Trabajo Constructores:

Java:

```
abstract class Animal {
String nombre;
int edad;

Public Animal (String nombre, int edad) {
this.nombre = nombre;
this.edad = edad;
}
abstract void informacion();

interface AccionesAnimal {
void sonido ();
}

public class Perro extends Animal implements
AccionesAnimal {
String raza;
double peso;

public perro (string nombre, int edad, String raza,
double peso) {
super (nombre, edad);
this.raza = raza;
this.peso = peso;
}

@Override
public void sonido () {
System.out.println ("Guau guau...");
}
@Override
void informacion () {
System.out.println (nombre);
System.out.println (edad);
System.out.println (raza);
System.out.println (peso);
}
```

```
Perro p = new Perro ("Max", 5, "Lobo Siberiano", 15);
p.informacion ();
p.sonido ();

abstract class Transporte {
String tipo;

public Transporte () {
this.tipo = "Desconocido";
}

abstract void informacion ();

interface Vehiculo {
void mover ();
}

public class Carro extends Transporte implements
Vehiculo {
String marca;
int modelo;

public Carro () {
this.tipo = "Carro";
this.marca = "Toyota";
this.modelo = 2020;
}

@Override
public void mover () {
System.out.println ("El carro está en movimiento.");
}

@Override
void informacion () {
System.out.println ("Tipo de transporte: " + tipo);
System.out.println ("Marca: " + marca);
System.out.println ("Modelo: " + modelo);
}

Carro car = new Carro ();
car.informacion ();
car.mover ();
```

```
interface AccionesPersona {
    void saludar ();
}

abstract class Persona {
    String nombre;
    String documento;

    public Persona (String nombre, String documento) {
        this.nombre = nombre;
        this.documento = documento;
    }

    public Persona (Persona otra) {
        this.nombre = otra.nombre;
        this.documento = otra.documento;
    }

    abstract void mostrarInformacion ();
}

public class Estudiante extends Persona implements
AccionesPersona {
    int edad;
    String curso;

    public Estudiante (String nombre, String documento, int
    edad, String curso) {
        super (nombre, documento);
        this.edad = edad;
        this.curso = curso;
    }

    public Estudiante (Estudiante otro) {
        super (otro);
        this.edad = otro.edad;
        this.curso = otro.curso;
    }

    @Override
    public void saludar () {
        System.out.println ("Hola soy" + nombre);
    }

    @Override
    void mostrarInformacion () {
        System.out.println ("Nombre: " + nombre);
        System.out.println ("Documento: " + documento);
        System.out.println ("edad: " + edad);
        System.out.println ("curso: " + curso);
    }
}
```

```
Estudiante estudiante1 = new Estudiante ("Carlos Pérez",
"1023456789", 20, "Programación");
System.out.println("Estudiante 1 (original)");
estudiante1.saludar();
estudiante1.mostrarInformacion();
System.out.println

Estudiante estudiante2 = new Estudiante(estudiante1);
System.out.println("Estudiante 2 (copia)");
estudiante2.saludar();
estudiante2.mostrarInformacion();
```

JS:

```
class AccionesPersona {
saludar() {
throw new Error("El método 'saludar()' debe ser
implementado por la clase hija.");
}

module.exports = AccionesPersona;

class Persona {
constructor(nombre, edad) {
this.nombre = nombre;
this.edad = edad;
}

static copiar(otraPersona) {
return new this(otraPersona.nombre, otraPersona.
edad);
}

mostrarInfo() {
throw new Error("El método 'mostrarInfo()' debe
ser implementado por la clase hija.");
}

module.exports = Persona;
const Persona = require("./Persona");
class Estudiante extends Persona {
constructor(nombre, edad, curso, promedio) {
super(nombre, edad);
this.curso = curso;
this.promedio = promedio;
}

static copiar(otroEstudiante) {
return new Estudiante(
otroEstudiante.nombre,
otroEstudiante.edad,
otroEstudiante.curso,
otroEstudiante.promedio
);
}
```

```
saludar(){
console.log(`Hola, soy ${this.nombre}`);
}

mostrarInfo(){
console.log(`Nombre: ${this.nombre}`);
console.log(`Edad: ${this.edad}`);
console.log(`Curso: ${this.curso}`);
console.log(`Promedio: ${this.promedio}`);
}

const e1 = new Estudiante("Johan", 20, "Programación", 4.5);
const e2 = Estudiante.copiar(e1);

e1.saludar();
e1.mostrarInfo();

console.log("COPIA");

e2.saludar();
e2.mostrarInfo();

module.exports = Estudiante;

class Dispositivo {
encender(){
throw new Error("El metodo encender() debe ser
implementado por la clase que herede.");
}

module.exports = Dispositivo;

const electronico {
constructor(){

this.marca = "Genérica";
this.año = 2024;
}

informacion(){
throw new Error("El metodo informacion() debe ser
implementado por la subclase!");
}

module.exports = Electronico;

class TelefonoInteligente extends Electronico {
constructor(modelo, almacenamiento) {
super();
this.modelo = modelo;
this.almacenamiento = almacenamiento;
}
```

```
encender () {
console.log("Encendido el teléfono...");
}

informacion () {
console.log("Marca:", this.marca);
console.log("Año:", this.año);
console.log("Modelo:", this.modelo);
console.log("Almacenamiento:", this.almacenamiento
+ "GB");
}
}
module.exports = TelefonoInteligente;
const t1 = new TelefonoInteligente("Galaxy A54", 128);
t1.encender();
t1.informacion();
```

