

Fundamentos de la POO¹

La correcta aplicación de los conocimientos de clases y objetos nos permitirán llegar a los objetivos planteados y a generar programas que puedan ser fáciles de comprender para escalar y mantener.

El lenguaje de Java, además ayudará a sensibilizarte en el uso de lenguajes orientados a objetos y te permitirá sentirte cómodo para usar otros con sintaxis y orientación similar como lo son C++, C# y PHP entre otros. Comencemos.

1. Objetos y Clases

Clases

Cuando escribimos un programa computacional en un lenguaje orientado a objetos, estamos modelando una representación del mundo real. Estos objetos se pueden agrupar en categorías dependiendo de sus características y estas categorías son las que llamamos clases.

Para hacer más clara la explicación veamos un ejemplo. Imagina que vamos a desarrollar un videojuego sobre autos. Podríamos enumerar fácilmente las características más claras de un auto genérico: color, número de llantas, modelo, número de caballos de fuerza. Ahora podríamos darle valores a esas características y decir que existe un auto rojo, con cuatro ruedas, modelo 2014 con 150 caballos de fuerza.

Nos damos cuenta entonces que podemos definir una plantilla básica para describir cualquier tipo de automóvil, una **clase** llamada Auto.

Como vimos en la unidad anterior, entonces podemos usar esta clase como “molde” para crear (más adelante veremos cómo) autos con los mismos atributos pero con diferentes valores.

Objetos

Ya describimos qué es una *clase*, ahora es el turno de ver qué es un *objeto*. Retomando nuestro ejemplo anterior, sabemos que tenemos una *clase* Auto, la cual tiene información básica que puede definir a cualquier auto.

Ahora, si tomamos esa clase para crear un caso particular, se dice que **instanciamos un objeto**. Es decir, ese objeto es un caso particular de dicha clase que usa la clase (molde) y toma valores en los atributos.

¹ Documento elaborado por el Ing. Arturo Nereu Núñez.

Sin entrar en detalle sobre código aún, podemos inferir que dada la definición de la clase Auto, podríamos generar dos autos (objetos) con las mismas propiedades pero diferentes valores.

Clase **Auto**.

Atributos **Color, Número de llantas, modelo, caballos de fuerza**.

Auto 1:

Color = **Rojo**, Número de llantas = **4**, modelo = 2014, caballos de fuerza = 150.

Auto 2:

Color = **Azul**, Número de llantas = **8**, modelo = 2004, caballos de fuerza = 80.

Vemos pues, que la relación entre una clase y un objeto significa que la clase es la definición abstracta de propiedades y un objeto es una instancia “real” de dicha clase con valores.

2. Fundamentos de Programación

En nuestro ejemplo de la unidad anterior, para imprimir un mensaje Hola Mundo, tuvimos que escribir el siguiente código.

```
public class HolaMundo
{
    public static void main(String[] args)
    {
        System.out.println("Hola Mundo");
    }
}
```

Como podemos ver, definimos una clase llamada HolaMundo, que se comprende por el cuerpo y todo lo que está adentro.

```
public class HolaMundo
{
    ...
}
```

Además, hay un método público y estático llamado *main*, que en Java, es el punto central de un programa. Ese método main es el que se ejecuta primero en un programa. Como en este caso lo único que se hace es mandar un mensaje a consola.

Lo interesante aquí es ver que ya creamos una clase y al momento de compilar y ejecutar el código con BlueJ, automáticamente se está instanciando un objeto del tipo HolaMundo y ejecutando su método main de inmediato.

Fundamentos de programación

Antes de continuar con la programación orientada a objetos, vamos a repasar los conceptos básicos de programación con Java, mismos que serán utilizados cuando veamos cómo se utilizan en la POO.

Tipos de datos en Java

Los tipos de datos primitivos en Java son definiciones primitivas para valores soportadas por el lenguaje, pero hay otros tipos de datos más complejos y éstos son las clases que nosotros definimos.

Java soporta varios tipos de datos primitivos, aquí hay una pequeña lista, pero para conocer todos los disponibles revisa la documentación oficial en:

<http://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html>

int: un valor numérico entero que sirve para guardar valores como: 1, 32332, -120, etc.

float: un valor numérico que permite guardar valores decimales como: 1.4, -2.22, 1234.56, etc.

boolean: un valor lógico que nos permite guardar valores binarios: true, false.

char: un valor que permite guardar caracteres del tipo: 'a', 'c', 'p', etc.

A partir de estos tipos de datos podremos definir otros más complejos mediante clases.

