|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| DISEÑO DE CASOS DE PRUEBAS | | | | |
| Clase | Método | Escenario | Valores de entrada | Resultado |
| Prueba No. 1 | **Objetivo**: Probar que el método permite crear y agregar una partida nueva al ABB (de acuerdo al nombre) en caso de que no existen partidas con el mismo nombre, de lo contrario lanza excepción PartidaYaExisteException. | | | |
| Partida | agregarPartida (Partida nodo) | No hay ninguna partida en el ABB | nodo: Partida | Permite agregar la partida sin problema. |
| Hay varias partidas en el ABB | nodo: Partida  diferente a los que existen | Permite agregar la partida, a la derecha porque tiene un nombre mayor que la raíz. |
| nodo: Partida  Igual a uno de los que existen | Lanza Excepción, debido a que la partida con ese nombre ya existe previamente. |
| Prueba No.2 | **Objetivo:** Probar que el método permite buscar una partida en el ABB. | | | |
| Partida | buscarPartida (String nombre) | No hay ninguna partida en el ABB | nombre: String | Retorna “null” porque no hay ninguna partida a la cual buscar. |
| Hay varias partidas en el ABB | nombre: String | Retorna la partida a la cual le corresponde el nombre pasado por parámetro. |
| Prueba No.3 | **Objetivo**: Probar que el método permite iniciar todos los DIFERENTES enemigos. | | | |
| Partida | inicializarEnemigos() | Existe la matriz de enemigos, sin ningún elemento | - | “Crea” e inicializa todas las posiciones de la matriz con diferentes enemigos. |
| Prueba No.4 | **Objetivo**: Probar que el método permite eliminar un enemigo (aquel que fue dado de baja por nuestro héroe) | | | |
| Partida | eliminarUnEnemigo (boolean elimino, Enemigo a) | La matriz de enemigos existe y contiene elementos dentro de ella | elimino: boolean, boolean que representa que un enemigo fue disparado  a: Enemigo, el enemigo al cual hay que eliminar de la matriz | Busca en la matriz al enemigo que le entra por parámetro y se encarga de eliminarlo de la matriz (volverlo null). |
| Prueba No.5 | **Objetivo**: Probar que el método verifica si la partida termino (Si nuestro héroe le gano a los invasores). | | | |
| Partida | terminarNivel() | Queda un solo enemigo en la matriz | - | Se encarga de verificar si se elimina al último enemigo invasor, retorna true si ocurre esto, de lo contrario retorna false. |
| Prueba No.6 | **Objetivo:** Probar que el método cambiar de nivel y a su vez lo inicializa. | | | |
| Partida | nivelCompleto() | El nivel actual es el nivel 1. | - | Si se acabó el juego cambia de nivel y lo inicializa |
| Prueba No.7 | **Objetivo:** Probar que el método elimina la partida que tiene el mismo nombre que recibe como parámetro, modificando el ABB de partidas hecho previamente. | | | |
| Partida | eliminar(String nombre) | Existen partidas en el ABB. preCondicion: La partida que se desea eliminar se encuentra en el árbol | nombre: String, nombre de la partida que se desea eliminar | Elimina la partida que tenga el mismo nombre. |
| Prueba No.8 | **Objetivo:** Probar que el método verifica si una partida es una “hoja” o no, es decir si sus hijos izquierda y derecha son nulos | | | |
| Partida | esHoja() |  |  |  |
| Prueba No.9 | **Objetivo:** Probar que el método verifica si la bala del jugador golpeo a un enemigo. | | | |
| Disparo | hitsEnemigo(Enemigo: a) | Se crea un enemigo (manipulando sus posiciones y su anchura), se crea una bala para el jugador (con las mismas posiciones de la nave enemiga) | a: Enemigo, el cual será impactado por la bala de nuestro héroe | Retorna true porque la bala de nuestro héroe impacto a un enemigo. |
| Se crea un enemigo (manipulando sus posiciones y su anchura), se crea una bala para el jugador (con DIFERENTES posiciones de la nave enemiga) | a: Enemigo, el cual NO será impactado por la bala de nuestro héroe | Retorna false, porque la bala de nuestro héroe (no tan fino con su puntería) no le dio al enemigo. |
| Prueba No.10 | **Objetivo:** Probar que el método verifica si la bala de un enemigo golpeo a nuestro héroe. | | | |
| Disparo | hitsJugador(NaveJugador: a) | Se crea una bala para un enemigo (manipulando sus posiciones y su anchura) y colocamos al jugador en la misma posición de la bala. | a: NaveJugador, el cual será impactado por la bala de un enemigo | Retorna true porque la bala del enemigo impacto a nuestro héroe. |
| Se crea una bala para un enemigo (manipulando sus posiciones) y colocamos al jugador en una posición diferente de la bala. | a: NaveJugador, el cual NO será impactado por la bala de un enemigo | Retorna false, porque la bala del enemigo (no tan fino) no le dio a nuestro héroe. |
