Objeto, clases y sobrecarga

¡Comencemos!

¿Qué aprenderás?

- Comprender el paradigma de la Programación Orientada a Objetos (POO).
- Comprender la estructura e instancia de una clase y sus atributos para aplicar la sobrecarga de métodos.

Introducción

En este capítulo veremos un concepto clave en la programación y el porqué es uno de los más utilizados en cualquier lenguaje. Su nombre es “Programación Orientada a Objetos” o “P.O.O” por sus iniciales.

Entenderemos el porqué y cómo se llegó a este paradigma y porqué es tan popular en diversos ámbitos del desarrollo. Esto nos permitirá observar el comportamiento de los objetos en Java y para esto utilizaremos nuevamente el IDE de Eclipse. Aplicaremos lo aprendido en unidades anteriores para consolidar y construir nuestras primeras clases con las buenas prácticas de Java.

¡Vamos con todo!

Los objetos

Cuando hablamos de objetos es necesario entender que nos estamos refiriendo a todo tipo de objetos que podamos ver incluso en el mundo real, los cuales tienen atributos y comportamientos.

El ejemplo más usado es el de un automóvil. Cuando hablamos de un automóvil, podemos ver atributos o propiedades tales como color, marca, modelo y todas esas características del objeto que lo hacen mantener un estado único que lo diferencia de otros objetos. Además de las propiedades, mencionamos que los objetos tienen un comportamiento, con esto nos referimos a lo que el objeto puede hacer, como Avanzar, Retroceder, Encenderse, entre otras.

Si analizamos estas funcionalidades sin mucho detalle, comprendemos que cada funcionalidad contempla una o más piezas diferentes que interactúan entre sí para lograr el objetivo, por ende, tenemos un sistema completo dentro del auto, compuesto por piezas modulares.

Esto mismo pasa en la Programación Orientada a Objetos, ya que tenemos muchas piezas y todas ellas con características que cumplen funciones por sí solas o trabajando en conjunto con otras piezas. A estas piezas les llamamos objetos (que en Java se les conoce como “Plain Old Java Object” o también conocidos como “POJO”). Las características de estos objetos las almacenamos en variables y las funciones las llamamos métodos; los métodos, no son nada más que porciones de código dentro de un objeto.

Las clases

Instancia de una clase

Cuando tenemos un auto en específico es porque ya pasamos por una fase de construcción del objeto, ya podemos decir que el auto tiene color, marca y atributos definidos, en programación a esto se le llama estado o instancia de una clase.

Podemos decir entonces que una instancia es una forma de representar una clase "dándole vida" en forma de objeto. Se dice que para **"instanciar"** una clase, debemos ocupar el operador y la palabra reservada `new`. Con esta palabra podemos crear una nueva instancia de la clase.

```
Cliente cliente = new Cliente();
```

Cuando se realiza una instancia con el operador `new` generamos copias de nuestras clases para realizar su ejecución y tratamientos de nuevos valores, pero sin afectar el objeto principal.

Estructura de una clase

Dentro del mundo tenemos muchos tipos de autos que tienen las mismas funcionalidades (Avanzar, Retroceder, Frenar, etc), pero con diferentes características, es decir, podemos tener autos de diferentes marcas, modelos, colores.

Entonces, podríamos decir que las clases son plantillas en las que nos basamos para crear objetos y que para crearlos debemos conocer los siguientes conceptos:

- **Palabras reservadas de Java:** Existen palabras únicas del lenguaje que no se pueden utilizar como variables, nombre de métodos o de clases. A continuación, veremos una tabla con algunas de las más utilizadas en Java:

boolean	byte	char	double	float
int	long	short	public	private
protected	abstract	final	native	static
synchronized	transient	volatile	if	else
do	while	switch	case	default
for	break	continue	assert	class
extends	implements	import	instanceof	interface
new	package	super	this	catch
finally	try	throw	throws	return
void	null	enum		

Imagen 1. Palabras reservadas de JAVA.
Fuente: Desafío Latam

- **Modificador de acceso:** Determina la accesibilidad que tendrán otras clases sobre la clase. Normalmente las clases utilizan el término `public`.
- **class:** Palabra reservada del lenguaje que determina la definición de una clase.
- **Nombre Clase:** Identificador que representa el nombre de la clase.
- **Atributo de instancia**: Dentro de una clase se definen atributos de tipo clase, esto quiere decir que cualquier método u operación en la clase puede utilizarlos.
- **Atributo local:** Estas variables se pueden crear dentro de los métodos y solo serán visibles desde dentro y no fuera de la clase.
- **Operaciones o Métodos:** Son todas las operaciones especiales de una clase que no se declaran como atributos de la clase. Estos métodos nos permiten que la clase realice acciones propias y existen de 1 a N métodos dentro de una clase. Estos métodos deben ser coherentes con la clase.

