



Nombre: Jhon Paúl Arcentales Cevallos

Carrera: Desarrollo de Software

Materia: Metodologías para resolver problemas informáticos

Clase: Clases

Actividad: Ejercicios sobre clases

Fecha: 30/11/2025

Objetivo de la actividad:

Creación de clases independientes mediante typescript, comprender como funcionan clases.

Desarrollo:

Clase Banco:

Se implementaron operaciones básicas como depositar, retirar y transferir dinero, incluyendo validaciones para evitar saldos negativos.

Clase Estudiante:

Se programaron métodos para calcular un nuevo promedio simple, verificar si un estudiante aprueba y llevar el registro de faltas.

Clase Auto:

Se simulan acciones del vehículo: acelerar, frenar y calcular velocidad media. También se hizo que no se puedan mostrar números negativos.

Clase Restaurante:

Se implementaron cálculos relacionados con el costo de un pedido: total, descuento aplicado y propina.

Clase Hospital:

Se programaron métodos utilizados en contextos médicos básicos: cálculo de dosis según peso, IMC y registro de visitas. Usamos console.log para mostrar los resultados finales.

Código:

```
// CLASE BANCO
```

```
//Suma el monto al saldo actual
```

```
class Banco {
```

```
    depositar(saldo: number, monto: number): number {
```

```
        return saldo + monto;
```

```
    }
```

```
//resta el monto al saldo actual si hay fondos suficientes
```

```
    retirar(saldo: number, monto: number): number {
```

```
        if (monto > saldo) {
```

```
            throw new Error("Fondos insuficientes");
```

```
        }
```

```
        return saldo - monto;
```

```
    }
```

```
//transferencia: resta del saldo de origen
```

```
    transferir(saldoOrigen: number, monto: number): number {
```

```
        if (monto > saldoOrigen) {
```

```
            throw new Error("No se puede transferir más del saldo disponible");
```

```
        }
```

```
        return saldoOrigen - monto;
```

```
    }
```

```
}
```

```
// CLASE ESTUDIANTE
```

```
//calcula nuevo promedio usndo dos notas
```

```
class Estudiante {
```

```
    agregarNota(promedio: number, nuevaNota: number): number {  
        return (promedio + nuevaNota) / 2;  
    }  
  
    //verifica si la nota es mayor o igual a la mínima para aprobar  
    aprobar(nota: number, minima: number): boolean {  
        return nota >= minima;  
    }  
  
    //suma las faltas actuales con las nuevas  
    calcularFaltas(faltasActuales: number, nuevas: number): number {  
        return faltasActuales + nuevas;  
    }  
}
```

// CLASE AUTO

//Aumenta la velocidad

```
class Auto {  
    acelerar(velocidadActual: number, incremento: number): number {  
        return velocidadActual + incremento;  
    }  
}
```

//reduce la velocidad sin ir por debajo de 0

```
    frenar(velocidadActual: number, decremento: number): number {  
        const nueva = velocidadActual - decremento;  
        return nueva < 0 ? 0 : nueva;  
    }  
}
```

//calcula la velocidad media

```
    recorrer(distancia: number, tiempo: number): number {  
        return distancia / tiempo;  
    }  
}
```

```
}  
}
```

```
// CLASE RESTAURANTE
```

```
//multiplica precio por cantidad
```

```
class Restaurante {  
    calcularTotal(precio: number, cantidad: number): number {  
        return precio * cantidad;  
    }  
}
```

```
//aplica un descuento porcentual al total
```

```
    aplicarDescuento(total: number, descuento: number): number {  
        return total - (total * descuento / 100);  
    }  
}
```

```
//calcula la propina segun porcentaje
```

```
    calcularPropina(total: number, porcentaje: number): number {  
        return total * (porcentaje / 100);  
    }  
}
```

```
// CLASE HOSPITAL
```

```
//calcula dosis segun peso
```

```
class Hospital {  
    calcularDosis(peso: number, mgPorKg: number): number {  
        return peso * mgPorKg;  
    }  
}
```

```
//calcula el indice de masa corporal
```

```
    calcularIMC(peso: number, altura: number): number {  
        return peso / (altura * altura);  
    }  
//suma las visitas nuevas  
    registrarVisitas(visitasActuales: number, nuevas: number): number {  
        return visitasActuales + nuevas;  
    }  
}
```

//EJEMPLOS DE USO

//BANCO

```
console.log("=== BANCO ===");  
const banco = new Banco();  
console.log("Depositar 100 + 50 =", banco.depositar(100, 50));  
console.log("Retirar 200 - 80 =", banco.retirar(200, 80));  
console.log("Transferir 500 -> 120 =", banco.transferir(500, 120));
```

//ESTUDIANTE

```
console.log("\n=== ESTUDIANTE ===");  
const estudiante = new Estudiante();  
console.log("Nuevo promedio =", estudiante.agregarNota(8, 10));  
console.log("¿Aprueba con 7 mínima? =", estudiante.aprobar(8, 7));  
console.log("Total de faltas =", estudiante.calcularFaltas(3, 2));
```