| Prueba No.11 | **Objetivo:** Probar que el método mueve a un enemigo (izquierda o derecha). | | | |
| Enemigo | mover(int dir) | Existe un enemigo con una posición en X definida y una velocidad de movimiento definida. | dir: int, valor que representa cuanto se desea mover y en qué dirección (1 Derecha) (-1 Izquierda). | Cambia la posición X de un enemigo. |
| Prueba No.12 | **Objetivo:** Probar que el método mueve a un enemigo (abajo). | | | |
| Enemigo | moverAbajo(int dir) | Existe un enemigo con una posición en Y definida. | dir: int, valor que representa cuanto se desea mover hacia abajo (1 hacia abajo). | Cambia la posición de Y de un enemigo si este ha tocado un borde. |
| Prueba No.13 | **Objetivo:** Este método se encarga de verificar si un enemigo ha tocado un borde. | | | |
| Enemigo | edge() | Existe un enemigo que está en un borde. | - | Retorna true si el enemigo está tocando algún borde, de lo contrario retorna false. |
| Prueba No.14 | **Objetivo:** Probar que el método mueve al jugador en dos direcciones (izquierda o derecha) | | | |
| Nave | mover(int dir) | Existe una NaveJugador con una posición en X definida y una velocidad definida (lo que permite el movimiento) | dir: int, valor que representa cuanto se desea mover y en qué dirección (1 Derecha) (-1 Izquierda). | Cambia la posición en X del jugador dependido del valor que le entre como parámetro. |
| Prueba No.15 | **Objetivo:** Probar que el método crea le permite crear un disparo a un enemigo o al jugador. | | | |
| Nave | disparar(int posX, int posY) | Existe un enemigo o jugador al cual asignarle un disparo | posX: int; posY: int, parámetros que representan la posición en donde se crea la bala | La bala del jugador o enemigo ha sido creada. |
| Prueba No.16 | **Objetivo:** Probar que el método elimina el disparo hecho ya sea por un enemigo o el jugador. | | | |
| Nave | eliminarDisparo() | Existe un enemigo o jugador al cual asignarle un disparo | - | Cambia el disparo del jugador o enemigo a null. |
| Prueba No.17 | **Objetivo:** Probar que el método verifique si el jugador o enemigo está vivo o no. | | | |
| Nave | estaViva() | Existe un jugador o enemigo vivo, que tiene 3 unidades de vida. | - | Retorna true porque el jugador aún sigue vivo. |
| Prueba No.18 | **Objetivo:** Probar que el método crea un jugador y lo agrega al vector (en el caso de que el vector este vacío o no haya ningún jugador con el mismo Nickname que pasa como parámetro), de lo contrario lanza NicknameYaExisteException. | | | |
| SpaceInvaders | agregarJugador(String nombre, String nickname) | El vector esta creado y NO existe ningún jugador en el vector | nombre: String, nombre del jugador.  nickname: String, sobre nombre del jugador con el cual se identificara siempre. | Agrega sin ningún problema al vector |
| Existen jugadores en el vector, pero ninguno tiene el mismo nickname del que se va agregar | Agrega sin ningún problema al vector |
| Existen jugadores en el vector, y hay un jugador que tiene el mismo nickname del que se quiere agregar |  |
| Prueba No.19 | **Objetivo:** Probar que el método busca y retorna a un jugador dependiendo del nickname que le llega por parámetro. | | | |
| SpaceInvaders | buscarJugador(String nickname) | - | nickname: String, sobre nombre del jugador con el cual se identificara siempre. | Retorna al jugador que está en el vector que tiene el mismo nickname que pasa por parámetro, si este no existe retorna null |
| Prueba No.20 | **Objetivo:** Probar que el método consulta el turno siguiente, si hay turnos en el programa | | | |
| SpaceInvaders | ordenarPorNickname() | Existe un vector de jugadores creado e inicializado | - | Retorna un nuevo arraylist de jugadores ordenado por el método de inserción. |
| Prueba No.21 | **Objetivo:** Probar que el método busca de FORMA BINARIA a un jugador a través del nickname | | | |
| SpaceInvaders | busquedaRapida(String nickname) | Existe un vector de jugadores creado, con elementos dentro de él. | nickname: String, nickname con el que se buscara al jugador de manera más rápida. | Debe de retornar true porque el jugador con ese nickname se encuentra en el vector |
| nickname: String, nickname de jugador ya existente en el vector | Debe retornar false porque el jugador con ese nickname no se encuentra en el vector |
| Prueba No.22 | **Objetivo:** Probar que el método agrega a la lista doblemente enlazada de puntajes, un puntaje nuevo | | | |
| SpaceInvaders | agregarPuntaje(Puntaje puntaje) | No existe ningún puntaje previo en la lista | puntaje: Puntaje, el puntaje que se desea agregar a la lista | Debe dejarlo agregar sin problema al puntaje. |
| Si existe algunos puntajes previo en la lista |