**Encapsulación**

**Nombres**

Emily García

Angelo Morón

David Robledo

Santiago Hoyos

**Presentado a:**

Edison Sandoval

**ADSO-2711993**

**2024**

**Encapsulamiento**

**Introducción**

Mecanismo para reunir datos y métodos dentro de una estructura. Que nos permite ocultar los datos de cualquier medio que no sean los servicios propuestos.

**Getters y Setters**

Los getter y setter son métodos especiales que se utilizan para acceder y modificar los valores de las propiedades (en si las propiedades privadas) de un objeto de una clase.

**Getters**: obtienen el valor de la propiedad. Para definir un getter en dart se usa la palabra clave get:

**Ejemplo:**

String get nombre {

Return \_nombre;

}

String=> define el valor a devolver, no es obligatorio al manejarlo

Get => define el getter en dart

**Setters:** establecen o modifican los valores de la propiedad. El setter se declara con la palabra clave set

**Ejemplo:**

Set nombre (String valor) {

\_nombre=valor;

}

Set=> define el setter

(String valor) => Valor a ingresar Para llamar los getter y setter de una clase al principal se maneja de este modo:

**Setter**

persona.nombre="Juan";

**Getter**

Print(persona.nombre) Propiedades privadas se usan dentro de las clases para un manejo solo interno de ellas

**Cómo definirlas:**

String \_nombre;

String UNDERSCORE nombre;

Hay que tener en cuenta que todas las propiedades por defecto son públicas.

**Parámetros por valor y por referencia**

**Por valor:** Se reciben y envían datos primitivos (int, double, bool, float, char, String).

Los tipos primitivos en Dart son inmutables. Cualquier cambio en el valor resulta en la creación de un nuevo valor en la memoria.

Los valores se copian y las modificaciones dentro de la función no afectan las variables originales.

**Por referencia:** Se reciben y envían objetos (instancias de clases).

Los objetos pueden ser mutables o inmutables. Los objetos mutables permiten que sus propiedades sean cambiadas sin crear un nuevo objeto.

Se pasa una referencia a la función, y las modificaciones dentro de la función afectan al objeto original.

**Ejemplos:**

**Por valor:** void incrementarPorValor(int numero) {

numero++;

print('Dentro de la función: $numero'); // Salida: 11

}

void main() {

int miNumero = 10;

incrementarPorValor(miNumero);

print('Fuera de la función: $miNumero'); // Salida: 10

}

**Por referencia**: class Contador {

int valor = 0;

}

void incrementarPorReferencia(Contador contador) {

contador.valor++;

print('Dentro de la función: ${contador.valor}'); // Salida: 1

}

void main() {

Contador miContador = Contador();

incrementarPorReferencia(miContador);

print ('Fuera de la función: ${miContador.valor}'); // Salida: 1}