

DIEGO,
FERNÁNDEZ



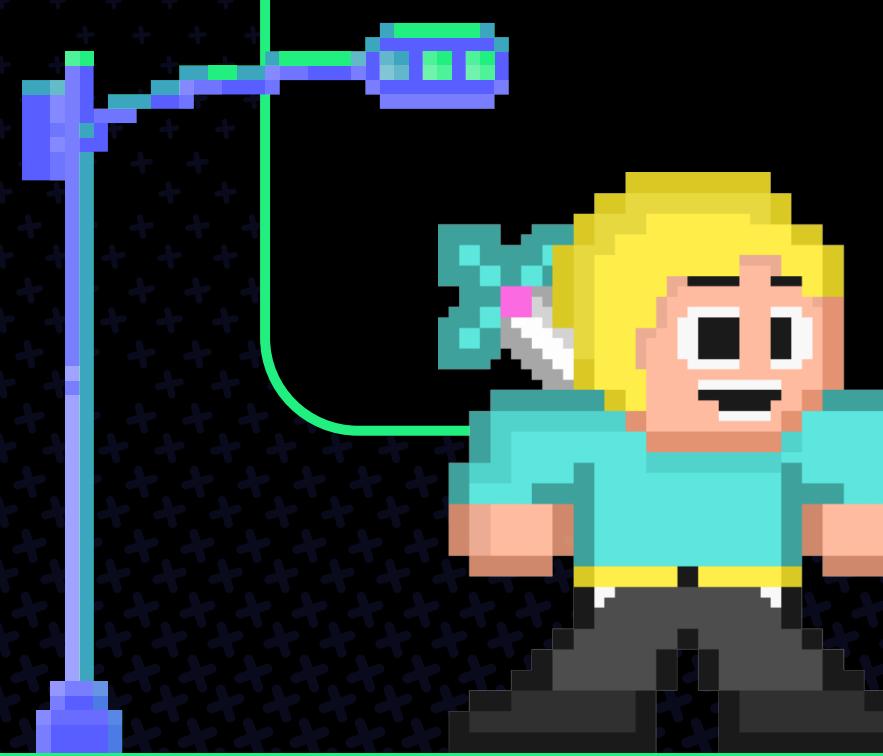
SERGI ESCRIVA

ABSTRACCIÓN ENCAPSULACIÓN OCULTACIÓN

START

MENU

SIGN IN



MENU

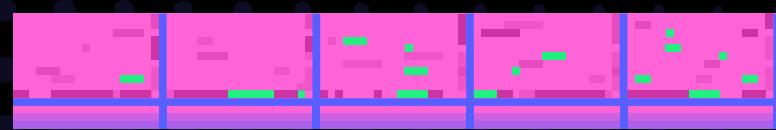
➡ 33

⚪ 33

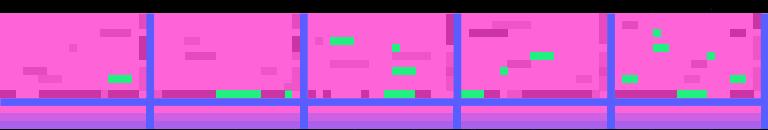
★ 33



INDICE



ABSTRACCION



ENCAPSULACION



OCULTACION

MENU

➔ 33

◆ 33

★ 33



ABSTRACCIÓN

LA ABSTRACCIÓN ES EL PROCESO DE **SIMPLIFICAR Y REPRESENTAR** UN SISTEMA COMPLEJO DE MANERA CONCEPTUAL. EN PROGRAMACIÓN ORIENTADA A OBJETOS, LA ABSTRACCIÓN IMPLICA IDENTIFICAR LAS **CARACTERÍSTICAS ESENCIALES** DE UN OBJETO Y REPRESENTAR SOLO LA **INFORMACIÓN RELEVANTE**, IGNORANDO LOS **DETALLES NO ESENCIALES**.

```
abstract class Vehiculo {  
    1 usage  
    private String modelo;  
    1 usage  
    private String marca;  
  
    1 usage new *  
    public Vehiculo(String modelo, String marca) {  
        this.modelo = modelo;  
        this.marca = marca;  
    }  
  
    no usages new *  
    public abstract void iniciar();  
  
    no usages new *  
    public abstract void detener();  
  
    no usages new *  
    public abstract void obtenerInformacion();  
}  
  
no usages new *  
class Automovil extends Vehiculo{  
    no usages new *  
    public Automovil(String modelo, String marca) {  
        super(modelo, marca);  
    }  
}
```