Ejemplo de una clase llamada Persona:

```
public class Persona {
    private String nombre;
    private String rut;
    private double altura;
}
```

Constructores y sobrecarga de métodos

En Java existe un operador llamado `this`, el cual nos permite hacer referencia a variables o métodos de la clase, este operador solo puede ser ocupado de esta forma, de lo contrario lanzará error de compilación.

Los constructores nos permiten crear instancias de nuestras clases en ejecución, estos permiten incluir parámetros de entradas para que cada instancia sea diferente a otra, es decir, cada clase en ejecución tiene los mismos atributos pero con distintos valores. Además los constructores:

- Llevan el mismo nombre de la clase y en sus paréntesis se declaran los parámetros de entradas.
- El operador `this` hace referencia a las variables de clases y se le asigna lo que viene como parámetro.

Un ejemplo de constructor es el que se puede ver a continuación:

```
  public Persona(String nuevoNombre,  String nuevoRut,  double
nuevaAltura) {
       this.nombre = nuevoNombre;
       this.rut = nuevoRut;
       this.altura = nuevaAltura;
    }
```

Con esto podemos crear una instancia de un objeto dentro o fuera de la clase, como se muestra a continuación con la clase `Persona`, donde los parámetros que le colocamos al interior del paréntesis son aquellos que declaramos en el constructor previo.

```
public Persona instanciaClase(){
      Persona personita =  new Persona ("Juan","1-9",12.2);
      return personita;
    }
```

En este método tenemos dos casos particulares:

- Una variable de tipo local llamada `personita` que es de tipo `Persona`
- Una instancia de clase Persona ocupando el operador `new` con nuevos valores provenientes del ejemplo constructor.

También podemos tener dos o más constructores según vayamos necesitando. Por ejemplo, si necesitamos un constructor que solo reciba el nombre y la altura pero sin el Rut, tendríamos lo siguiente:

```
public Persona(String nuevoNombre,  double nuevaAltura) {
      this.nombre = nuevoNombre;
      this.altura = nuevaAltura;
   }
```

Este mismo constructor lo podemos llamar dentro del primer constructor, esto para realizar sobrecarga de métodos, es decir, generamos un ahorro de lógica y optimizamos recursos en Java.

```
public Persona(String nuevoNombre,  String nuevoRut,  double
nuevaAltura) {
      // A esto se le llama sobrecarga de métodos
      this(nuevoNombre,nuevaAltura);
      this.rut = nuevoRut;
   }
   public Persona(String nuevoNombre,  double nuevaAltura) {
      this.nombre = nuevoNombre;
      this.altura = nuevaAltura;
   }
```

¿Qué son los access modifiers o “modificadores de acceso”?

En Java existen 2 métodos convencionales que son usados para recibir y actualizar el valor de una variable: `getter` y `setter`. Estos métodos de acceso nos permiten acceder al valor de un atributo (getter = obtención) y colocar el valor de un atributo (setter = colocar), en otras palabras, estos modificadores de acceso nos permite definir una propiedad para un atributo determinado que puede iniciarse tanto dentro como fuera de una clase.

