5.C. Métodos.

1. Métodos.

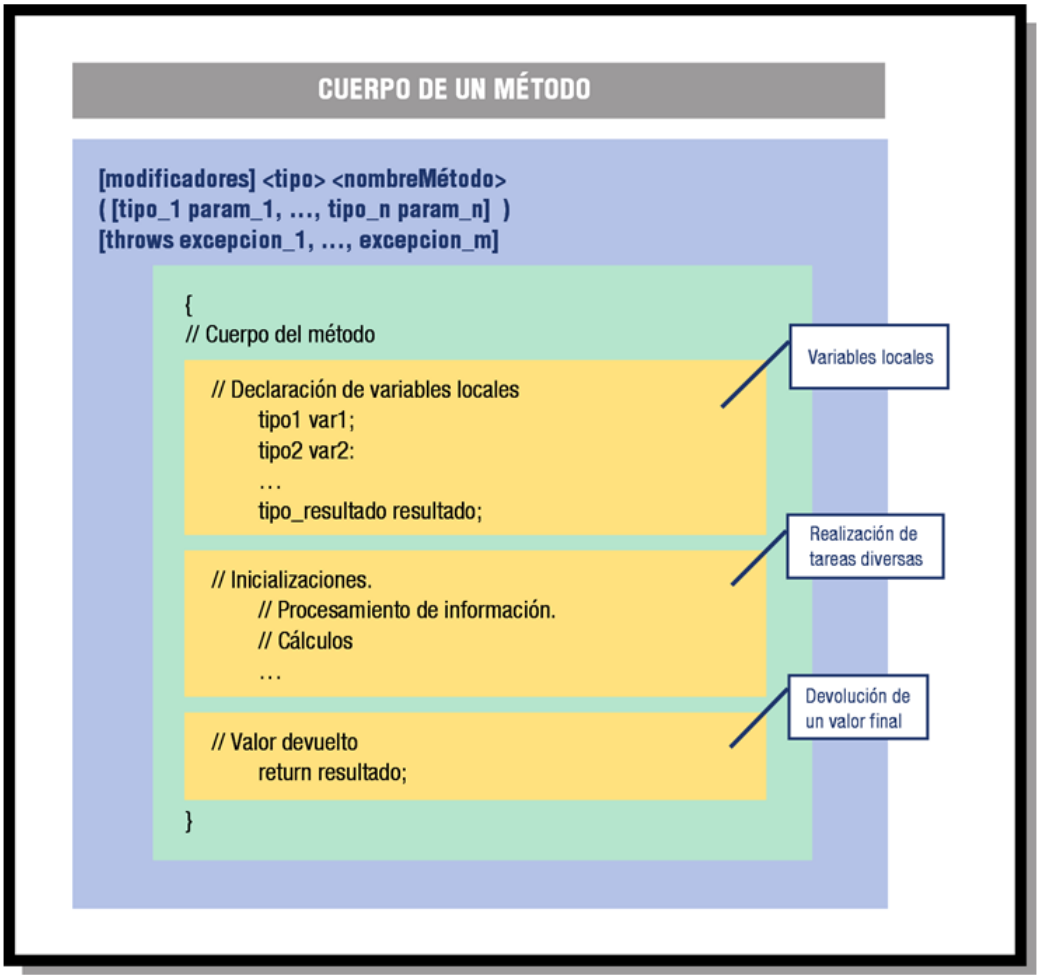
1.5. Cuerpo de un método.

El interior de un método (cuerpo) está compuesto por una serie de sentencias en lenguaje Java:

- Sentencias de **declaración de variables locales** al método.

- Sentencias que implementan la **lógica del método** (estructuras de control como bucles o condiciones; utilización de métodos de otros objetos; cálculo de expresiones matemáticas, lógicas o de cadenas; creación de nuevos objetos, etc.). Es decir, todo lo que has visto en las unidades anteriores.

