# **Herencia en Java - OCP y LSP**

## **Conceptos**

* Herencia.
* Clases abstractas.
* Arrays de objetos.
* Arrays de longitud variable.

## **Principios SOLID**

Estos principios guían el diseño ágil de tu aplicación:

* SRP (S) o Principio de Única Responsabilidad (*Single Responsibility Principle*): una clase sólo debe tener un motivo para cambiar.
* OCP (O) Open/Closed Principle: las entidades de software (clases, módulos, funciones, etc.) deberían estar abiertas a su extensión pero cerradas a su modificación.
* LSP (L) o Principio de sustitución de Liskov: los objetos de un programa deberían ser reemplazables por instancias de sus tipos base sin alterar el correcto funcionamiento del programa (herencia y polimorfismo).

## **Ejercicio**

Diseña un programa que utilice el mecanismo de Herencia en Programación Orientada a Objetos, que permita realizar las siguientes operaciones.

Escribe un programa que cree objetos de las figuras geométricas que se proponen a continuación, los añada a un array (lee antes el apartado sobre arrays al final de este documento) y lo recorra mostrando en consola el nombre de la figura y el área de todos los objetos incluídos en él.

### **Tareas**

1. Define una clase Circulo para instanciar objetos de esta figura geométrica, que permita calcular su área a partir de su radio.
2. Define una clase Rectangulo para instanciar objetos de esta figura geométrica, que permita calcular su área a partir de sus lados.
3. Escribe un programa que almacene en un Array objetos de tipo Círculo y Rectángulo.
4. 3.a. El programa instancia objetos de tipo Círculo y de tipo Rectángulo, define sus propiedades, y los añade a un Array.
5. 3.b. A continuación, muestra por pantalla el nombre y el área de todas las figuras almacenadas en el array.
6. Define ahora una clase Cuadrado para instanciar objetos de esta figura geométrica, que permita calcular su área a partir de sus lados. Para añadir este nuevo tipo a tu programa y que éste siga funcionando con corrección, *no debes modificar el código que ya has escrito* (las clases y el programa principal), sino únicamente *extenderlo con nuevas sentencias*. Este es el principio de desarrollo ágil de software llamada Open/Closed Principle (OCP): las entidades de software (clases, módulos, funciones, etc.) deberían estar abiertas a su extensión pero cerradas a su modificación.
7. 4.a. Si llegas a la conclusión de que necesitas modificar el código del programa para incluir el tipo Cuadrado y poder calcular su área, intenta realizar un *nuevo diseño para anticipar la inclusión de nuevos tipos de figuras geométricas*, que permita añadirlas sin necesidad de alterar el código, sino *simplemente extenderlo con nuevas clases*.
8. 4.b. Para comprobar que el nuevo diseño que has realizado es correcto, añade ahora una nueva clase para instanciar objetos de tipo Elipse. Instancia este objeto en el programa principal, añádelo al array de figuras y visualiza su área junto al resto de objetos del array.

## **Arrays**

En el libro [Beginning Java 8 Fundamentals](https://drive.google.com/open?id=0B68CprMt4M4PREViRUkyNUx2MVU) en el capítulo 15 encontrarás todo lo que necesitas conocer sobre la estructura de datos Array, ArrayList y Vector.

* Un Array es una estructura de datos de lontigud fija que almacena valores de un mismo tipo de datos. Todos los objetos del array han de ser del mismo tipo. Has de indicar en la declaración del array su longitud (que no podrá variar a partir de ese momento, es decir, fijas la longitud del array en tiempo de compilación).
* La librería Java provee de las clases ArrayList y Vector que permiten crear objetos que se comportan como un array de longitud variable. Así, puedes añadir elementos en tiempo de ejecución, es decir, variar el tamaño en memoria del array (su longitud) en función de tus necesidades.
* Es importante incidir en que todos los objetos que añadas al ArrayList han de ser del mismo tipo:String, Integer, Character, Double, etc. o instancias (objetos) de una misma clase que hayas definido en el programa.