Polimorfismo por Sobrecarga

Material de clase elaborado por Sandra Victoria Hurtado Gil

Una de las características importantes de los objetos, y en general de la Programación Orientada a Objetos (POO), es la capacidad de comportarse diferente ante un mismo mensaje, dependiendo de los parámetros que recibe o del tipo de objeto que sea. Esto se conoce como **polimorfismo**.

Esta capacidad de comportarse diferente puede darse en dos casos:

- Cuando se pide el mismo servicio a diferentes objetos, y cada uno de ellos puede hacerlo de forma diferente. Esto se conoce como **polimorfismo por sobrescritura**.
- Cuando la información que se le da al objeto para solicitarle el servicio cambia. Es decir, se envían diferente tipo y cantidad de parámetros a un método. Esto se conoce como **polimorfismo por sobrecarga**.

El polimorfismo por sobrecarga se presenta cuando se tiene un mismo método pero que cambia su comportamiento dependiendo de los parámetros que recibe.

Por ejemplo: Las cadenas (objetos *String*), tienen el método "substring*"*, que permite obtener una sub-cadena a partir de otra cadena, lo cual se puede hacer de dos formas:

- Enviando como parámetros dos números enteros que representan las posiciones inicial y final dentro de la cadena inicial, o
- Enviando un solo número entero, de manera que la sub-cadena se obtendrá a partir de esa posición y hasta el final de la cadena inicial.

Por ejemplo:

```
String cadenaOriginal = new String("Esta es una cadena de texto");
String subcadenaUno = cadenaOriginal.substring(5,18);
String subcadenaDos = cadenaOriginal.substring(5);
System.out.println(subcadenaUno);
System.out.println(subcadenaDos);
```

La salida del anterior código es:

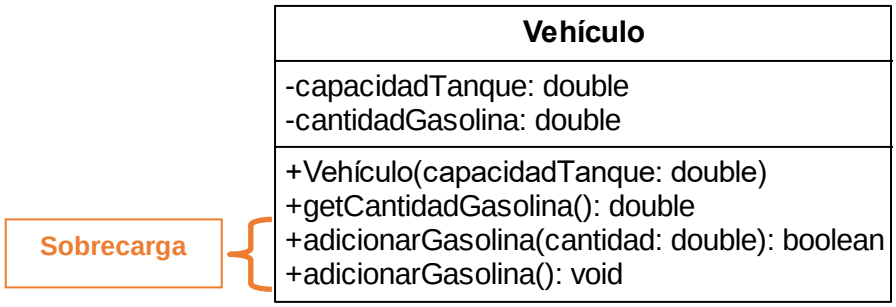
```
es una cadena
es una cadena de texto
```

En este caso se usa el mismo método: "substring", sobre el mismo objeto, pero como tienen un número diferente de parámetros el resultado cambia.

Implementación en Java

En Java el polimorfismo por sobrecarga se logra cuando en la misma clase se tienen **varios métodos con el mismo nombre**. Sin embargo, los métodos deben diferenciarse en el tipo o el número de parámetros que reciben.

Por ejemplo, se tiene la clase Vehículo, que se muestra a continuación:



En este caso al vehículo se le puede adicionar más gasolina de dos formas: indicando la cantidad de gasolina que se desea echar o sin indicar nada. Si no se indica nada entonces se llena el tanque.

El código de esta clase es:

```
/**
 * Medio de transporte mecánico al cual se le puede adicionar gasolina
 * @version 2.0
 */
public class Vehiculo {
	private double capacidadTanque;
	private double cantidadGasolina;

	public Vehiculo(double capacidadTanque) {
		this.capacidadTanque = capacidadTanque;
		this.cantidadGasolina = 0;
	}

	public double getCantidadGasolina() {
		return this.cantidadGasolina;
	}

                                                    //continúa
```

```
	/**
	 * Adicionar gasolina, máximo hasta la capacidad del tanque
	 * @param   cantidad    litros de gasolina que se desean adicionar
	 * @return si pudo o no adicionar la cantidad.
	 */
	public boolean adicionarGasolina(double cantidad) {
		double maximo = this.capacidadTanque - this.cantidadGasolina;
		if (cantidad <= maximo) {
			this.cantidadGasolina += cantidad;
			return true;
		}
		return false;
	}

	/**
	 * Adicionar gasolina hasta la capacidad del tanque
	 */
	public void adicionarGasolina() {
		this.cantidadGasolina = this.capacidadTanque;
	}
}  //Fin clase Vehiculo
```

Java determina automáticamente cuál es el método que se debe llamar dependiendo de los parámetros que se envíen.