

# (Más) operadores

Clases y Patrones de diseño



Yosafat Coronel  
GDSC ESCOM IPN  
GitHub: YosafatM



Developer Student Clubs  
ESCOM - IPN

# Itinerario

- Operadores de acceso
  - Miembro
  - Suscriptibles
  - Guión bajo
- Operadores para nulidad
  - Asignación
  - Lectura
- MiCard



Acceso

Condicionales

De miembro

Suscriptibles



# Punto

## <op>

```
_id = '1';
```

Como vimos, podemos acceder a las propiedades de los objetos que hacemos, pero a veces, no queremos que eso se pueda.

*Algunas cosas sería mejor mantenerlas ocultas, como la información que sirve como llave en nuestro sistema. Más adelante se explica el guión bajo.*

# Punto

## <op>

El operador de punto sirve para acceder a propiedades/métodos de una expresión, Dart le dará acceso a todo lo que no comience con guión bajo\*:

*dogtor.dart*

```
class Dogtor {  
  final _id = 1;  
  final saludo = 'Hola';  
}
```

*main.dart*

```
void main() {  
  var doc = Dogtor();  
  print(doc._id);  
  print(doc.saludo);  
}
```

Otros

<op>

[]

Suscriptible

.

Miembro

()

Llamada

De arriba hacia abajo, retorna un valor, se puede sobreescribir, permite acceder a las propiedades de una expresión, pone una llamada en el stack y ejecuta la función a la que referencia.

# Nota\*

\_<id>

```
class Dogtor {  
    final _id = 1;  
    final saludo = 'Hola';  
}
```

*Usa archivos para mantener el  
código organizado*

Para decirle a Dart que una propiedad solo puede ser accedida en el marco de trabajo del archivo, entonces le ponemos guión bajo al identificador. **NO, NO EXISTEN LAS PALABRAS PUBLIC O PRIVATE.**

# Nota\*

\_<id>

Sin embargo, si el acceso está en el mismo archivo (marco) no marca ningún error:

```
class Dogtor {  
    final _id = 1;  
    final saludo = 'Hola';  
}  
  
void main() {  
    var doc = Dogtor();  
    print(doc._id);  
    print(doc.saludo);  
}
```

# Nulidad

## Acceso

## Lectura

## Asignación



# Acceso

?[]

# Suscriptible

?.

# Miembro

Hacen lo que los originales, pero **no da error si la expresión sobre la que lo hacen es null**. Si lo es, entonces retorna null, si no se utiliza ese null, simplemente es ignorado.

# Asignar

# Leer

# ?? = Asignación

# ?? Lectura

```
void main() {  
    int? a, b;  
    a ??= 5;  
    a ??= 3;  
    print(a);  
    print(b ?? 'Es null');  
}
```

Dada una expresión, si es nula, entonces **asignan** o **retornan** (respectivamente) el valor a la derecha:

*Salida: 5 y 'Es null'*

# Ejemplo

*qt significa quantity*

```
qt = ctr?.cart?.length ?? 0;
```

Algo común en Flutter y Dart es que los operadores se usan mucho en una sola línea, haciendo mucho más expresivo el código. Aquí, preguntamos si cada parte es null, eso retorna null, y con ??, regresamos 0.

# MiCard

[github.com/YosafatM/UFlutter-mi-card](https://github.com/YosafatM/UFlutter-mi-card)



# Crédito



MiCard es una de las 12 apps del curso de la Dr. Angela Yu (por parte de AppBrewery) en colaboración con Google para el curso de Flutter de 2020 (aunque recibe actualizaciones, Flutter cambia más rápido).

# ¡Muchísimas gracias!



Developer Student Clubs  
ESCOM - IPN