Para guardar un valor de un tipo de dato, necesitamos nombrar una variable, a este nombre se le llama identificador, para saber las reglas para nombrar identificadores, visita:

<http://ova.ufps.edu.co/drupal/files/OA1.3/html/subtema131.html>

Hablando ya de clases y objetos, vamos a referirnos al estado de un objeto como el conjunto de valores de sus diferentes propiedades. En el ejemplo del auto, el estado del objeto es: Color = **Rojo**, Número de llantas = **4**, modelo = 2014, caballos de fuerza = 150.

Métodos

En la siguiente unidad veremos cómo se utilizan los métodos dentro de las clases y objetos, por el momento recordemos la estructura básica de la definición de un método.

```
[modificadores] valor_de_retorno nombre_del_metodo(parametros)
{
    cuerpo_del_mensaje
}
```

Los modificadores en Java, son palabras reservadas que sirven para cambiar el alcance del método (más adelante veremos esto).

El valor de retorno es el tipo de dato que regresa dicho método después de ejecutarse, generalmente es el resultado de diferentes operaciones.

El nombre del método se refiere al identificador único que utilizaremos para mandar a ejecutarlo, hay que tomar en cuenta que un método puede llamarse con cualquier identificador válido, además hay que buscar nombres claros que describan lo que hacen, por ejemplo, para un método que hace la suma de dos números podríamos llamarlo *sumaNumeros*

Los parámetros, son valores adicionales que se envían al método para ser utilizados dentro del cuerpo del mismo

Por último, el cuerpo del mensaje es donde se ejecutan las instrucciones que dan lugar a cambio de estado del objeto, un resultado a regresar o un cambio en los valores que se envían como parámetro.

Programando

Vamos a poner manos a la obra y a practicar los conceptos que hemos visto utilizando JAVA. Abre BlueJ y crea un nuevo proyecto llamado **Programacion101**. Dentro crea una clase y nómbrala **Calculadora**. Borra todo el contenido y comienza con el siguiente código solamente:

```
public class Calculadora
{
    public static void main(String[] args)
    {
    }
}
```

Vamos a crear una calculadora sencilla que nos permita sumar, restar, multiplicar y dividir. Para ello comenzaremos definiendo los métodos de suma y resta, mismos que quedarán de la siguiente manera:

```
public class Calculadora
{
    public static void main(String[] args)
    {
    }

    int suma(int a, int b)
    {
        int resultado = a + b;
        return resultado;
    }

    int resta(int a, int b)
    {
        int resultado = a - b;
        return resultado;
    }
}
```

Vemos que nuestros métodos regresan un valor del tipo `int`, tienen su nombre y reciben como parámetros dos valores enteros: `a` y `b`.

Además, dentro del método, se define una nueva variable llamada `resultado` que guarda el valor de la operación efectuada y al final se regresa.

Para usar nuestros métodos, dentro del método *main* (que se ejecuta cuando comienza el programa de forma automática), vamos a hacer lo que se llama invocar un método:

```
public class Calculadora
{
    public static void main(String[] args)
    {
        int valorA = 10;
        int valorB = 20;

        Calculadora calc = new Calculadora();

        int resultadoSuma = calc.suma(valorA, valorB);
        int resultadoResta = calc.resta(valorA, valorB);

        System.out.println("A + B = " + resultadoSuma);
        System.out.println("A - B = " + resultadoResta);
    }

    int suma(int a, int b)
    {
        int resultado = a + b;
        return resultado;
    }

    int resta(int a, int b)
    {
        int resultado = a - b;
        return resultado;
    }
}
```

A pesar de que hay más código, no te preocupes, es bastante sencillo:

En el método *main* declaramos dos valores *valorA* y *valorB* a los cuales asignamos valores numéricos.

Luego, vamos a instanciar un objeto de la clase calculadora llamado **calc**, después veremos cómo funciona más a detalle, por ahora, sabemos que un objeto instanciado con la clase *Calculadora* tiene como propiedades dos métodos, uno que suma y otro que resta.

Después, declaramos dos variables para guardar los resultados de las operaciones dentro de los métodos que estamos invocando del objeto *calc*. Para eso usamos el operador “.” (punto) que básicamente nos dice: “del objeto *calc*, ejecuta el método *suma*”

Al final, simplemente vamos a imprimir el resultado a pantalla, esto lo hacemos **concatenando** el texto “A + B = ” con el valor obtenido de las invocaciones de los métodos.

Referencias

<http://www.cs.princeton.edu/courses/archive/spr96/cs441/notes/l1.html>

Libros de texto en la biblioteca digital