C#

```
public string Titulo { get; set; }
public string Autor { get; set; }
public Libro(string titulo, string autor) {
Titulo = titulo;
Autor = autor;
}

public abstract void Informacion();
}

public interface AccionesLibro {
void Abrir();
}

public class Novela:Libro, AccionesLibro {
public int Paginas { get; set; }
public string Genero { get; set; }

public Novela(string titulo, string autor, int paginas,
string genero)
:base(titulo, autor)
{
Paginas = paginas;
Genero = genero;
}

public void Abrir(){
Console.WriteLine("Abriendo la novela...");
}
```

```
public override void Informacion (){
Console.WriteLine ($"Título: {Titulo}");
Console.WriteLine ($"Autor: {Autor}");
Console.WriteLine ($"Páginas: {Paginas}");
Console.WriteLine ($"Género: {Genero}");
}

Novela n = new Novela ("Cien años de soledad", "Gabriel
García Márquez", 38, "Literario");
n.Abrir ();
n.Informacion ();

public abstract class Transporte {
public string Tipo {get; set;}
public Transporte (){
Tipo = "Desconocido";
}
public abstract void Informacion ();
}

public interface Vehiculo {
void Mover ();
}

public class Carro : Transporte, Vehiculo {
public string Marca {get; set;}
public Carro (){
Tipo = "Carro";
Marca = "Toyota";
Modelo = 2020;
}

public void Mover () {
Console.WriteLine ("El carro está en movimiento.");
}

public override void Informacion (){
Console.WriteLine ($"Tipo de transporte: {Tipo}");
Console.WriteLine ($"Marca: {Marca}");
Console.WriteLine ($"Modelo: {Modelo}");

Carro carro = new Carro ();
carro.Mover ();
carro.Informacion ();

public interface AccionesPersona {
```

```
void Saludar ();

abstract class Persona{
public string Nombre;
public string Documento;
public Persona (string nombre, string documento) {
Nombre = nombre;
Documento = documento;

public Persona (Persona otra) {
Nombre = otra.Nombre;
Documento = otra.Documento;

public abstract void MostrarInformacion ();

class Estudiante : Persona, AccionesPersona {
public int Edad;
public string Curso;
public Estudiante (string nombre, string documento,
int edad, string curso)
: base (nombre, documento)
Edad = edad;
Curso = curso;

public void Saludar (){
Console.WriteLine("Hola, soy" + Nombre + ".");

public override void MostrarInformacion (){
Console.WriteLine("Nombre: " + Nombre);
Console.WriteLine("Documento: " + Documento);
Console.WriteLine("Edad: " + Edad);
Console.WriteLine("Curso: " + Curso);

Estudiante e1 = new Estudiante ("Johan Acero", "123456",
20, "Programacion");
Console.WriteLine("Estudiante Original");
e1.Saludar ();
e1.MostrarInformacion ();
```

```
Estudiante e2 = new Estudiante(e1);
Console.WriteLine("Estudiante Copiado");
e2.Saludar();
e2.MostrarInformacion();
}
```

Go

```
package main

type Device interface {
Encender()
Informacion()
}
```

```
package main

type Electronico struct {
Marca string
}

func NewElectronico(marca string) Electronico {
return Electronico {Marca: marca}
}
```

```
package main

import "fmt"

type Telefono struct {
 Electronico
Modelo string
Precio float64
}

func NewTelefono(modelo string) Telefono {
return Telefono {
Electronico: NewElectronico("Genérica"),
Modelo:  modelo,
Precio:  0,
}
}

func NewTelefonoFull(marca, modelo string, precio float64)
Telefono {
return Telefono {
Electronico: NewElectronico(marca),
Modelo:  modelo,
Precio:  precio,
}
}

func (t Telefono) Informacion() {
fmt.Println("Marca:", t.Marca)
```

```
fmt.Println("Modelo:", s.Modelo)
fmt.Println("Precio:", s.Precio)
}

fmt.Println("Ejemplo 3: Constructor sobrecargado")
cel := NewTelefonoFull("Samsung", "S24", 3500.0)
cel.Encender()
cel.Informacion()
```

```
package main

type Transporte struct {
Tipo string
}

func NewTransporte() Transporte {
return Transporte {Tipo: "Desconocido"}
}
```

```
package main

type Vehiculo interface {
Mover()
Informacion()
}
```

```
package main
import "fmt"

type Carro struct {
Transporte
Marca string
Modelo int
}

func NewCarro() carro {
return Carro {
Transporte: NewTransporte(),
Marca:  "Toyota",
Modelo:  2020,
}
}

func (c Carro) Mover() {
fmt.Println("El carro esta en movimiento.")
}

func (c Carro) Informacion() {
fmt.Println("Tipo:", c.Tipo)
fmt.Println("Marca:", c.Marca)
fmt.Println("Modelo:", c.Modelo)
}
```

```
Fmt.Println ("Ejemplo 1: Constructor sin Parametros")
Carro := NewCarro()
Carro.Mover()
Carro.Informacion()
```