre = nombre;
        this.materias = materias; // arreglo
    }

    listarMaterias() {
        console.log("Materias de " + this.nombre + ":");
        for (let i = 0; i < this.materias.length; i++) {
            console.log("- " + this.materias[i]);
        }
    }
}
```

```
// Crear objeto estudiante
let e1 = new Estudiante("Ana", ["Matemáticas", "Inglés",
"Programación"]);
e1.listarMaterias();
```

UNIDAD 3: Encapsulamiento

2. Actividad2, Ejercicio1

2.1 Enunciado

Crear una clase Empleado con sueldo privado y método para aplicar aumento.

2.1.1 solución

Diagrama de Clases

Empleado
+ nombre: string - sueldo: number
+constructor(nombre, sueldo) +aplicarAumento(porc: number): void + getSueldo(): number

Código en JavaScript:

```
class Empleado {

    constructor(nombre, sueldo) {

        this.nombre = nombre;

        this._sueldo = sueldo; // privado por convención con "_"

    }
```

```
    aplicarAumento(porcentaje) {
        this._sueldo += this._sueldo * (porcentaje / 100);
    }
    getSueldo() {
        return this._sueldo;
    }
}
// Uso
let emp1 = new Empleado("Diana", 2000000);
console.log("Sueldo inicial:", emp1.getSueldo());
emp1.aplicarAumento(10);
console.log("Sueldo con aumento:", emp1.getSueldo());
```

3. Actividad3, Ejercicio1

3.1. Enunciado

Implementar un getter y un setter para correo que valide el formato.

3.1.2. Solución

Empleado
+ nombre: string - correo: string
+ constructor(nombre, correo) + getCorreo(): string + setCorreo(correo: string): void

Código en JavaScript:

```
class Empleado {
    constructor(nombre, correo) {
        this.nombre = nombre;
        this._correo = correo;
    }
    getCorreo() {
        return this._correo;
    }
    setCorreo(nuevoCorreo) {
        // Validar formato con expresión regular básica
        let regex = /^[^@\s]+@[^@\s]+\.[^@\s]+$/;
        if (regex.test(nuevoCorreo)) {
            this._correo = nuevoCorreo;
        } else {
            console.log("Correo inválido. Intente nuevamente.");
        }
    }
}
// Uso
let emp2 = new Empleado("Pedro", "pedro@mail.com");
console.log("Correo inicial:", emp2.getCorreo());
emp2.setCorreo("correo_invalido"); // dará error
emp2.setCorreo("nuevo@mail.com");  // aceptado
console.log("Correo actualizado:", emp2.getCorreo());
```

4. Actividad3, Ejercicio1

4.1 Enunciado

• Crear una clase Persona y dos clases hijas, una Instructor y otra Aprendiz:

o Persona:

▪ Atributos: identificación, nombre, correo

o Aprendiz:

▪ Atributo: puntajeIcfes ▪ Métodos: info(): muestra un mensaje con todos los atributos del aprendiz

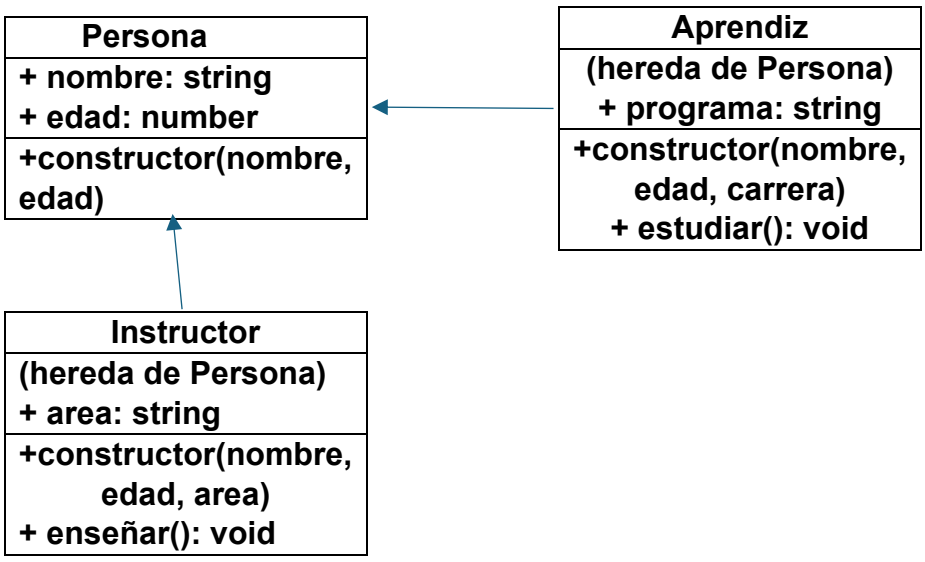
o Instructor:

▪ Atributo: especialidad

▪ Méodos: info(): muestra un mensaje con todos los atributos del instructor

4.1.1 Solución

Diagramas de Clases



4.1.2. solución

Código en JavaScript:

