2.9. Mensajes

Un objeto solo no es útil, por lo que frecuentemente aparece como una componente de un programa que contiene otros muchos objetos. Es a través de la interacción entre estos objetos como se consigue un alto nivel de funcionalidad del programa. Estos objetos interaccionan y se comunican enviándose *mensajes* unos a otros. El objeto que recibe el mensaje necesita la información adecuada para saber qué debe hacer. Esta información se pasa a través del mensaje como un *parámetro*. Un mensaje está formado por tres componentes:

- El objeto al que se envía el mensaje.
- El nombre del método a utilizar.
- Los parámetros que necesita el método.

Un ejemplo es el que sigue.

```
c3.suma(c1,c2);
```

en el que `c3` es el objeto al que se envía el mensaje, `suma` es el método a utilizar, y `c1` y `c2` son los parámetros que necesita el método.

2.10. Liberación de memoria de objetos no utilizados

Java permite crear objetos sin tener que destruirlos posteriormente. Cuando se crea un objeto, el intérprete de Java reserva suficiente espacio en memoria para él. En particular guardará suficiente memoria para sus variables miembros. Cuando un objeto deja de ser accesible, Java lo destruirá automáticamente y devolverá el espacio reservado para el objeto al espacio de memoria libre, que podrá ser utilizada por nuevos objetos. Este proceso se conoce como "recogida de basura"(*garbage collection*).

Antes de que la memoria asociada a un objeto sea liberada, el sistema llamará a su método `finalize()`. Una clase puede definir su propia finalización redefiniendo el método `finalize()` de la clase `java.lang.Object`. Dicho método debe ser declarado de la forma siguiente:

```
protected void finalize() throws throwable{
 ...
}
```

Práctica 2.1 (Clases Punto y Rectángulo). Crear un clase que se llame **Punto** de la que no se puedan generar subclases y que tenga las características siguientes:

1. Pertenece al paquete **geometric**.
2. Se emplea para definir puntos en el espacio dados por sus coordenadas, por ello habrá dos variables $\boldsymbol{x}$ e $\boldsymbol{y}$ asociadas a la clase que pueden tomar cualquier valor real con la máxima precisión posible (tipo `double`).

3. El constructor recibe dos argumentos que se corresponden con las coordenadas $\boldsymbol{x}$ e $\boldsymbol{y}$ que se quieren dar al punto.

4. Tiene un método llamado **distancia** dentro de la clase **Punto** que recibe como argumentos las coordenadas $\boldsymbol{x}_1$ e $\boldsymbol{y}_1$ de forma que calcula y devuelve la distancia $\boldsymbol{d}$ de ese punto con respecto al punto $\boldsymbol{x}$ e $\boldsymbol{y}$ definido por las variables de instancia asociadas a la clase **Punto**. La distancia se calcula con la fórmula:

$$\boldsymbol{d} = \sqrt{(\boldsymbol{x} - \boldsymbol{x}_1)^2 + (\boldsymbol{y} - \boldsymbol{y}_1)^2}, \tag{2.1}$$

   para la cual existen dos métodos tipo `static` de la clase `Math`, uno llamado `sqrt` que recibe como argumento un número tipo `double` y devuelve la raíz cuadrada en una variable también tipo `double`, y el otro `pow` que recibe dos argumentos tipo `double` y devuelve el valor tipo `double` de la potencia del primer argumento elevado al segundo argumento.

5. También tiene otro método **distancia** con distinto tipo de argumentos, en este caso recibe una variable llamada **p** de la clase **Punto**. Dentro de este método se empleará el método **distancia** definido en el punto 4 para el cálculo y devolución de la distancia $\boldsymbol{d}$.

A continuación crear una clase llamada **Rectangulo** con las siguientes características:

1. Pertenece también al paquete **geometric**.

2. Se emplea para definir un rectángulo y tiene como variables de instancia $\boldsymbol{p}_{il}$, $\boldsymbol{p}_{ur}$, **diagonal**, **ancho** y **alto**, donde $\boldsymbol{p}_{il}$ es un objeto de la clase **Punto** que se corresponde con la esquina inferior izquierda del rectángulo, $\boldsymbol{p}_{ur}$ es un objeto de la clase **Punto** que se corresponde con la esquina superior derecha del rectángulo, y las restantes variables son longitudes de la diagonal, del ancho y del alto del rectángulo, respectivamente. Todas las dimensiones son de tipo `double`.

3. El primer constructor recibe como argumentos un ancho $\boldsymbol{a}_1$ y un alto $\boldsymbol{a}_2$ de forma que la esquina inferior izquierda del objeto rectángulo generado con este constructor sea el origen de coordenadas, mientas que las coordenadas del segundo punto se obtienen a partir del origen y de los datos de ancho y alto. La diagonal queda sin definir.

4. El segundo constructor recibe como argumentos dos puntos $\boldsymbol{p}_1$ y $\boldsymbol{p}_2$, objetos de tipo **Punto**, que se emplearán para dar valores a los puntos $\boldsymbol{p}_{il}$ y $\boldsymbol{p}_{ur}$, respectivamente. Las demás dimensiones quedan sin definir.

