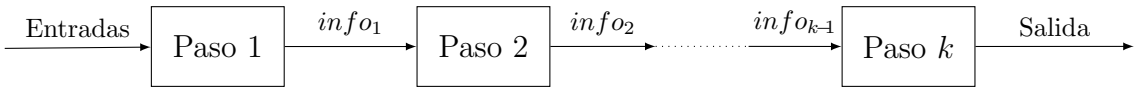
Módulo 1 Lección 2

¿Qué es la orientación a objetos (OO)?

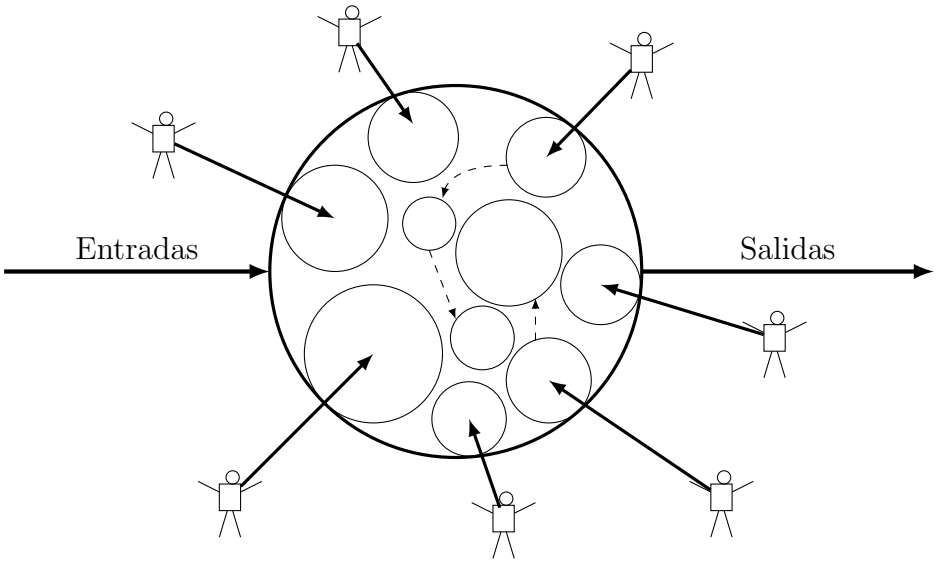
Mencionamos en la sesión anterior que los lenguajes de programación están diseñados para diferentes objetivos y que, en particular, Java es un lenguaje orientado a objetos. Pero ¿qué quiere decir *orientación a objetos*?

La *orientación a objetos* corresponde a una manera de ver un problema dado y los pasos que se toman para resolverlo. Aunque no importa qué metodología uses para analizar un problema, un principio que siempre usarás es el de ***divide y vencerás***. Esto quiere decir que al problema buscarás dividirlo en problemas más pequeños, que a su vez podrás dividir, hasta que los problemas a resolver sean sencillos e inmediatos.

En la llamada programación estructurada o imperativa, como Pascal, C, Cobol, divides los problemas de acuerdo al suceder del tiempo, lo que va primero, lo que sigue, etc., viendo a cada resultado de un subproceso como la entrada al siguiente:



En la orientación a objetos se divide el problema entre actores y sus respectivas responsabilidades en la solución del mismo y todos los actores, cada uno con lo que sabe hacer, contribuyen a su solución:



La colaboración, por supuesto, debe ser organizada y ordenada. Si pensamos en un concierto que ofrece una orquesta, cada músico afina su instrumento y a lo mejor ensaya en su casa, pero hasta que no tocan *armoniosamente*, dirigidos por el director de la orquesta, lo que se oye (obtiene) no es la música del concierto, sino una colección de sonidos que no tienen, en ese momento, relación entre sí.

Usemos la analogía de un concierto ofrecido por una orquesta. Cada persona que toca un instrumento (o instrumento, para abreviar) pertenece a un grupo: piano, instrumentos de cuerdas, instrumentos de viento, de percusión, que en nuestro caso llamaremos ***clase***.

Una *clase* describe ciertas características del instrumento. En el nivel que estamos ahora describe cómo está construido el instrumento, qué tipo de sonidos debe emitir. No es el mismo papel el que juega el piano en el concierto, que un violín o una tarola, por lo que la participación de cada instrumento va a ser distinta, aunque todos los instrumentos tienen presente la pieza que deben tocar.

También los instrumentos de cuerdas se clasifican, a su vez, en violines, contrabajos, guitarras, entre otros, pero dependiendo del concierto que se está preparando va a ser el tipo y número de instrumentos que participen. Si la orquesta es muy buena, podemos pensar en que la entrada está constituida por los tipos de instrumentos y el número de cada tipo, además de las distintas partituras que corresponden a cada instrumento, dependiendo del concierto de que se trate.

En una orquesta puede haber un primer violín y un segundo violín, además del conjunto de violines. Las responsabilidades del primer violín son distintas que las de los violines comunes. El primer violín *especializa* o *extiende* las responsabilidades de cualquier violín.

El primer violín tiene su propio violín, la partitura marcada especialmente y sabe tocar lo que le toca. Cada partitura dice cuándo deben tocar, en relación con otros instrumentos, lo que también organiza el director de la orquesta.

El concierto consiste en poner a trabajar a cada uno de los músicos, bajo la dirección del Director de Orquesta, para que juntos produzcan el sonido total de forma armoniosa (***concurrente***).

En la programación orientada a objetos el director es un método que se llama ***main*** y que aparece en alguna de las clases que forman parte de tu proyecto. Si no hay al menos un método main no se puede ejecutar tu proyecto.

