AWAKELAB

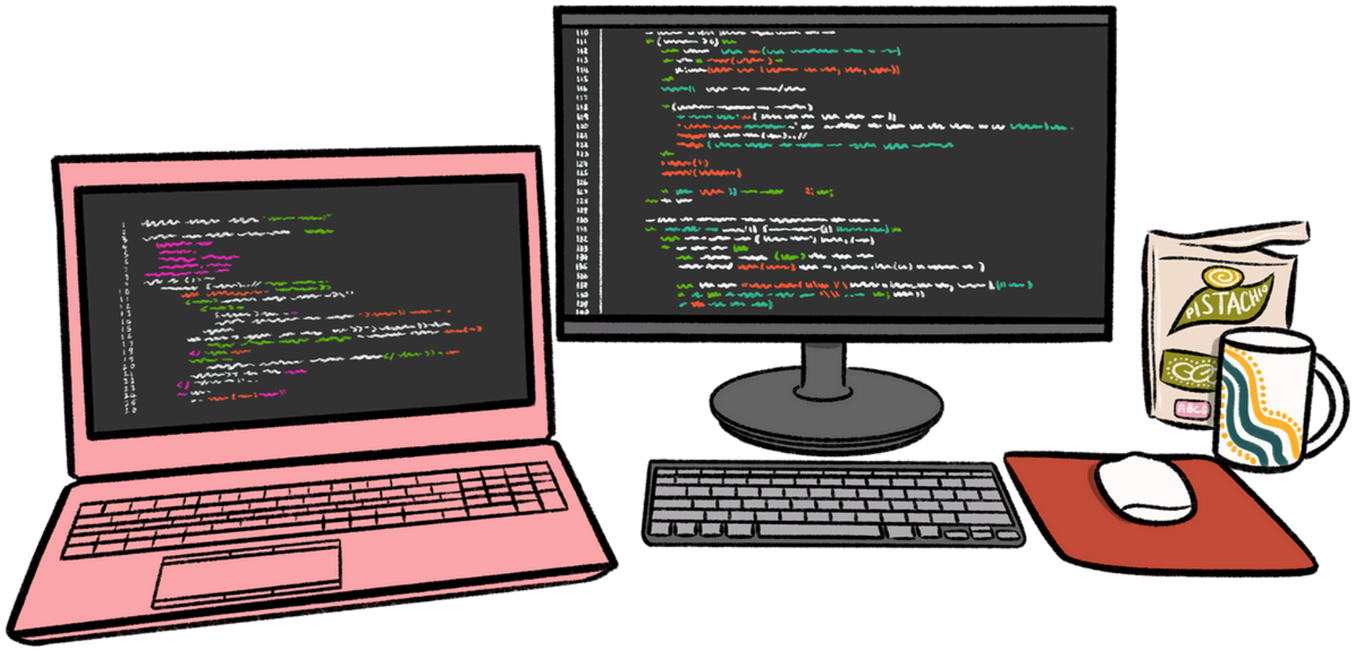
#programmingbootcamp

EL PARADIGMA DE ORIENTACIÓN A OBJETO

EL PARADIGMA DE ORIENTACIÓN A OBJETO

¿QUÉ VAMOS A VER?

- POO.
- Clases y Objetos.
- Modificadores de acceso.
- Accesadores y Mutadores.
- Colaboración entre objetos.



PROGRAMACIÓN ORIENTADA A OBJETOS

- Es un paradigma de programación basado en objetos.

- Cada uno de los objetos manipula los datos de entrada para retornar resultados específicos.

- Los objetos pueden interactuar entre sí mediante procedimientos definidos (métodos) y evolucionan su información interna (propiedades) en función de su finalidad.

PROGRAMACIÓN ORIENTADA A OBJETOS

- **Modificable:** Es sencillo añadir, modificar o eliminar nuevos objetos.

- **Gestión de los errores:** Al ser un trabajo modular, se pueden dividir los errores en secciones más pequeñas y así es más fácil aislarlos para poder resolverlos.

- **Trabajo en grupo:** Lo facilita gracias a minimizar la duplicación de funciones.

