7.G. Anexos de ejercicios resueltos.

1. Anexo I.- Elaboración de los constructores de la clase Rectangulo.

ENUNCIADO

Intenta rescribir los constructores de la clase `Rectangulo` teniendo en cuenta ahora su nueva estructura de atributos (dos objetos de la clase `Punto`, en lugar de cuatro elementos de tipo `double`):

1. Un constructor sin parámetros (para sustituir al constructor por defecto) que haga que los valores iniciales de las esquinas del rectángulo sean (0,0) y (1,1).
2. Un constructor con cuatro parámetros, **x1**, **y1**, **x2**, **y2**, que cree un rectángulo con los vértices (x1, y1) y (x2, y2).
3. Un constructor con dos parámetros, **punto1**, **punto2**, que rellene los valores iniciales de los atributos del rectángulo con los valores proporcionados a través de los parámetros.
4. Un constructor con dos parámetros, **base** y **altura**, que cree un rectángulo donde el vértice inferior derecho esté ubicado en la posición (0,0) y que tenga una base y una altura tal y como indican los dos parámetros proporcionados.
5. Un constructor copia.

POSIBLE SOLUCIÓN

Durante el proceso de creación de un objeto (**constructor**) de la **clase contenedora** (en este caso **Rectangulo**) hay que tener en cuenta también la creación (llamada a **constructores**) de aquellos objetos que son contenidos (en este caso objetos de la clase `Punto`).

En el caso del primer **constructor**, habrá que crear dos **puntos** con las coordenadas (0,0) y (1,1) y asignarlos a los atributos correspondientes (`vertice1` y `vertice2`):

```
public Rectangulo ()
{
    this.vertice1= new Punto (0,0);
    this.vertice2= new Punto (1,1);
}
```

Para el segundo **constructor** habrá que crear dos puntos con las coordenadas `x1`, `y1`, `x2`, `y2` que han sido pasadas como parámetros:

```
public Rectangulo (double x1, double y1, double x2, double y2)
{
    this.vertice1= new Punto (x1, y1);
    this.vertice2= new Punto (x2, y2);
}
```

En el caso del tercer **constructor** puedes utilizar directamente los dos puntos que se pasan como parámetros para construir los vértices del rectángulo:

Ahora bien, esto podría ocasionar un **efecto colateral** no deseado si esos objetos de tipo `Punto` son modificados en el futuro desde el código cliente del **constructor** (no sabes si esos puntos fueron creados especialmente para ser usados por el rectángulo o si pertenecen a otro objeto que podría modificarlos más tarde).

Por tanto, para este caso quizá fuera recomendable crear dos nuevos puntos a imagen y semejanza de los puntos que se han pasado como parámetros. Para ello tendrías dos opciones:

1. Llamar al **constructor** de la clase `Punto` con los valores de los atributos (x, y).
2. Llamar al **constructor copia** de la clase `Punto`, si es que se dispone de él.

Aquí tienes las dos posibles versiones:

Constructor que “extrae” los atributos de los parámetros y crea nuevos objetos:

```
public Rectangulo (Punto vertice1, Punto vertice2)
{
    this.vertice1= vertice1;
    this.vertice2= vertice2;
}
```

Constructor que crea los nuevos objetos mediante el **constructor copia** de los parámetros:

```
public Rectangulo (Punto vertice1, Punto vertice2)
{
    this.vertice1= new Punto (vertice1.obtenerX(), vertice1.obtenerY() );
    this.vertice2= new Punto (vertice2.obtenerX(), vertice2.obtenerY() );
}
```

En este segundo caso puedes observar la utilidad de los **constructores de copia** a la hora de tener que **clonar** objetos (algo muy habitual en las inicializaciones).

Para el caso del **constructor** que recibe como parámetros la base y la altura, habrá que crear sendos vértices con valores (0,0) y (0 + base, 0 + altura), o lo que es lo mismo: (0,0) y (base, altura).

```
public Rectangulo (Punto vertice1, Punto vertice2)
{
    this.vertice1= new Punto (vertice1 );
    this.vertice2= new Punto (vertice2 );
}
```

Quedaría finalmente por implementar el **constructor copia**:

```
// Constructor copia
public Rectangulo (Rectangulo r) {
   this.vertice1= new Punto (r.obtenerVertice1() );
   this.vertice2= new Punto (r.obtenerVertice2() );
}
```

En este caso nuevamente volvemos a **clonar** los atributos **`vertice1`** y **`vertice2`** del objeto **r** que se ha pasado como parámetro para evitar tener que compartir esos atributos en los dos rectángulos.