- Sentencia de **devolución del valor de retorno** (`return`). Aparecerá al final del método y es la que permite devolver la información que se le ha pedido al método. Es la última parte del proceso y la forma de comunicarse con la parte de código que llamó al método (paso de mensaje de vuelta). Esta sentencia de devolución siempre tiene que aparecer al final del método. Tan solo si el tipo devuelto por el método es `void` (vacío) no debe aparecer (pues no hay que devolver nada al código llamante).



En el ejemplo de la clase `Punto`, tenías los métodos `obtenerX` y `obtenerY`.

```
int obtenerX ()
```

```
{
```

```
    return x;
```

```
}
```

```
int obtenerY ()
{
    return y;
}
```

Veamos uno de ellos:

En ambos casos lo único que hace el método es precisamente devolver un valor (utilización de la sentencia `return`). No recibe parámetros (mensajes o información de entrada) ni hace cálculos, ni obtiene resultados intermedios o finales. Tan solo devuelve el contenido de un atributo. Se trata de uno de los métodos más sencillos que se pueden implementar: un método que devuelve el valor de un atributo. En inglés se les suele llamar métodos de tipo **get**, que en inglés significa **obtener**.

Además de esos dos métodos, la clase también disponía de otros dos que sirven para la función opuesta (`establecerX` y `establecerX`). Veamos uno de ellos:

```
void establecerX (int vx)
{
    x= vx;
}
```

En este caso se trata de pasar un valor al método (parámetro `vx` de tipo `int`) el cual será utilizado para modificar el contenido del atributo **x** del objeto. Como habrás podido comprobar, ahora no se devuelve ningún valor (el tipo devuelto es `void` y no hay sentencia `return`). En inglés se suele hablar de métodos de tipo **set**, que en inglés significa poner o fijar (**establecer** un valor). El método **establecerY** es prácticamente igual pero para establecer el valor del atributo **y**.

Normalmente el código en el interior de un método será algo más complejo y estará formado un conjunto de sentencias en las que se realizarán cálculos, se tomarán decisiones, se repetirán acciones, etc. Puedes ver un ejemplo más completo en el siguiente ejercicio.

Ejercicio resuelto

Vamos a seguir ampliando la clase en la que se representa un rectángulo en el plano (clase **Rectangulo**). Para ello has pensado en los siguientes métodos **públicos**:

- Métodos **obtenerNombre** y **establecerNombre**, que permiten el acceso y modificación del atributo **nombre** del rectángulo.
- Método **calcularSuperfice**, que calcula el área encerrada por el rectángulo.
- Método **calcularPerímetro**, que calcula la longitud del perímetro del rectángulo.
- Método **desplazar**, que mueve la ubicación del rectángulo en el plano en una cantidad X (para el eje X) y otra cantidad Y (para el eje Y). Se trata simplemente de sumar el desplazamiento X a las coordenadas x1 y x2, y el desplazamiento Y a las coordenadas y1 e y2. Los **parámetros** de entrada de este método serán por tanto X e Y, de tipo `double`.
- Método **obtenerNumRectangulos**, que devuelve el número de rectángulos creados hasta el momento.

Incluye la implementación de cada uno de esos métodos en la clase **Rectangulo**.

Solución

En el caso del método **obtenerNombre**, se trata simplemente de devolver el valor del atributo **nombre**:

```
public String obtenerNombre () {
   return nombre;
}
```

Para el implementar el método **establecerNombre** también es muy sencillo. Se trata de modificar el contenido del atributo **nombre** por el valor proporcionado a través de un parámetro de entrada:

```
public void establecerNombre (String nom) {
   nombre= nom;
}
```

Los métodos de cálculo de superficie y perímetro no van a recibir ningún parámetro de entrada, tan solo deben realizar cálculos a partir de los atributos contenidos en el objeto para obtener los resultados perseguidos. En cada caso habrá que aplicar la expresión matemática apropiada:

- En el caso de la superficie, habrá que calcular la longitud de la **base** y la **altura** del rectángulo a partir de las coordenadas de las esquinas inferior izquierda (x1, y1) y superior derecha (x2, y2) de la figura. La base sería la diferencia entre x2 y x1, y la altura la diferencia entre y2 e y1. A continuación tan solo tendrías que utilizar la consabida fórmula de "base por altura", es decir, una multiplicación.
- En el caso del perímetro habrá también que calcular la longitud de la **base** y de la **altura** del rectángulo y a continuación sumar dos veces la longitud de la base y dos veces la longitud de la altura.