A la forma como se lleva a cabo una *responsabilidad* es a lo que en orientación a objetos llamamos un ***método*** o ***función propia***. Cada clase y los objetos que pertenezcan a esa clase tienen la especificación de cómo cumplir con sus responsabilidades.

Cuando le pides a un objeto que haga algo, le estás enviando un ***mensaje*** o ***invocando a un método*** del objeto.

Cada uno de los músicos, inmediatamente antes del concierto, realiza pruebas o ensayos con su instrumento (*afina* el instrumento). A este tipo de pruebas, en programación, se le llaman ***pruebas unitarias***. En este tipo de pruebas el programador verifica que cada objeto o actividad (función o método) trabaje como la documentación (partitura y descripción del instrumento) dice que debe trabajar y produzca el sonido que debe producir (la salida que se espera de la actividad).

A distintos músicos les puede tocar ser primer violín en una misma pieza pero en distinto concierto. Sin embargo, sea quien sea el que juegue el papel del primer violín, éste siempre tendrá que tocar exactamente lo mismo. Su comportamiento está dictado por la partitura de la pieza y por el Director de Orquesta que dirige el concierto.

Podrías decir que el papel del primer violín constituye una ***clase*** cuyas características son que posea un violín, tenga una copia de la partitura, sepa leer música, sepa tocar violín, tenga un frac (para el concierto), entre otros. Estas características las podemos clasificar en "artículos" que debe tener (***atributos***) y actividades que debe saber realizar (***conducta, comportamiento***).

Si ves al conjunto de los músicos como una clase que tienen características en común, puedes distinguir entre ellos al Director de la Orquesta, que también es un músico, pero sólo puede haber uno en un concierto y tiene asignadas responsabilidades distintas que cada uno de los músicos que toca un instrumento. Se encarga de supervisar que cada músico toque lo parte que le corresponde,

en el momento en que lo debe tocar y lo debe hacer bien. ***Coordina*** y ***organiza*** la realización del Concierto.

Los ensayos antes del concierto es lo que en programación llamas ***pruebas de integración***, ya que el Director se encarga de supervisar que la colaboración entre los músicos sea la adecuada, ***integra*** a los músicos para que actúen como uno solo.

Otro término de programación que puedes observar en un concierto, es que aunque la melodía debe sonar integrada, cada músico está, realmente, trabajando por su lado con una determinada responsabilidad. La responsabilidad del desarrollo del concierto está ***distribuida*** entre los músicos quienes, simplemente hacen lo que les toca hacer, de manera coordinada. Por tanto, el concierto es un ejemplo de ***ejecución distribuida***.

Para terminar puedes notar que la programación es como la preparación para un concierto: debes tener los instrumentos, las partituras, a los músicos, al Director de Orquesta. La ejecución, sin embargo, no sucede hasta que no se lleva a cabo el concierto, hasta que los artistas no tocan frente a un público o los graban. Así, puedes escribir tus programas (y clases) pero hasta que no reciba entradas y no lo "sueltes" para lo que se conoce como producción, no has terminado.

Los conceptos más importantes relacionados con la orientación a objetos, y que de alguna manera has revisado relacionándolo con una orquesta, se encuentran a continuación:

Clase: Es una plantilla del comportamiento y datos o atributos que deberá tener un objeto que se declare de esa clase.

Superclase: Agrupa los conceptos y métodos comunes a varias clases. Es, a su vez, una clase.

Subclase: Extiende la definición, agregando métodos y/o atributos de una clase. Es, a su vez, una clase.

Objeto: Responden a un ejemplar o instancia[1] concreta de una clase dada.

Método o función propia: La manera como una clase define el cumplimiento de alguna responsabilidad.

Atributo o variables propias: Son los campos o características que define una clase para sus objetos.

Mensaje: Invocación o llamada a un método (se le pide al objeto que ejecute algún método).

Acceso: Dice qué otros objetos de clases distintas pueden ver, manipular o invocar a métodos de un objeto de una cierta clase.

Encapsulamiento: Limitando el acceso a algunas de las características de un objeto, logramos de cierta forma aislarlo del exterior. Quien usa al objeto sabe qué pedirle pero no sabe cómo le hace el objeto para responder. Es como si tuvieras una cápsula alrededor del objeto.

Polimorfismo: Cuando dos métodos se llaman de la misma forma, pero dependiendo del objeto al que se dirija la petición, el resultado sea distinto: misma forma pero distinto resultado.

Herencia: La definición de subclases de una clase dada, en tantos niveles como se desee.

[1] Traducción literal, pero incorrecta, del término *instance* del inglés

Java como un lenguaje orientado a objetos

Si bien Java no es el primer lenguaje orientado a objetos que se construyó, es uno de los más populares hoy en día porque se orientó mucho a su uso en Internet. El lenguaje o alguna derivación del mismo está presente en muchos dispositivos electrónicos y en muchos navegadores para el Web.

Java no es un lenguaje "puro" de objetos porque puedes tener atributos o tipos que se conocen como *primitivos*, como los números, los caracteres, los valores lógicos de verdadero y falso, que no son objetos. Sin embargo, para programar un proyecto en Java se tiene que hacer mediante clases y objetos, dentro de los cuales puedes tener tipos primitivos o de clase. Además, como dijimos anteriormente, para lograr la ejecución del proyecto se tiene que solicitar desde una clase que tenga un método `main`.

Las clases de Java se compila a un lenguaje de máquina, llamado *bytecode*, que es ejecutado posteriormente por la Máquina Virtual de Java (JVM)[2], que es, a su vez un programa en la computadora que reconoce el bytecode y sabe ejecutarlo.

Como un proyecto está organizado en clases y los objetos de esas clases interaccionan entre sí, Java es un lenguaje orientado a objetos.

[2]por sus siglas en inglés de *Java Virtual Machine*