TALENTO DIGITAL
INTELIGENCIA HUMANA

Talento Digital para Chile:

MÓDULO 1
PROGRAMACIÓN BÁSICA EN JAVA

UN PROYECTO DE:

Ministerio de Hacienda · Ministerio del Trabajo y Previsión Social · CORFO · sence · SOFOFA · OTIC SOFOFA Capital Humano · CPC · ACTI · BID · FCH Fundación Chile · kodea

DESARROLLADO POR:

MÓDULO 1 - PROGRAMACIÓN BÁSICA EN JAVA

1.3 EL PARADIGMA DE ORIENTACIÓN A OBJETOS

Semana 3 - Día 13

Objetivo de la jornada

- Conocer el concepto y la importancia de la Programación Orientada a Objetos para estructurar soluciones según el paradigma más utilizado en la industria del Software.
- Entender los conceptos claves de la POO: Clases y objetos, con el objetivo de desarrollar el pensamiento orientado a Objetos..
- Comprender los conceptos de Atributos y Métodos de una Clase para luego utilizarlos para resolver problemas planteados.
- Comprender y aplicar Métodos Constructores para dar solución a un problema según requerimientos establecidos.

Importancia de la Programación Orientada a Objetos

La programación orientada a objetos (POO) es un paradigma de programación basado en objetos, los cuales manipulan los datos de entrada para la obtención de datos de salida específicos, y donde cada objeto ofrece una funcionalidad especial.

Estos objetos interactúan entre sí mediante procedimientos definidos (métodos) y van evolucionando su información interna (propiedades) en función de su historia.

Si se hace una analogía, los objetos se parecen mucho a cómo funciona una empresa. Cada uno de los departamentos internos de la empresa como: ventas, administración, personal o producción, son objetos abstractos a los que el director solicita información (datos) o procesos (sacar las nóminas) sin que este conozca muy bien cuáles son los procedimientos internos a realizar.

El director tiene una imagen de la responsabilidad de cada sección (objeto) y de los procedimientos que cada objeto puede realizar, o de la información que le puede conseguir, y por eso el reporte de ventas por producto, sabe que corresponde a Ventas y no a nóminas, pero no se le ocurre pedir los datos concretos, sino solo el resumen procesado de acuerdo a ciertos criterios establecidos.

Además vale la pena destacar, que cada departamento de la empresa dispone de sus propias datos internos privados, que utiliza para cumplir sus procedimientos, es decir, que al igual que la OOP mezcla datos y procedimientos en el modelo de trabajo.
Son muchas las ventajas de una Programación Orientada a Objetos, pero las más relevantes son:

- Modificabilidad: en la POO es sencillo añadir, modificar o eliminar nuevos objetos o funciones que nos permiten actualizar programas fácilmente.

- Gestión de los errores: cuando se trabaja con un lenguaje POO se sabe exactamente dónde mirar cuando se produce un error, ventaja del trabajo modular de los lenguajes POO. Al poder dividir los problemas en partes más pequeñas se pueden probar de manera independiente y aislar los errores que puedan producirse en el futuro.

- Trabajo en grupo: es más fácil trabajar en grupo ya que la POO permite minimizar la posibilidad de duplicar funciones cuando varias personas trabajan sobre un mismo objeto al mismo tiempo.

- Reducción de costes de programación: especialmente en proyectos grandes la POO reduce los costos de programación ya que se los programadores pueden usar el trabajo de los otros, ahorrando horas de desarrollo. Crear librerías y compartirlas o reutilizar librerías de otros proyectos es algo habitual en la programación orientada a objetos.

Objetos y clases

En el paradigma de Programación Orientada a Objetos, los objetos son programas que tienen un estado y un comportamiento, conteniendo datos almacenados y tareas realizables durante su ejecución. Algunos ejemplos de objetos son: persona, silla, mesa, casa y auto.

Las clases son un pilar fundamental de la POO y representan un conjunto de variables y métodos para operar con datos. Una clase es una plantilla que define la forma de un objeto, especifica los datos y el código que operará en esos datos. Java usa una especificación de clase para construir objetos. Los objetos son instancias de una clase. Por lo tanto, una clase es esencialmente un conjunto de planes que especifican cómo construir un objeto.

Una clase es una plantilla. Define de manera genérica cómo van a ser los objetos de determinado tipo. Por ejemplo, en un juego, una clase para representar a personas puede llamarse Persona y tener una serie de atributos como Nombre, Apellidos o Edad (que normalmente son propiedades), y una serie de comportamientos que pueden tener, como Hablar(), Caminar() o Comer() y que se implementan como métodos de la clase (funciones).

