# Explicación de la Clase BaseEscenario

La clase `BaseEscenario` es una plantilla para todos los escenarios del juego. Proporciona las bases para la gestión del entorno, como su tamaño, la carga de recursos y la actualización de elementos.

## 1. Definición de la Clase

Ubicación: `juegoprog.escenarios.BaseEscenario`

Tipo: Clase abstracta

Extiende: `JPanel` (permite que cada escenario se dibuje en la interfaz gráfica)

## 2. Explicación Detallada del Código

### Atributos

```java  
protected int ancho;  
protected int alto;  
```

Estos atributos almacenan el ancho y el alto del escenario en píxeles. Son `protected` porque queremos que las clases que hereden de `BaseEscenario` puedan acceder a ellos directamente.

### Constructor

```java  
public BaseEscenario(int ancho, int alto) {  
 this.ancho = ancho;  
 this.alto = alto;  
 setLayout(null);  
}  
```

El constructor recibe el ancho y alto del escenario y los almacena en los atributos. Luego, se usa `setLayout(null);` para permitir que los elementos del escenario se puedan posicionar libremente sin restricciones de un gestor de diseño (`LayoutManager`).

### Método cargar()

```java  
public void cargar() {  
 System.out.println("Cargando escenario...");  
}  
```

Este método es un marcador de posición que puede ser sobrescrito por los escenarios específicos. Actualmente, solo imprime un mensaje, pero en un escenario real podría cargar imágenes, sonidos o estructuras necesarias para el entorno.

### Método actualizar()

```java  
public void actualizar() {  
 System.out.println("Actualizando escenario...");  
}  
```

Este método también es un marcador de posición que puede ser sobrescrito en escenarios concretos. Se usaría para actualizar el estado del escenario, como cambios en la iluminación, movimiento de objetos o eventos dinámicos.

### Método paintComponent(Graphics g)

```java  
@Override  
protected void paintComponent(Graphics g) {  
 super.paintComponent(g);  
}  
```

Este método es clave para la representación gráfica del escenario. Cuando se necesita dibujar el escenario en la ventana, se llama a este método automáticamente.

✔️ `super.paintComponent(g);` limpia la pantalla antes de redibujar, evitando que se acumulen imágenes anteriores.

En escenarios concretos, se puede sobrescribir para dibujar el fondo del escenario con `g.drawImage(...)`.

### Métodos getAncho() y getAlto()

```java  
public int getAncho() { return ancho; }  
public int getAlto() { return alto; }  
```

Estos métodos devuelven el ancho y el alto del escenario. Son útiles para que otras clases puedan consultar el tamaño del escenario sin acceder directamente a los atributos.

## 3. Ejemplo de Uso

Un ejemplo de cómo se utilizaría esta clase en un escenario concreto sería:

```java  
public class EscenarioDistritoSombrio extends BaseEscenario {  
 public EscenarioDistritoSombrio(int ancho, int alto) {  
 super(ancho, alto);  
 }  
  
 @Override  
 public void cargar() {  
 System.out.println("Cargando distrito sombrío...");  
 }  
  
 @Override  
 public void paintComponent(Graphics g) {  
 super.paintComponent(g);  
 g.drawImage(new ImageIcon("fondo.png").getImage(), 0, 0, this);  
 }  
}  
```

En este caso, `EscenarioDistritoSombrio` hereda de `BaseEscenario`, sobrescribe `cargar()` para personalizar su contenido y dibuja un fondo en `paintComponent()`.

## 4. Conclusión

La clase `BaseEscenario` proporciona una estructura base para todos los escenarios del juego, facilitando la organización del código y la reutilización de lógica común. Es necesario que las clases que hereden de ella sobrescriban los métodos `cargar()` y `paintComponent()` para adaptar el escenario a sus necesidades.