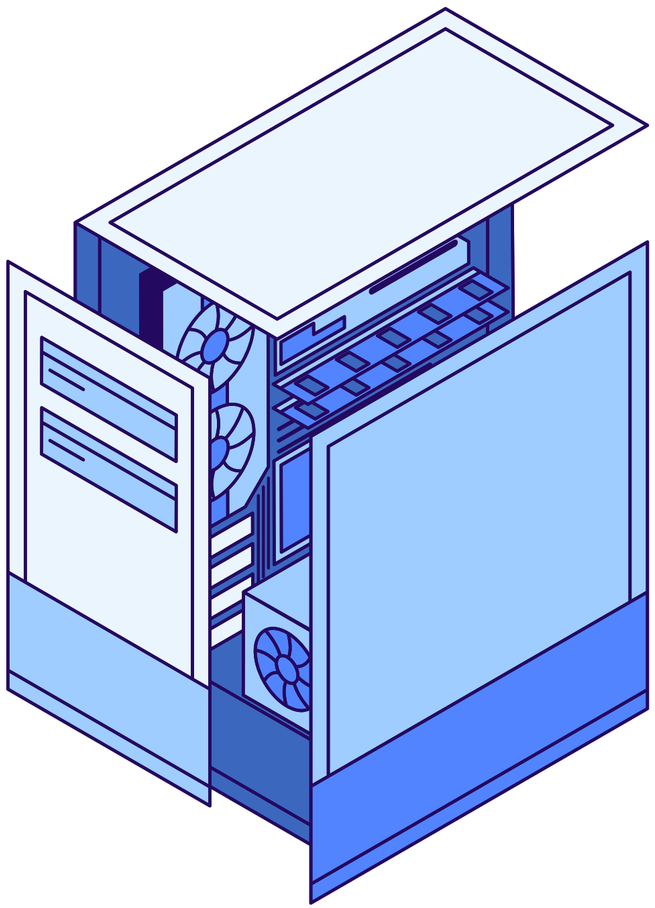
- **Reducción de costes de programación:** Si una funcionalidad ya fue creada, se ahorra tiempo y recursos al ser mucho más fácil reutilizarla.

CLASES Y OBJETOS

- Los objetos son programas que tienen un estado y un comportamiento, conteniendo datos almacenados y tareas realizables durante su ejecución.

- Las clases representan un conjunto de variables y métodos para operar con datos. Definen la forma de un objeto a través de la especificación de datos y código que opera sobre ellos.

- En Java los objetos son instancias de una clase, por ende, una clase es esencialmente un conjunto de procesos que especifican como construir un objeto.



CLASES Y OBJETOS

- En un juego, una clase puede recibir el nombre de “Persona” y tener una serie de atributos como nombre, apellidos o edad, y una serie de comportamientos como hablar(), caminar() o comer(), que vendrían a ser métodos (o funciones) de la misma clase.

ATRIBUTOS DE UNA CLASE Y ESTADO DE UN OBJETO

- Los atributos son valores o características de los objetos. Permiten definir el estado del objeto u otras cualidades.

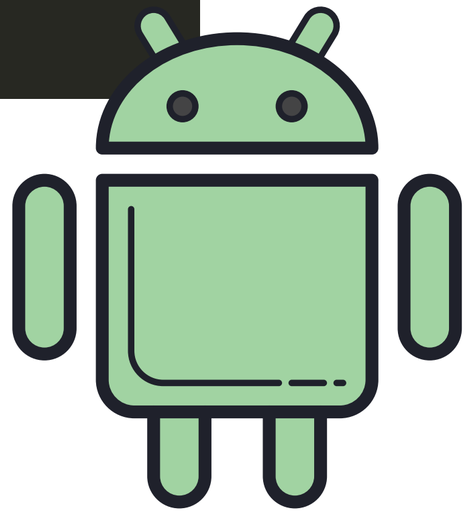
- En el ejemplo podemos ver la clase Auto con sus atributos de marca, color y velocidad.

- Recordar que una clase siempre es una simplificación de la realidad. Es por eso que debemos centrarnos en los atributos que necesitamos para la problemática del momento.

```
class Auto {

    String marca;
    String color;
    double velocidad;

}
```



MÉTODOS DE UNA CLASE Y COMPORTAMIENTO DE UN OBJETO

- Un método es una subrutina que puede pertenecer a una clase u objeto. En resumen, son sentencias para llevar a cabo una acción.