```
// Clase base
```

```
class Persona {
    constructor(nombre, edad) {
        this.nombre = nombre;
        this.edad = edad;
    }
}
// Subclase Estudiante
class Estudiante extends Persona {
    constructor(nombre, edad, carrera) {
        super(nombre, edad);
        this.carrera = carrera;
    }
    estudiar() {
        console.log(this.nombre + " está estudiando la carrera de " + this.carrera);
    }
}
// Subclase Instructor
class Instructor extends Persona {
    constructor(nombre, edad, area) {
        super(nombre, edad);
        this.area = area;
    }
    enseñar() {
        console.log(this.nombre + " enseña en el área de " + this.area);
    }
}
// Uso
let est1 = new Estudiante("Juan", 22, "Sistemas");
```

est1.estudiar();

let ins1 = new Instructor("Marta", 40, "Programación");

ins1.enseñar();

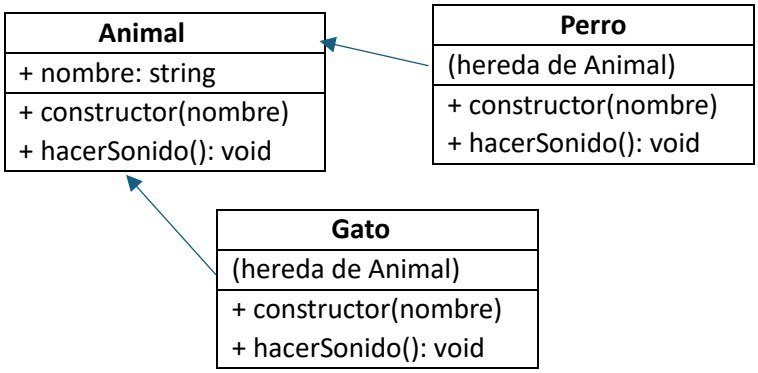
5. Actividad5, Ejercicio1

5.1.1 enunciado.

• Crear una clase Animal y clases hijas Perro, Gato, con comportamiento específico.

5.1.2 solución

Diagramas de Clases



Código javaScript

```
// Clase base
class Animal {
   constructor(nombre) {
      this.nombre = nombre;
   }
   hacerSonido() {
      console.log(this.nombre + " hace un sonido...");
   }
}
// Subclase Perro
class Perro extends Animal {
```

```
    constructor(nombre) {
        super(nombre);
    }
    hacerSonido() {
        console.log(this.nombre + " dice: ¡Guau guau!");
    }
}
// Subclase Gato
class Gato extends Animal {
    constructor(nombre) {
        super(nombre);
    }
    hacerSonido() {
        console.log(this.nombre + " dice: ¡Miau miau!");
    }
}
// Uso
let perro1 = new Perro("Firulais");
perro1.hacerSonido();
let gato1 = new Gato("Misu");
gato1.hacerSonido();
```

UNIDAD 5: Composición y Relaciones

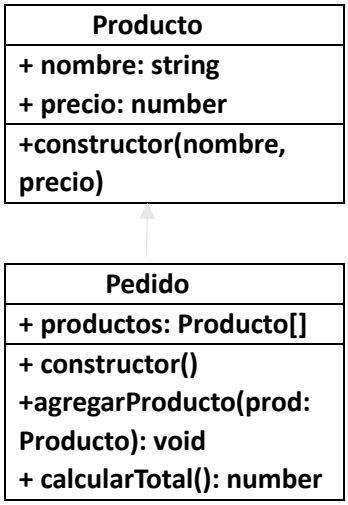
Actividad 6, Ejercicio 1

6.1 Enunciado

Crear una clase Pedido que contenga una lista de objetos Producto

6.1.2 Solución

Diagramas de Clases



6.1.2 Solución

Código javaScript

