**package Tema3\_POO;**

/\*Una clase es una plantilla para crear tantos objetos como queramos\*/

**public class Rectangulo{**

/\*Los atributos son las características comunes que tiene un objeto. A los atributos se les debe poder dar un valor. Dejan de existir cuando el objeto deja de existir. Los atributos que no tienen como modificador la palabra static, se denominan atributos de instancia, lo que significa que forman parte de un objeto y por tanto debemos tener creado el objeto. Su ámbito de aplicación es toda la clase\*/

private int base;

private int altura;

/\*Los atributos que tiene el modificador static son atributos de clase. Es común para todos los objetos. Existen aunque no haya un objeto creado\*/

private static int contador=0;

/\*Los métodos indican las acciones que se pueden hacer con un objeto.

Los métodos que llevan el modificador static son métodos de clase, es decir, su implementación es igual para todos los objetos. Los métodos static no pueden hacer referencia a los atributos de un objeto, solo a los static, ni tampoco a métodos de instancia, ya que tanto los atributos como métodos de instancia identifican a los objetos\*/

**public static void contar\_objetos(){**

System.out.println("Creamos "+Rectangulo.contador+" objetos"); **}**

/\*Los métodos que no llevan el modificador static son métodos de instancia, lo que significa que forman parte de un objeto, y por tanto debemos tener creado el objeto.

Los métodos get devuelven el valor de un atributo\*/

**public int getBase(){**

return base;

**}**

**public int getAltura(){**

return altura;

**}**

/\*Los métodos set nos permiten dar valor o modificar el valor de un atributo, asignándole el valor del parámetro de entrada.

En los métodos set podemos implementar el control de los atributos de instancia (validaciones simples)\*/

**public void setAltura(int alt){**

altura=Math.abs(alt);/\*Controlamos que la altura del rectángulo sea siempre positiva, usando el método abs de la clase Math\*/

**}**

**public void setBase(int base){**

this.base=base;/\*Cuando el atributo se llama igual que el parámetro debemos poner la palabra reservada this. antes del nombre del atributo, puesto que si no, el parámetro oculta al atributo y no podemos tener acceso a él\*/

**}**

**public int calcular\_area(){**

return base\*altura;

**}**

/\*El método toString nos permite obtener resultados a través de una cadena de texto\*/

**public String toString(){**

return "El área del rectángulo de base "+base+" y de altura "+altura+" es "+calcular\_area();

**}**

/\*Los constructores son una especie de métodos que no devuelven ningún valor (no ponemos ni tan siquiera void), coincidiendo su nombre con el de la clase donde están definidos. Su función es construir el objeto, inicializando sus atributos. Los constructores pueden tener parámetros.

Si no se tiene escrito o implementado ningún constructor, el compilador de java crea uno por defecto (sin parámetros ni cuerpo), inicializando los atributos a sus valores por defecto, pero si escribimos al menos un constructor nosotros, java no incluye el constructor por defecto, y si queremos que los atributos tengan sus valores por defecto, tendremos que escribir un constructor donde se asignes a los atributos esos valores\*/

**public Rectangulo(){**

base=0;

altura=0;

contador++;

**}**

**public** **Rectangulo(int base, int alt){**

setAltura(alt);

setBase(base);

contador++;

**}**

**}**

/\*Por lo general, los constructores suelen estar sobrecargados, de esta forma aportamos dinamismo y flexibilidad a nuestra clase al poder inicializar un objeto de diferentes formas.

Los métodos, sean constructores o no, están sobrecargados si tienen el mismo nombre, pero deben diferenciarse por sus parámetros (por su número, por su tipo o por estar colocados en distinto orden). De esta forma el compilador no dará error al encontrar dos métodos o constructores que se llamen igual, pues mirará la versión del método cuyos parámetros coincidan con los argumentos. El tipo devuelto por el método no se tiene en cuenta para la sobrecarga\*/