Universidad Distrital Francisco José de Caldas

Ingeniería de Sistemas

Programación Avanzada – Taller No. 2

Taller: Simulación del juego La Argolla Llanera

Profesor: Jhon Herrera Cubides

1. Introducción

Este documento presenta los resultados de las pruebas unitarias JUnit aplicadas a dos clases del paquete control del proyecto Argolla Llanera. El objetivo de las pruebas es garantizar el correcto funcionamiento de la lógica del programa antes de la integración con la interfaz gráfica.

2. Clases probadas

Clase	Descripción	Clase de prueba
ControlEquipo	Gestiona la creación y administración de equipos, jugadores y puntajes.	ControlEquipoTest
ControlTiro	Controla los tipos de lanzamiento (tiros), sus puntajes y validaciones.	ControlTiroTest

3. Detalle de pruebas unitarias

Clase: ControlEquipoTest

Método probado	Propósito	Resultado esperado	Estado
testCrearEquipo()	Crea un nuevo equipo y retorna su índice.	Índice 0 y nombre correcto.	✅
testAgregarEquipoYObtenerTamaño()	Agrega un equipo y verifica el tamaño.	Tamaño = 1.	✅

testAgregarJugador()	Añade un jugador a un equipo.	Jugador registrado correctamente.	✅
testAgregarPuntosYObtenerPuntaje()	Suma puntos al equipo.	Puntaje = 5.	✅
testResetearPuntaje()	Reinicia puntajes de todos los equipos.	Puntaje = 0.	✅
testBorrarTodo()	Elimina todos los equipos.	Lista vacía.	✅
testObtenerNombresJugador()	Devuelve nombre y apodo del jugador.	Pedro / El Toro.	✅
testSetYGetEquipos()	Verifica asignación de lista de equipos.	Valor conservado.	✅
testRemoveEquipo()	Elimina un equipo por índice.	Tamaño disminuye.	✅

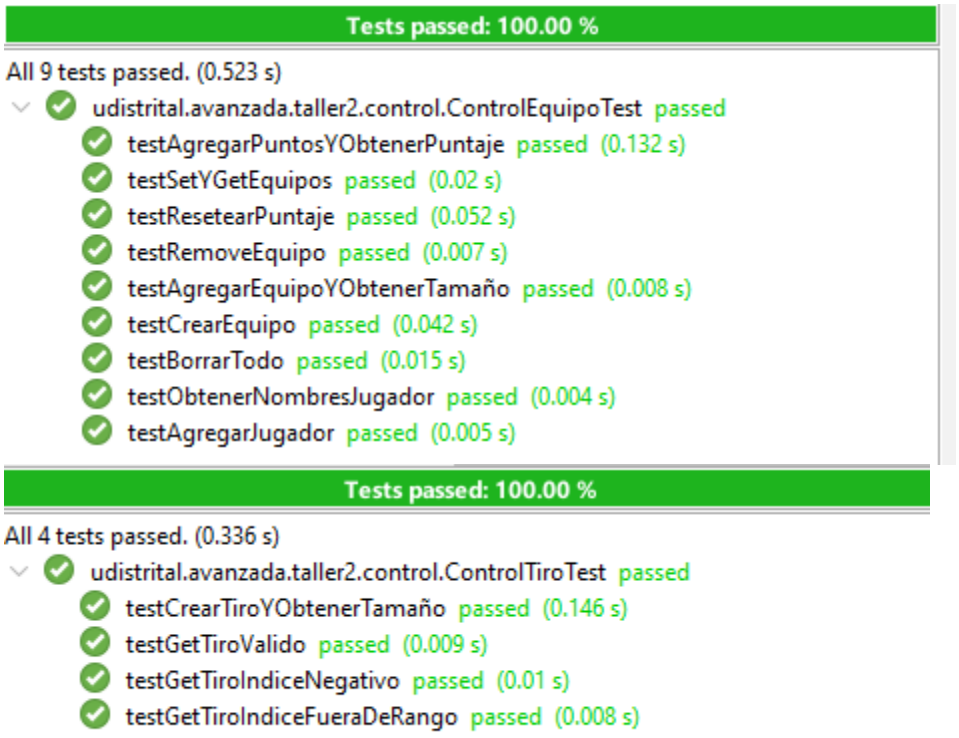
Clase: ControlTiroTest

Método probado	Propósito	Resultado esperado	Estado
testCrearTiroYObtenerTamaño()	Crea tiros y verifica tamaño de la lista.	Lista con 2 tiros.	✅
testGetTiroValido()	Obtiene un tiro válido y revisa datos.	Moñona / 8 puntos.	✅
testGetTiroIndiceNegativo()	Solicita índice negativo.	Retorna null.	✅
testGetTiroIndiceFueraDeRango()	Solicita índice	Retorna null.	✅

	inexistente.		

4. Evidencias gráficas

A continuación, se presentan los pantallazos de la ejecución de las pruebas en NetBeans:




```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running udistrital.avanzada.taller2.control.ControlEquipoTest
Tests run: 9, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.412 s -- in udistrital.avanzada.taller2.control.ControlEquipoTest

Results:

Tests run: 9, Failures: 0, Errors: 0, Skipped: 0

------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  9.053 s
```

```java
/**
 * Método ejecutado una sola vez antes de todas las pruebas.
 * Se usa para inicializar recursos globales
 */
@BeforeAll
static void setUpBeforeAll() {
}

/**
 * Método ejecutado una sola vez después de todas las pruebas.
 * Se usa para liberar recursos o imprimir mensajes de finalización.
 */
@AfterAll
static void tearDownAfterAll() {
}
/**
 * Método que se ejecuta antes de cada prueba individual.
 * Crea un nuevo objeto ControlEquipo limpio para cada test.
 */
@BeforeEach
void setUp() {
    control = new ControlEquipo();
}

/**
 * Método que se ejecuta después de cada prueba individual.
 * Libera los recursos usados en la prueba actual.
 */
@AfterEach
void tearDown() {
    control = null;
}

/**
 * Comprueba que el método ControlEquipo resetearPuntaje reinicie
 * correctamente los puntajes de todos los equipos a cero.
 */
@Test
void testResetearPuntaje() {
    int indice = control.crearEquipo("Llaneros");
    control.agregarPuntos(indice, 10);
    control.resetearPuntaje();
    assertEquals(0, control.obtenerPuntaje(indice),
                 "El puntaje debe reiniciarse a cero");
}
```

Uso del @BeforeAll, @AfterAll, @BeforeEach, @AfterEach, @Test

5. Resultados globales

Todas las pruebas fueron ejecutadas exitosamente sin fallos. Los controladores del sistema funcionan de acuerdo a la lógica esperada. Las pruebas permiten validar la estabilidad y confiabilidad de la capa de control.