Si se toma de ejemplo de la clase Auto, en código Java quedaría de la siguiente manera:

class Auto { } Como se puede observar, el nombre de la clase va precedido por la palabra reservada class. Además, se incluyen llaves de inicio de bloque de la clase y otro de término, entendiendo que todo lo que se encuentre dentro de esas llaves, será parte de la clase Auto.

Atributos de una clase y estado de un objeto

Una vez entendido lo que son las clases y objetos, se podrá continuar agregando propiedades que disponen los objetos que definen sus características individuales y le permiten diferenciarse de otros.

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

Retomando el ejemplo anterior de la clase Auto, se pueden reconocer algunos de sus posibles atributos: marca, color, capacidad de combustible, potencia del motor, velocidad, etc. Al llevar algunos de estos atributos a código Java quedaría de la siguiente manera:

class Auto {

String marca; String color; float velocidad; } De esta forma, es posible crear la estructura de la clase con los atributos que corresponden y que permitirán diferenciar objetos.

Cabe recordar que una clase es siempre una simplificación de la realidad. Por esta razón, nunca se deben declarar atributos para todas y cada una de las características del objeto que queramos representar; sólo crear aquellos que se necesiten para resolver el problema planteado. Por ejemplo, si se debe crear una clase Auto para guardar datos como marca, color y velocidad, no es necesario agregar atributos como: número de ruedas, modelo, capacidad de pasajeros o sistema de freno.

Métodos de una clase y comportamiento de un objeto

Un método es una subrutina que puede pertenecer a una clase u objeto, y son una serie de sentencias para llevar a cabo una acción.

El objeto auto tiene los métodos: acelerar, frenar, indicar izquierda, indicar derecha, etc.

Llevando a código Java algunos de los métodos señalados, se vería de la siguiente forma:

```
class Auto {
    String marca;
    String color; float
    velocidad;

        void acelerar() {
    velocidad++; } void
                frenar() {
    velocidad = 0; } } Para ambos ejemplos de métodos de la clase Auto, se modifica el
```

atributo velocidad. Esto quiere decir que cualquier instancia Auto podrá utilizar estos métodos.

Métodos constructores

Existe un método que se ejecuta automáticamente cuando se instancia un objeto de una clase. Este método se llama Constructor, el cual tiene por objetivo asignar los valores iniciales del nuevo objeto recién creado.

Un constructor de la clase Auto se vería de la siguiente forma en código Java:

```
class Auto {
    String marca;
    String color; float
    velocidad;

        void acelerar() {
    velocidad++; } void
                frenar() {
    velocidad = 0; } Auto() { } }
```

Por ser métodos, los constructores también aceptan parámetros. Cuando en una clase no se especifica ningún tipo de constructor, el compilador añade uno por omisión sin parámetros, el cual NO hace nada por lo que los valores de los atributos de ese objeto tomarán los valores por defecto o con basura, según se haya escrito el código.

El mismo ejemplo de la clase Auto pero sumando un constructor con parámetros sería:

```
class Auto {
    String marca;
    String color; float
    velocidad;

        void acelerar() {
    velocidad++; } void
                frenar() {
    velocidad = 0; }
    Auto() { } Auto(String marca, String color, float
    velocidad) {
this.marca = marca; this.color = color; this.velocidad = velocidad; } }
```

Se debe tomar en consideración que los nombres de los parámetros de entrada NO necesariamente deben tener el mismo nombre de los atributos de la clase. Hay algunas convenciones que sugieren utilizar el mismo nombre, como también otras que prefieren usar un guión bajo antes del nombre del atributo de la clase.

Cuando una función accede a un atributo lo hace en el contexto de un objeto. Por eso no es necesario que se especifique a qué objeto se está haciendo referencia. Sin embargo, hay casos en los que puede ser interesante disponer de dicho objeto, por ejemplo para pasarlo por parámetro a otra función o para distinguirlo de una variable o un parámetro homónimo.

Las funciones miembro pueden acceder al objeto actual mediante el objeto predefinido **this**
.
En el caso del ejemplo de la clase Auto, se utiliza this para distinguir entre el atributo de la clase y el parámetro de la función. El “this.marca” hace referencia al atributo “marca” del objeto actual (this), mientras que “marca”, sin el this, hace referencia al parámetro de la función.