En ambos casos el resultado final tendrá que ser devuelto a través de la sentencia return. También es aconsejable en ambos casos la utilización de variables locales para almacenar los cálculos intermedios (como la base o la altura).

```
 public double calcularSuperficie () {

    double area, base, altura;   // Variables locales

    // Cálculo de la base

    base= x2-x1;

    // Cálculo de la altura

    altura= y2-y1;

    // Cálculo del área

    area= base * altura;

    // Devolución del valor de retorno

    return area;

}


public double calcularPerimetro () {

    double perimetro, base, altura;   // Variables locales

    // Cálculo de la base

    base= x2-x1;

    // Cálculo de la altura

    altura= y2-y1;

    // Cálculo del perímetro

    perimetro= 2*base + 2*altura;

    // Devolución del valor de retorno

    return perimetro;

}
```

En el caso del método **desplazar**, se trata de modificar:

- Los contenidos de los atributos x1 y x2 sumándoles el parámetro X,
- Los contenidos de los atributos y1 e y2 sumándoles el parámetro Y.

```
·       public void desplazar (double X, double Y) {

·           // Desplazamiento en el eje X

·           x1= x1 + X;

·           x2= x2 + X;

·           // Desplazamiento en el eje X

·           y1= y1 + Y;
```

```
·           y2= y2 + Y;
·       }
```

En este caso no se devuelve ningún valor (tipo devuelto vacío: void).

Por último, el método **obtenerNumRectangulos** simplemente debe devolver el valor del atributo **numRectangulos**. En este caso es razonable plantearse que este método podría ser más bien un método de clase (estático) más que un método de objeto, pues en realidad es una característica de la clase más que algún objeto en particular. Para ello tan solo tendrías que utilizar el modificador de acceso static:

```
public static int obtenerNumRectangulos () {
   return numRectangulos;
}
```

Veamos todo el código:

```
**----------------------------------------------------------------
 * Clase Rectangulo
 ----------------------------------------------------------------*/
public class Rectangulo {

   // Atributos de clase
   private static int numRectangulos;                          // Número total de rectángulos creados
   public static final String nombreFigura= "Rectángulo";      //  Nombre de la clase
   public static final double PI= 3.1416;                      // Constante PI

   // Atributos de objeto
   private String nombre;      // Nombre del rectángulo
   public double x1, y1;       // Vértice inferior izquierdo
   public double x2, y2;       // Vértice superior derecho


   // Método obtenerNombre
   public String obtenerNombre () {
      return nombre;
   }

   // Método establecerNombre
   public void establecerNombre (String nom) {
      nombre= nom;
   }

   // Método CalcularSuperficie
   public double CalcularSuperficie () {
      double area, base, altura;
```

```
    // Cálculo de la base
    base= x2-x1;

    // Cálculo de la altura
    altura= y2-y1;

    // Cálculo del área
    area= base * altura;

    // Devolución del valor de retorno
    return area;
}

// Método CalcularPerimetro
public double CalcularPerimetro () {
    double perimetro, base, altura;

    // Cálculo de la base
    base= x2-x1;

    // Cálculo de la altura
    altura= y2-y1;

    // Cálculo del perímetro
    perimetro= 2*base + 2*altura;

    // Devolución del valor de retorno
    return perimetro;
}

// Método desplazar
public void desplazar (double X, double Y) {

    // Desplazamiento en el eje X
    x1= x1 + X;
    x2= x2 + X;

    // Desplazamiento en el eje X
```

```
        y1= y1 + Y;
        y2= y2 + Y;
    }

    // Método obtenerNumRectangulos
    public static int obtenerNumRectangulos () {
        return numRectangulos;
    }

}
```