```
class Producto {
   constructor(nombre, precio) {
      this.nombre = nombre;
      this.precio = precio;
   }
}

class Pedido {
   constructor() {
      this.productos = []; // array de productos
   }

   agregarProducto(producto) {
      this.productos.push(producto);
   }

   calcularTotal() {
      let total = 0;
      for (let i = 0; i < this.productos.length; i++) {
         total += this.productos[i].precio;
      }
      return total;
   }
}

// Uso
```

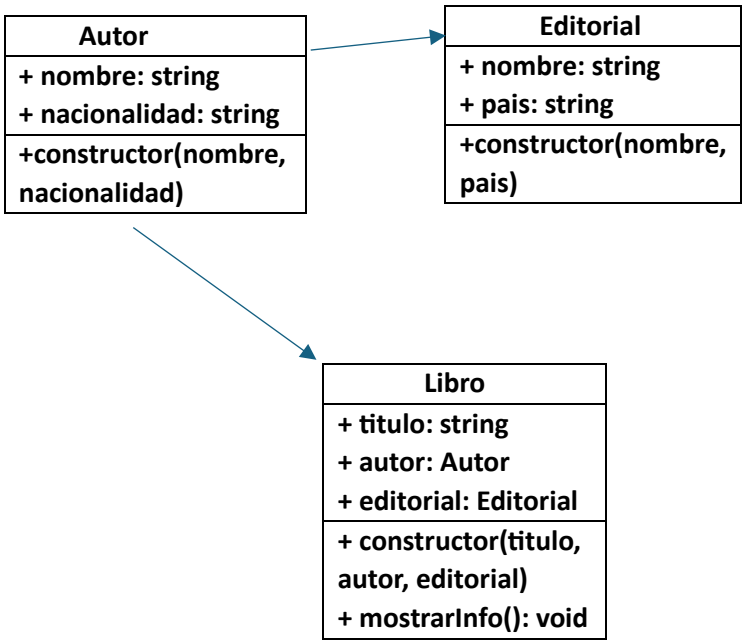
```
let pedido1 = new Pedido();
pedido1.agregarProducto(new Producto("Pan", 1500));
pedido1.agregarProducto(new Producto("Queso", 5000));
pedido1.agregarProducto(new Producto("Leche", 2500));

console.log("Total del pedido:", pedido1.calcularTotal());
```

6.2 Enunciado

Modelar un sistema de biblioteca con Libro, Autor, Editorial relacionados

6.2.1 Solución:

Diagramas de Clases



6.2.2. Solución:
Código javaScript

```
class Autor {
   constructor(nombre, nacionalidad) {
      this.nombre = nombre;
      this.nacionalidad = nacionalidad;
   }
}

class Editorial {
   constructor(nombre, pais) {
      this.nombre = nombre;
      this.pais = pais;
   }
}

class Libro {
   constructor(titulo, autor, editorial) {
      this.titulo = titulo;
      this.autor = autor;        // composición
      this.editorial = editorial; // composición
   }

   mostrarInfo() {
      console.log("Título: " + this.titulo);
      console.log("Autor: " + this.autor.nombre + " (" + this.autor.nacionalidad + ")");
      console.log("Editorial: " + this.editorial.nombre + " - " + this.editorial.pais);
   }
}

// Uso
let autor1 = new Autor("Gabriel García Márquez", "Colombiano");
let editorial1 = new Editorial("Sudamericana", "Argentina");

let libro1 = new Libro("Cien Años de Soledad", autor1, editorial1);

libro1.mostrarInfo();
```

6. Bibliografía

- Material de apoyo del instructor Cesar Marino Cuellar

- 1. Tutorial de Javascript Modernohttps://es.javascript.info/
- 2. Tutorial de w3schools: https://www.w3schools.com/js/
- 3. Curso en youtube: https://www.youtube.com/watch?v=Z34BF9PCfYg&t=106s
- 4. Curso internet: https://lenguajejs.com/javascript/