CLASES ABSTRACTAS E INTERFACES

INTERFAZ - PALABRA CLAVE
"IMPLEMENTS"

```
J Main.java > ↗ Main
1  interface VehiculoAMotor {
2
3      public void arrancar();
4  }
5
6  interface VehiculoTerrestre {
7
8      public void virar();
9  }
10
11 class Coche implements VehiculoAMotor, VehiculoTerrestre {
12
13     public void arrancar() {
14         System.out.println("Accionar el contacto.");
15     }
16
17     public void virar() {
18         System.out.println("Manejar el volante.");
19     }
20 }
21
22 public class Main {
23
24     Run | Debug
25     public static void main(String[] args) {
26
27         Coche lanciaStratos = new Coche();
28         lanciaStratos.arrancar();
29         lanciaStratos.virar();
30     }
31 }
```

CLASE ABSTRACTA - PALABRA CLAVE "INHERITS"

```
J Main.java > ↗ Main > ↗ main(String[])
1  abstract class Coche {
2
3      public abstract void arrancar();
4
5      public void acelerar() {
6          System.out.println("Vroom!");
7      }
8
9
10 class CocheElectrico extends Coche {
11
12     public void arrancar() {
13
14         System.out.println("Para arrancar, presione el botón de arranque.");
15     }
16
17 class CocheDiesel extends Coche {
18
19     public void arrancar() {
20
21         System.out.println("Para arrancar, gire la llave de contacto.");
22     }
23
24
25
26 public class Main {
27
28     Run | Debug
29     public static void main(String[] args) {
30
31         CocheElectrico ioniq5 = new CocheElectrico();
32         CocheDiesel freemander99 = new CocheDiesel();
33         ioniq5.arrancar();
34         ioniq5.acelerar();
35         freemander99.arrancar();
36         freemander99.acelerar();
37     }
38 }
```

MENU



MODIFICADORES DE ACCESO

PRIVATE

PROTECTED

PUBLIC

[VUELVE A LA
PÁGINA AGENDA](#)

- ◆ PARA CLASES, MÉTODOS Y CONSTRUCTORES

El código o clase es accesible desde cualquier otra clase.

- ◆ PARA MÉTODOS Y CONSTRUCTORES

El código es accesible sólo dentro de la misma clase.

- ◆ PARA MÉTODOS Y CONSTRUCTORES

El código es accesible desde el paquete en el que se encuentra, y por sus subclases.



◆ 33



MENU



ENCAPSULACIÓN

LA ENCAPSULACIÓN TIENE DOS ACEPCIONES VÁLIDAS:

- ◆ LA PRÁCTICA POR LA QUE SE SEPARAN DEL RESTO DEL CÓDIGO LOS ATRIBUTOS Y FUNCIONALIDADES ESPECÍFICAS DE LOS OBJETOS
- ◆ LA PRÁCTICA POR LA CUAL SE TRATA DE ASEGURAR CIERTOS DATOS, E IMPEDIR SU ACCESO INDISCRIMINADO POR PARTE DEL USUARIO. PARA ELLO SE USAN LOS MODIFICADORES DE ACCESO Y LOS MÉTODOS SETTER Y GETTER (MÉTODOS DE ESCRITURA Y DE LECTURA)



J Canyon.java > Canyon

```
1  public class Canyon {
2      private String modelo;
3      private int calibre;
4      private boolean cargado;
5
6      public Canyon(String modelo, int calibre, boolean cargado) {
7          this.modelo = modelo;
8          this.calibre = calibre;
9          this.cargado = cargado;
10     }
11 }
```

EJEMPLO

J AbrirFuego.java > AbrirFuego > main(String[])

```
1  public class AbrirFuego {
2
3      Run | Debug
4      public static void main(String[] args) {
5
6          Canyon canyonT34 = new Canyon(modelo:"L-11", calibre:76, cargado:true);
7
8          if (canyonT34.cargado) {
9              System.out.println("El cañón " + canyonT34.modelo + " de "
10                  + canyonT34.calibre + "mm está listo para disparar.");
11
12      }
13 }
```