- Cualquier instancia de una clase puede acceder a los métodos que son parte de ella.

```
class Auto {

    String marca;
    String color;
    double velocidad;

    void acelerar() {
        velocidad++;
    }

    void frenar() {
        velocidad = 0;
    }

}
```

MÉTODOS CONSTRUCTORES

- Existe un método que se ejecuta automáticamente cuando se instancia un objeto de una clase.
- Este método Constructor, tiene por objetivo asignar los valores iniciales del objeto recién creado.
- Se utiliza **this** para distinguir entre el atributo de la clase y el parámetro de la función.
- **this.marca** hace referencia al atributo "marca" del objeto actual **(this)**, mientras que marca hace referencia al parámetro de la función.

```
class Auto {

    String marca;
    String color;
    double velocidad;

    Auto() { }

    Auto(String marca, String color, double velocidad) {
        this.marca = marca;
        this.color = color;
        this.velocidad = velocidad;
    }

    void acelerar() {
        velocidad++;
    }

    void frenar() {
        velocidad = 0;
    }

}
```

MODIFICADORES DE ACCESO

- Determinan desde qué clases se puede acceder a un determinado elemento.

- En caso de no especificar ningún modificador, los elementos “por defecto” solo pueden ser accedidos desde las clases que pertenezcan a su mismo paquete.

- **public:** Permite acceder al elemento desde cualquier clase, independientemente de que esta pertenezca o no al paquete del elemento.

- **private:** Especifica que los elementos solo pueden ser utilizados dentro de la clase en la que se encuentran.

MODIFICADORES DE ACCESO

- **protected:** Solo pueden ser accedidos desde su mismo paquete y por cualquier clase que extienda la clase en la que se encuentra, independientemente de si están en el mismo paquete o no.

MODIFICADORES DE ACCESO				
Modificador	**La misma clase**	**Otra clase del mismo paquete**	**Subclase de otro paquete**	**Otra clase de otro paquete**
default	X	X		
public	X	X	X	X
private	X			
protected	X	X	X	

ACCESADORES Y MUTADORES

- En conjunto nos van a permitir acceder a los atributos que podamos tener dentro de una clase, sobre todo cuando se encuentran con un acceso privado.

- Su papel es devolver y establecer los valores del estado de un objeto.

- **¿No es más simple poner los atributos como públicos?** En esencia, *sí*, pero no debemos olvidar que se busca mantener un grado de seguridad en nuestro software, por lo tanto, no es óptimo ir revelando los datos de un objeto.

ACCESADORES Y MUTADORES

ACCESADOR

- Se utiliza para devolver el valor de un campo privado.
- Lo identificamos por la palabra “get” al principio del nombre del método.
- Siempre retornan el mismo tipo de dato que corresponde al campo privado.

```
public String getMarca() {
    return this.marca;
}

public String getColor() {
    return this.color;
}

public double getVelocidad() {
    return this.velocidad;
}
```

ACCESADORES Y MUTADORES

MUTADOR

- Se utiliza para establecer el valor de un campo privado.
- Lo identificamos por la palabra “set” al inicio del nombre del método.
- Los mutadores no retornan ningún dato, es más, reciben un atributo que es el mismo tipo de dato que su campo privado.

```
public void setMarca(String marca) {
    this.marca = marca;
}

public void setColor(String color) {
    this.color = color;
}

public void setVelocidad(double velocidad) {
    this.velocidad = velocidad;
}
```

COLABORACIÓN ENTRE OBJETOS

- Es común en POO utilizar más de una clase para solucionar las problemáticas presentadas.

- En Java, las relaciones entre los objetos pueden ser persistentes a lo largo del tiempo, mientras la comunicación sea continua y pueda ser utilizada en cualquier momento.

- En caso de no tener una relación persistente, el vínculo entre dos objetos desaparece tras ser empleado.

COLABORACIÓN ENTRE OBJETOS

ASOCIACIÓN

- En una asociación, dos instancias relacionadas entre sí existen de forma independiente.
- La creación o desaparición de una de ellas, no afecta la integridad de la otra.
- Por ejemplo, un cliente puede tener varios pedidos de compra o ninguno.

COLABORACIÓN ENTRE OBJETOS

TODO-PARTE

- Una instancia forma parte de la otra, es decir, A está compuesto de B o A tiene a B.

- **Agregación:** Es una asociación binaria que responde a “pertenece a, tiene un, es parte de”. Por ejemplo, un centro comercial tiene clientes.

- SI bien las dos instancias no están relacionadas desde su creación, si es posible eliminar a ambas al momento de cortar la relación.

COLABORACIÓN ENTRE OBJETOS

TODO-PARTE

- **Composición:** En este caso, los componentes constituyen un parte importante del objeto compuesto.

- Es en este tipo de Todo-Parte en dónde la supresión de un objeto conlleva la eliminación total de sus componentes.

- Ejemplos, el auto tiene llantas o el notebook tiene un teclado.

AWAKELAB

#programmingbootcamp