**DISEÑO DE PRUEBAS**

**Clase Bonificación**

**Clase Nave**

**Clase Proyectil**

**Clase Pelota**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prueba: Verifica que el método de hayColision retorna efectivamente si existe o no Colision | | | | |
| Clase | Método | Escenario | Valores de entrada | Resultado |
| Juego | hayColision(Colisionable): boolean | Hay una pelota con ubicación (50.,50) | Un Proyectil con x=50 y=50 | True |
| Juego | hayColision(Colisionable):  boolean | Hay una pelota con ubicación (50.,50) | Un Proyectil con x=100 y=100 | False |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prueba: Verifica que el método de colisionaCon realiza su respectiva acción dependiendo del objeto Colisionable recibido | | | | |
| Clase | Método | Escenario | Valores de entrada | Resultado |
| Juego | colisionaCon(Colisionable): void | Hay una pelota | Un Proyectil Normal | Vida = 2 |
| Juego | colisionaCon(Colisionable): void | Hay una pelota | Un Proyectil Rápido | Vida = 3 |
| Juego | colisionaCon(Colisionable): void | Hay una pelota | Un Proyectil Fuerte | Vida = 0  Visible = false |
| Juego | colisionaCon(Colisionable): void | Hay una pelota | Una nave | Vida = 5 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prueba: Verifica que el método de existenColisiones verifica exitosamente si existe alguna colisión | | | | |
| Clase | Método | Escenario | Valores de entrada | Resultado |
| Juego | existenColisiones(Colsionable): boolean | El árbol de pelotas tiene 3 pelotas ubicadas en  Pelota 1: (50,50)  Pelota 2: (100,100)  Pelota 3: (40,100) | Un Proyectil Normal ubicado en (50,50) | true |
| Juego | existenColisiones(Colsionable): boolean | El mismo anterior | Un Proyectil Normal ubicado en (100,100) | true |
| Juego | existenColisiones(Colsionable): boolean | El mismo anterior | Un Proyectil Normal ubicado en (40,100) | true |
| Juego | existenColisiones(Colsionable): boolean | El mismo anterior | Un proyectil normal ubicado en (10,20) | false |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prueba: Verifica que el método de hayVivas verifica exitosamente si existe alguna Pelota viva | | | | |
| Clase | Método | Escenario | Valores de entrada | Resultado |
| Juego | hayVivas(): boolean | El árbol de pelotas tiene 3 pelotas  Pelota 1: visible  Pelota 2: visible  Pelota 3: visible | Ninguno | true |
| Juego | hayVivas(): boolean | El árbol de pelotas tiene 3 pelotas  Pelota 1: visible  Pelota 2: invisible  Pelota 3: visible | Ninguno | true |
| Juego | hayVivas(): boolean | El árbol de pelotas tiene 3 pelotas  Pelota 1: visible  Pelota 2: visible  Pelota 3: invisible | Ninguno | true |
| Juego | hayVivas(): boolean | El árbol de pelotas tiene 3 pelotas  Pelota 1: invisible  Pelota 2: invisible  Pelota 3: invisible | Ninguno | false |