Getter

Es un método que obtiene el valor de una variable, por lo que si accedemos a este método desde otra función o clase, podremos ver el valor guardado en un objeto. La función getter es típicamente pequeña y simple, ya que retorna el dato guardado en un objeto. Por ejemplo, podemos ver a continuación cómo se inicia una variable con getter.

```
public class RazaPerro{

 private String raza;

  public String getRaza(){
    return raza;
    }
}
```

Setter

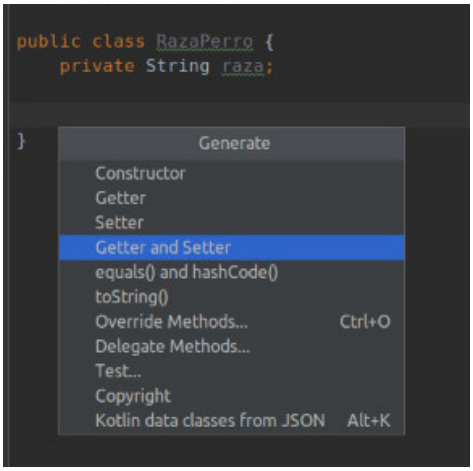
Por su parte, el método setter permite cambiar el valor guardado de una variable de un objeto, es decir, el método permite actualizar el valor de una variable. A continuación veremos un ejemplo de cómo sería iniciar y "setear" esa variable.

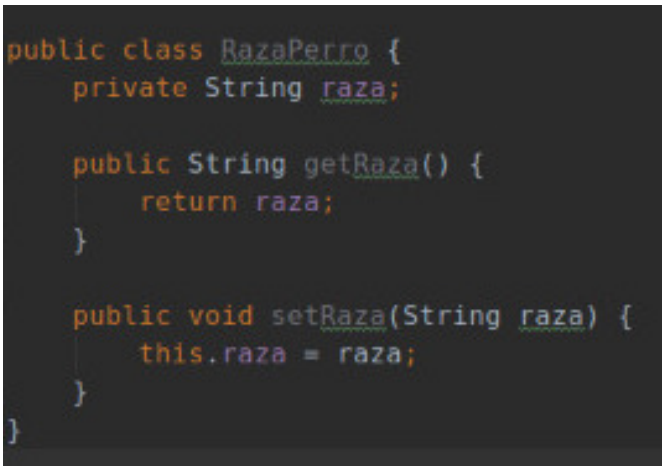
```
public class RazaPerro{

 private String raza;

  public void setRaza(String raza){
    this.raza = raza;
    }
}
```

Para generar ambos métodos, podemos inicializar las variables en la clase, hacer click secundario al interior de la clase y seleccionar la opción "getters and setters" para que genere automáticamente los modificadores de acceso en esa clase con esos parámetros.





Luego de seleccionar “Getter and Setter” nos queda la clase como se muestra en la imagen de la derecha.

Ejercicio guiado: Auto

Paso 1: Crear una clase llamada “Auto” dentro de un package cualquiera de un nuevo proyecto.

```
Click secundario sobre el package -> new -> class
```

Paso 2: Declaramos las siguientes variables o características en la clase Auto.

- altura: double
- ancho: double
- tipo material: String
- color: String

```
public class Auto {
        private double altura;
        private double ancho;
        private String tipoMaterial;
        private String color;
}
```

Paso 3: Posterior a la última declaración de variable que es color, generamos el constructor para los parámetros de altura y tipo material. Para esto, podemos hacer clic secundario

sobre la clase auto y buscamos la opción “Source” -> “Generate Constructor using fields” y seleccionamos las variables. Nos debería quedar algo como esto:

```
public class Auto {
        private double altura ;
        private double ancho ;
        private String tipoMaterial;
        private String color;

    public Auto (double altura, String tipoMaterial) {
      this.altura = altura;
      this.tipoMaterial = tipoMaterial;
      }
}
```

Paso 4: Crear los “getter and setter” haciendo clic secundario sobre la clase `Auto` nuevamente y buscamos la opción “Source” -> “Generate Getters and Setters” y seleccionamos las variables. Nos debería quedar algo como esto:

```
public class Auto {
        private double altura ;
        private double ancho ;
        private String tipoMaterial;
        private String color;

      public Auto (double altura, String tipoMaterial) {

            this.altura = altura;
            this.tipoMaterial = tipoMaterial;
      }
      public double getAltura() {
            return altura;
      }

      public void setAltura(double altura) {
            this.altura = altura;
      }

      public double getAncho() {
            return ancho;
      }
```

```
	public void setAncho(double ancho) {
		this.ancho = ancho;
	}

	public String getTipoMaterial() {
		return tipoMaterial;
	}

	public void setTipoMaterial(String tipoMaterial) {
		this.tipoMaterial = tipoMaterial;
	}

}
```