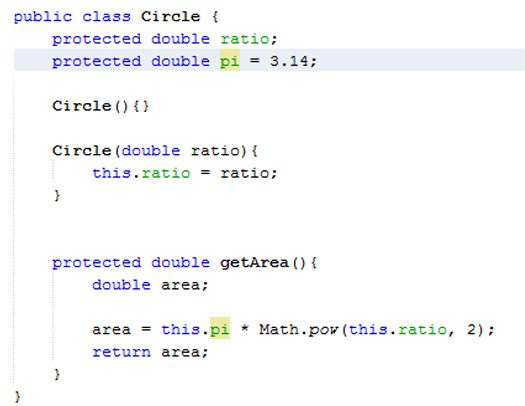
¿Cuando utilizarlos?

**Override.**

**La sobreescritura (override) se utiliza a traves de la herencia, ya que este se da unicamente cuando un metodo heredado dentro de una clase hija, posee las mismas caracteristicas que el metodo de la clase padre; (cantidad de parametros, tipos de datos, y tipo de dato de retorno del metodo) y lo unico que cambia es la definicion de dicho metodo.**

**Ejemplo:**

**Para este ejemplo supogamos que tenemos una clase llamada “Circle” en la cual existe un metodo llamado “getArea()” el cual calcula el area de un circulo.**



***Atributos de clase***

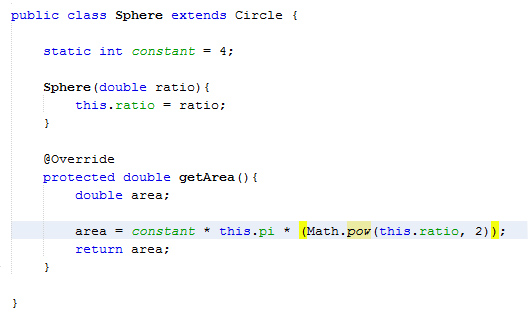
*Constructores de la clase, uno vacio y otro que recibe un parametro que en este caso es el radio del circulo*

*Metodo que calcula el area (πr2) de un circulo, este metodo es el que vamos a sobreescribir (override)*

**Ahora bien supongamos que tenemos otra clase que se llama “Sphere” la cual define una esfera que vendria siendo un circulo en 3 dimensiones, por lo que hereda de la clase “Circle”, en dicha herencia todos los atributos y metodos de la clase “Circle” seran heredados a la clase “Sphere”, incluyendo el metodo “getArea()” el cual calcula el area del ciculo, pero para el caso de la esfera el calculo se realiza de forma distinta, por que convendria sobreescribir el metodo y quedaria de la siguiente manera.**

*Herencia de la clase “****Circle****”*

*Herencia de la clase “****Circle****”*



*Sobreescritura del metodo* ***getArea()*** *ya que el caclculo del area de una Esfera es (4πr2)*

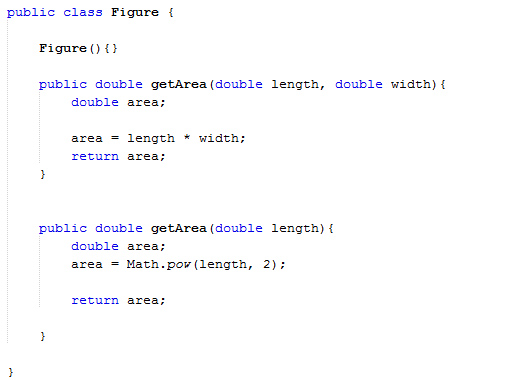
*Es importante resaltar que ambos metodos poseen el mismo argumento es decir la cantidad de parametros como el tipo de dato de los mismo. En este caso ambos metodos no reciben parametros y retornan un valor de tipo* ***double.***

**Overload.**

**La sobrecarga (overload), a diferencia de la sobreescritura, se utiliza cuando necesitamos que un metodo dentro de una misma clase, realice la misma funcionalidad pero recibiendo parametros distintos (cantidad y tipos de datos) y asi como tambien retorne un tipo de dato distinto, en sus diferentes definiciones.**

**Ejemplo:**

**Para este caso supongamos que tenemos una clase que se llama “*Figure*” la cual posee un metodo “*getArea()*” el cual calcula el area de un cuadrado o de un rectangulo dependiendo de la cantida y el tipo de parametros que tenga el argumento de dicho metodo.**



*Metodo para calcular el area de un rectangulo, el cual recibe 2 parametros en su argumento el lado “****length****” y el ancho “****width****” para poder realizar el calculo.*

*Es importante resaltar que ambos metodos se encuentran dentro de la misma clase, y los que los diferencia es su argumento en este caso, ya que dependiendo de los parametros que reciba, es donde se define que metodo se ejecuta*

*Metodo para calcular el area de un cuadrado, el cual solo recibe un parametro en su argumento en este caso es el lado “****length****” para poder realizer el calculo.*

|  |  |
| --- | --- |
| Override  (SobreEscritura) | Overload  (SobreCarga) |
| La firma del metodo(numero de parametros, tipos de datos etc.) y tipo de dato de retorno del mismo tienen que ser iguales (Lo cual impulsa al concepto de Polimorfismo). | La firma del metodo (numero de parametros, tipos de datos etc.) tienen que ser distintos para poder sobrecargar un metodo. |
| Solo ocurre si un metedo heredado en la clase hija necesita una definicion distinta que la que se encuentra especificada en la clase padre. | Solo ocurre sin un metodo dentro de una misma clase necesita una definicion diferente. |
| Ocurre en tiempo de ejecucion del Sistema | Ocurre o se ejecuta al en tiempo de compilacion de la aplicacion. |
| El constructor de una clase no puede ser sobreescrito | Se pude sobrecargar un constructor |
| Un metodo estatico (static) no puede ser sobreescrito | Cualquier metodo puede ser sobrecargado |

Diferencias entre Override y Overload