//AUTO

```
console.log("\n=== AUTO ===");  
const auto = new Auto();
```

```
console.log("Acelerar 40 + 20 =", auto.acelerar(40, 20));  
console.log("Frenar 50 - 70 =", auto.frenar(50, 70));  
console.log("Velocidad media =", auto.recorrer(120, 2));
```

```
//RESTAURANTE
```

```
console.log("\n=== RESTAURANTE ===");  
const restaurante = new Restaurante();  
const total = restaurante.calcularTotal(12, 3);  
console.log("Total sin descuento =", total);  
console.log("Total con descuento 10% =", restaurante.aplicarDescuento(total,  
10));  
console.log("Propina 15% =", restaurante.calcularPropina(total, 15));
```

```
//HOSPITAL
```

```
console.log("\n=== HOSPITAL ===");  
const hospital = new Hospital();  
console.log("Dosis para 70kg y 5mg/kg =", hospital.calcularDosis(70, 5));  
console.log("IMC de 70kg y 1.75m =", hospital.calcularIMC(70, 1.75));  
console.log("Visitas totales =", hospital.registrarVisitas(4, 2));
```

Código usado:

```
1 // CLASE BANCO
2
3 //Suma el monto al saldo actual
4 class Banco {
5     depositar(saldo: number, monto: number): number {
6         return saldo + monto;
7     }
8     //resta el monto al saldo actual si hay fondos suficientes
9     retirar(saldo: number, monto: number): number {
10        if (monto > saldo) {
11            throw new Error("Fondos insuficientes");
12        }
13        return saldo - monto;
14    }
15    //transferencia: resta del saldo de origen
16    transferir(saldoOrigen: number, monto: number): number {
17        if (monto > saldoOrigen) {
18            throw new Error("No se puede transferir más del saldo disponible");
19        }
20        return saldoOrigen - monto;
21    }
22 }
23
24 // CLASE ESTUDIANTE
25
26 //calcula nuevo promedio usndo dos notas
```

[LOG]: "=== BANCO ==="

[LOG]: "Depositar 100 + 50 =", 150

[LOG]: "Retirar 200 - 80 =", 120

[LOG]: "Transferir 500 -> 120 =", 380

[LOG]: "

=== ESTUDIANTE ==="

[LOG]: "Nuevo promedio =", 9

[LOG]: "¿Aprueba con 7 mínima? =", true

[LOG]: "Total de faltas =", 5

```
1 // CLASE BANCO
2
3 //Suma el monto al saldo actual
4 class Banco {
5     depositar(saldo: number, monto: number): number {
6         return saldo + monto;
7     }
8     //resta el monto al saldo actual si hay fondos suficientes
9     retirar(saldo: number, monto: number): number {
10        if (monto > saldo) {
11            throw new Error("Fondos insuficientes");
12        }
13        return saldo - monto;
14    }
15    //transferencia: resta del saldo de origen
16    transferir(saldoOrigen: number, monto: number): number {
17        if (monto > saldoOrigen) {
18            throw new Error("No se puede transferir más del saldo disponible");
19        }
20        return saldoOrigen - monto;
21    }
22 }
23
```

[LOG]: "

=== AUTO ==="

[LOG]: "Acelerar 40 + 20 =", 60

[LOG]: "Frenar 50 - 70 =", 0

[LOG]: "Velocidad media =", 60

[LOG]: "

=== RESTAURANTE ==="

[LOG]: "Total sin descuento =", 36

[LOG]: "Total con descuento 10% =", 32.4

```
1 // CLASE BANCO
2
3 //Suma el monto al saldo actual
4 class Banco {
5     depositar(saldo: number, monto: number): number {
6         return saldo + monto;
7     }
8     //resta el monto al saldo actual si hay fondos suficientes
9     retirar(saldo: number, monto: number): number {
10        if (monto > saldo) {
11            throw new Error("Fondos insuficientes");
12        }
13        return saldo - monto;
14    }
15    //transferencia: resta del saldo de origen
16    transferir(saldoOrigen: number, monto: number): number {
17        if (monto > saldoOrigen) {
18            throw new Error("No se puede transferir más del saldo disponible");
19        }
20        return saldoOrigen - monto;
21    }
22 }
23
24 // CLASE ESTUDIANTE
25
26 //calcula nuevo promedio usndo dos notas
```

[LOG]: "

=== RESTAURANTE ==="

[LOG]: "Total sin descuento =", 36

[LOG]: "Total con descuento 10% =", 32.4

[LOG]: "Propina 15% =", 5.3999999999999995

[LOG]: "

=== HOSPITAL ==="

[LOG]: "Dosis para 70kg y 5mg/kg =", 350

[LOG]: "IMC de 70kg y 1.75m =", 22.857142857142858

[LOG]: "Visitas totales =", 6

Conclusión personal:

Gracias a esta actividad aprendí a cómo usar clases en typescript y como es que las clases nos ayudan a organizar, modelar y reutilizar código de una manera eficiente.