**Análisis de la Caja Blanca de clase Shot**

Clase **Shot**:

* Método initShot(x, y)

|  |  |
| --- | --- |
| 1 | private void initShot(int x, int y) { |
| 2 |  |
| 3 | var shotImg = "src/main/resources/images/shot.png"; |
| 4 | var ii = new ImageIcon(shotImg); |
| 5 | setImage(ii.getImage()); |
| 6 |  |
| 7 | int H\_SPACE = 6; |
| 8 | setX(x + H\_SPACE); |
| 9 |  |
| 10 | int V\_SPACE = 1; |
| 11 | setY(y - V\_SPACE); |
| 12 | } |

Imagen en blanco y negro

Descripción generada automáticamente con confianza baja

Dado que no existen bifurcaciones, condiciones, o ciclos, todos los pasos en este diagrama se ejecutan de manera continua de principio a fin.

*Este tipo de estructura puede simplificarse en un único nodo.*

**Complejidad ciclomática:**

El número de regiones del grafo es igual a la complejidad ciclomática = 1

**Caminos independientes:**

Inicio → 3,4,5 → 7,8 → 10,11 → Fin

En conclusión, el método initShot, debido a su naturaleza secuencial y su capacidad para representarse como un único nodo en el grafo de flujo, presenta una complejidad ciclomática de uno. Esto significa que solo existe un camino posible de ejecución, sin bifurcaciones ni ramificaciones.

Como resultado, no se vio necesario diseñar casos de prueba específicos en JUnit5 para cubrir múltiples escenarios, ya que la simple invocación del método garantiza la cobertura completa de su flujo.