5. Por último, se crean tres métodos llamados **calculadiag**, **calculancho** y **calculargo** que se encargan de dar valor a las variables **diagonal**, **ancho** y **alto**, respectivamente, a partir de las coordenadas de los puntos $\boldsymbol{p}_{il}$ y $\boldsymbol{p}_{ur}$.

Para poder emplear estas dos clases crear la clase **Principal** definida en el Comentario 2.2, que pertenezca al paquete **geometric**, y que contiene el método `main()`. En ella se realizan las siguientes acciones:

1. Crear una variable $\boldsymbol{p}_1$ de tipo **Punto** con coordenadas $(0, 0)$.
2. Crear una variable $\boldsymbol{p}_2$ de tipo **Punto** con coordenadas $(30, 40)$.
3. Declarar una variable de tipo **Rectangulo** llamada r_1 pero sin inicializarla.
4. Inicializar la variable r_1 empleando los puntos $\boldsymbol{p}_1$ y $\boldsymbol{p}_2$.
5. Escribir la distancia entre los puntos $\boldsymbol{p}_1$ y $\boldsymbol{p}_2$ usando el método `System.out.println()` y el método correspondiente de la clase **Punto**.
6. ¿Podrías escribir los valores de las variables **diagonal**, **ancho** y **alto** del objeto r_1? ¿Por qué?
7. Escribir los valores de esas variables empleando el método que creáis conveniente.

Solución: La solución de todos los apartados se da a continuación. En primer lugar se describe la clase **Punto** que habrá de ubicarse en un fichero llamado "Punto.java":

```
1. package geometric;

   final class Punto {
```

```
2. double x,y;
```

```
3. public Punto(double x,double y) {
            this.x=x;
            this.y=y;
       }
```

```
4. public double distancia(double x1,double y1) {
            double d;
            d = Math.sqrt(Math.pow(x - x1, 2) + Math.pow(y - y1, 2));
            return (d);
       }
```

```
5. public double distancia(Punto p) {
            double d = this.distancia(p.x,p.y);
            return(d);
       }
   }// Final de la definicion de la clase Punto
```

A continuación se describe la clase **Rectangulo** que habrá de ubicarse en un fichero llamado "Rectangulo.java":

```
1. package geometric;

       public class Rectangulo {
```

```
2. Punto pil,pur;
          double diagonal, ancho, alto;
```

```
3. public Rectangulo(double a1,double a2) {
               pil.x = 0.0;
               pil.y = 0.0;
               pur.x = a1;
               pur.y = a2;
               ancho = a1;
               alto = a2;
          }
```

```
4. public Rectangulo(Punto p1,Punto p2) {
               pil = p1;
               pur = p2;
          }
```

```
5. public void calculadiag () {
               diagonal = pil.distancia(pur);
          }

          public void calculancho () {
               ancho = pil.distancia(pur.x,pil.y);
          }

          public void calculalto () {
               alto = pil.distancia(pil.x,pur.y);
          }
      }// Fin de la definicion de la Rectangulo
```

Para finalizar, se crea la clase **Principal** en un fichero llamado "Principal.java" perteneciente al mismo paquete **geometric**:

```
package geometric;

public class Principal {
    public Principal() {
    }
    public static void main(String[] args) {
```

Todo lo que viene a continuación está ubicado en el cuerpo del método `main`:

```
1. Punto p1=new Punto(0,0);
```

2.
```
Punto p2=new Punto(30,40);
```

3.
```
Rectangulo r1;
```

4.
```
r1 = new Rectangulo(p1,p2);
```

5.
```
System.out.println("La distancia entre los punto p1 y p2 es de "
             + p1.distancia(p2));
```

6. No, porque el constructor empleado no define los valores de las variables **diagonal**, **ancho** y **alto** del objeto r_1, y hasta que no estén definidos sus valores no se puede acceder a ellos.

7.
```
r1.calculadiag();
          r1.calculalto();
          r1.calculancho();
          System.out.println("La diagonal del rectangulo r1 mide " + r1.diagonal
              + " unidades");
          System.out.println("El ancho del rectangulo r1 mide " + r1.ancho
              + " unidades");
          System.out.println("El alto del rectangulo r1 mide " + r1.alto
              + " unidades");
```

■

Ejercicios del Capítulo

Ejercicio 1. Diseñar un programa que dibuje varias figuras geométricas (rectángulos, óvalos, triángulos, pentágonos, hexágonos, etc.) cuyo contorno tenga diferentes grosores y puedan rellenarse de diferentes sombreados y colores.

Se pide: Diseñar un conjunto de clases que lo hagan fácilmente implementable y eficaz para dicho propósito.

Ejercicio 2. En una empresa se quiere trabajar con una base de datos para control de la misma que incluya la información sobre:

- El personal, clasificado por categorías.
- Los clientes, con sus nombres, apellidos, direcciones, teléfonos, números de fax, productos más comprados, etc.
- Productos fabricados, con sus precios, cantidades disponibles en almacén, etc.