**Clase Juego**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prueba: Verifica que el método de iniciarJuego inicia el juego de manera correcta | | | | |
| Clase | Método | Escenario | Valores de entrada | Resultado |
| Juego | iniciarJuego(boolean cargado): void | Un juego recién creado  Jugador  Nickname: Armando | False | Crea una nave nueva  El número de pelotas es 3  Hay tres pelotas en el árbol  jugando= true  La cantidad de decoraciones es 5 |
| Juego | iniciarJuego(boolean cargado): void | Un juego en nivel 2 nuevo  Jugador Nickname:  Armando | False | Crea una nave nueva  El número de pelotas es 4  Hay cuatro pelotas en el árbol  jugando= true |
| Juego | iniciarJuego(boolean cargado): void | Inicia un juego ya existente | True | La nave es la nave ya existente  El nickname del jugador es el ya existente  El puntaje del jugador es el ya existente  El nivel del jugador es el ya existente |
| Juego | iniciarJuego(boolean cargado): void | Inicia un juego nuevo con un jugador ya insertado  Jugador nickname: Camila | False | Lanza Excepcion JugadorRepetidoException |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prueba: Verifica que el método de darDecoraciones retorna las mismas decoraciones que hay en la lista decoraciones. | | | | |
| Clase | Método | Escenario | Valores de entrada | Resultado |
| Juego | darDecoraciones():  ArrayList<Decoracion> | Se crea un juego  Se llama al método iniciarJuego: false | Ninguna | El size del ArrayList obtenido debe ser igual al número de elementos de la lista decoraciones. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prueba: Verifica que el método de subirNivel crea el número de Pelotas correspondientes al nivel alcanzado | | | | |
| Clase | Método | Escenario | Valores de entrada | Resultado |
| Juego | subirNivel() | Crea un juego  Llama al método iniciarJuego(false) | Ninguno | El peso del árbol pelotas es 3 |
| Juego | subirNivel() | Crea un juego  Llama al método iniciarJuego(false)  Se llama al método subir niveldos veces (se encuentra en nivel 3) | Ninguno | El peso del árbol pelotas es 4 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prueba: Verifica que el método de crearBonus añade un bonus a la lista bonus | | | | |
| Clase | Método | Escenario | Valores de entrada | Resultado |
| Juego | crearBonus() | Crea un juego  Llama al método iniciarJuego(false) | Ninguno | El primerBonus es diferente de null |
| Juego | crearBonus() | Crea un juego  Llama al método iniciarJuego(false)  Llama al método crearBonus | Ninguno | El siguiente del primerBonus es diferente de null |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prueba: Verifica que el método de addJugador añade correctamente un Jugador | | | | |
| Clase | Método | Escenario | Valores de entrada | Resultado |
| Juego | addJugador(): void | Raizjugador = null  Jugador =  Nickname: “Joan”  Puntos: 100  Nivel: 1 | Ninguna | Raizjugador es igual el jugador |
| Juego | addJugador(): void | Raizjugador= Joan  Jugador =  Nickname: “Camila”  Puntos: 50  Nivel: 1 | Ninguna | Camila es el subárbol izquierdo de Joan |
| Juego | addJugador(): void | Raizjugador= Joan  Jugador =  Nickname: “Sariana”  Puntos: 500  Nivel: 4 | Ninguna | Sariana es el subárbol derecho de Joan |
| Juego | addJuagdor(): void | Raizjugador = Joan  Jugador =  Nickname: “Joan”  Puntos: 200  Nivel: 1 | Ninguna | Lanza JugadorRepetidoException |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prueba: Verifica que el método de mostrar la lista ordenada por puntaje de manera ascendente y descendente funciona bien. | | | | |
| Clase | Método | Escenario | Valores de entrada | Resultado |
| Juego | ordernarPuntajeAscendente(): ArrayList<Jugador> | Hay un árbol binario donde la raíz es  Nickname: “Julian”  Puntos: 100  Nivel: 3  Y en conjunto el árbol tiene los siguientes elementos  Nickname: “Javier”  Puntos: 200  Nivel: 4  Nickname: “Alejandro”  Puntos: 50  Nivel: 2  Nickname: “Manyolml”  Puntos: 500  Nivel: 7  Nickname: “JuanMa”  Puntos: 400  Nivel: 6 | Ninguna | Alejandro  Julian  Javier  JuanMa  Manyolml |
| Juego | ordernarPuntajeDescendente(): ArrayList<Jugador> | El mismo de arriba | Ninguna | Manyolml  JuanMa  Javier  Julian  Alejandro |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prueba: Verifica que el método de mostrar la lista ordenada por nombre de manera ascendente y descendente funciona bien. | | | | |
| Clase | Método | Escenario | Valores de entrada | Resultado |
| Juego | ordernarNombreAscencente(): ArrayList<Jugador> | El mismo de arriba | Ninguna | Manyolml  Julian  JuanMa  Javier  Alejandro |
| Juego | ordernarNombreDescencente(): ArrayList<Jugador> | El mismo de arriba | Ninguna | Alejandro  Javier  JuanMa  Julian  Manyolml |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prueba: Verifica que el método de mostrar la lista ordenada por nivel de manera ascendente y descendente funciona bien. | | | | |
| Clase | Método | Escenario | Valores de entrada | Resultado |
| Juego | ordernarNivelAscencente(): ArrayList<Jugador> | El mismo de arriba | Ninguna | Alejandro  Julian  Javier  JuanMa  Manyolml |
| Juego | ordernarNivelAscencente(): ArrayList<Jugador> | El mismo de arriba | Ninguna | Manyolml  JuanMa  Javier  Julian  Alejandro |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prueba: Verifica que el método de buscar un jugador por puntaje encuentra al jugador correspondiente | | | | |
| Clase | Método | Escenario | Valores de entrada | Resultado |
| Juego | buscarJugadorPuntos(int puntos): Jugador | El mismo de arriba | 500 | Retorna a Manyo |
| Juego | buscarJugadorPuntos(int puntos): Jugador | El mismo de arriba | 2 | Lanza PuntajeNoExisteException |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prueba: Verifica que el método de buscar un jugador por puntaje encuentra al jugador correspondiente | | | | |
| Clase | Método | Escenario | Valores de entrada | Resultado |
| Juego | BuscarJugadorNombre(String nombre): Jugador | El mismo de arriba | “Javier” | Retorna al jugador “Javier” |
| Juego | buscarJugadorNombre(String nombre): Jugador | El mismo de arriba | “Karol” | Lanza NombreNoExisteException |