GET

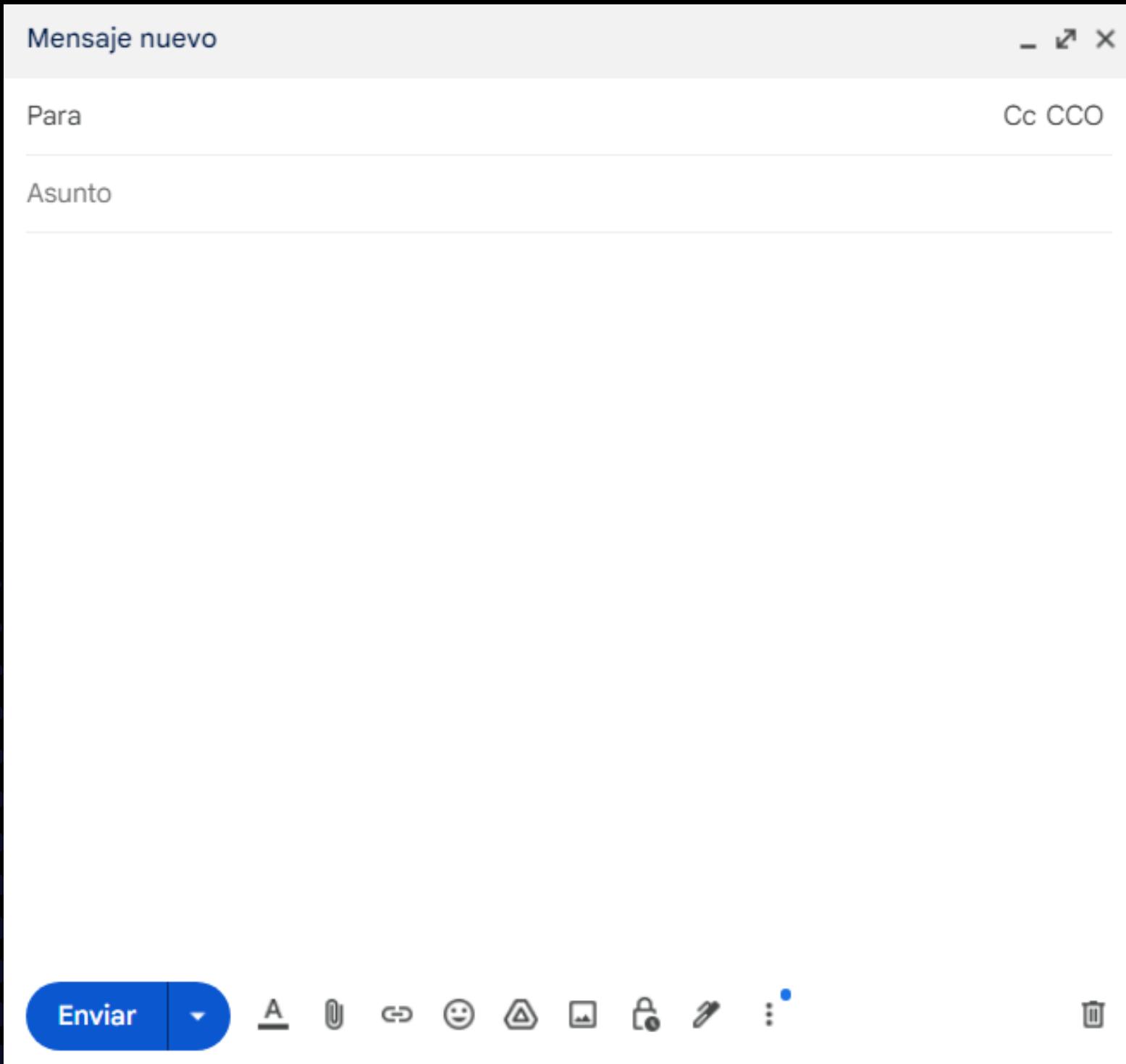
SET

LOS MÉTODOS GET Y SET, SON **SIMPLES MÉTODOS** QUE USAMOS EN LAS CLASES PARA MOSTRAR (GET) O MODIFICAR (SET) EL VALOR DE UN ATRIBUTO **PROTEGIDO**. EL NOMBRE DEL MÉTODO SIEMPRE SERÁ GET O SET Y A CONTINUACIÓN EL NOMBRE DEL ATRIBUTO, SU MODIFICADOR SIEMPRE ES PUBLIC YA QUE QUEREMOS MOSTRAR O MODIFICAR DESDE FUERA LA CLASE. POR EJEMPLO, **GETNOMBRE** O **SETNOMBRE**.

```
public String getNombre() {  
    return nombre;  
}  
  
no usages new *  
public void setNombre(String nombre) {  
    this.nombre = nombre;  
}
```



MENU



OCULTACION

LA OCULTACION SE REFIERE AL CONCEPTO
DE OCULTAR DETALLES DE
IMPLEMENTACION DE UNA CLASE Y
EXPONER SOLO LO NECESARIO PARA LA
INTERACCION DE LA CLASE



MENU

EL NANO



¡MUCHAS GRACIAS!

