AlmacenTest

Id. De la prueba	Objetivo	Entrada	Condiciones de ejecución[1]	Resultado esperado	Resultado obtenido	Comentarios
cargarXML()	Leer el fichero para asegurarnos de su correcta lectura	String pDif	El sistema arranca al leer dicho fichero. Nos aseguramos de que el fichero existe en el propio método.	Fichero inicializado	Fichero correctamente inicializado	Este método tiene la etiqueta @BeforeClass que lanzara su ejecución previa al resto de test
getMyAlmacen()	Asegurar el patrón Singleton	-	getMyAlmacen()	notNull()	notNull()	-
comprarArma()	Comprobar que se accede al arma y que es posible comprarla	String pArma	El fichero XML ha sido leído previamente	Equals()	true	-
getPrecioArma()	Comprobar que las armas tienen precio y es accesible	String pArma	El fichero XML ha sido leído previamente	Equals()	true	-
getCantidadRestante()	Comprobar que podemos saber cuantas armas de cada tipo quedan	-	El fichero XML ha sido leído previamente	Equals()	true	-

	en el almacen					

[1] En qué estado se encuentra el sistema para que esa entrada cumpla el objetivo de la prueba

ArmaFactoryTest

Id. De la prueba	Objetivo	Entrada	Condiciones de ejecución[1]	Resultado esperado	Resultado obtenido	Comentarios
setUp()	Inicializar varias amas para que en los siguientes métodos no halla problemas	-	El sistema ejecutará este método con etiqueta (@Before Method) antes de cada prueba	-	-	-
tearDown()	Cierre de la prueba	-	-	-	-	-
getArmaFactory()	Asegurar el patrón Singleton	-	-	NotNull()	NotNull()	-
getType()	Asegurar que somos capaces de manejar las armas a través de su tipo de clase	ArmapArma	El sistema dispone de un patrón Singleton en ArmaFactory()	-	-	-
crearArma()	Probar que la factoria de armas es capaz de generar un arma del tipo especificado	StringpArma	El sistema dispone de un patrón Singleton en ArmaFactory()	NotNull() & Equals(type)	true & true	Comprobamos que las armas se generan y tras esto, nos aseguramos de que cumplen con el tipo especificado en su creación

[1] En qué estado se encuentra el sistema para que esa entrada cumpla el objetivo de la prueba

BarcoFactoryTest

Id. De la prueba	Objetivo	Entrada	Condiciones de ejecución [1]	Resultado esperado	Resultado obtenido	Comentarios
setUp()	Inicializar los objetos necesarios para realizar el test.			Los objetos se crearán correctamente.	Los barcos se crean según lo esperado.	
testGetBarcoFactory()	Obtener el elemento BarcoFactory.			El elemento BarcoFactory es diferente de null.	El elemento cumple los requisitos establecidos.	
testCrearBarco()	Se comprobará que los barcos creados en el setUp() se crean correctamente.			Los barcos creados son diferentes de null.	Los barcos se crean correctamente.	

[1] En qué estado se encuentra el sistema para que esa entrada cumpla el objetivo de la prueba

BarcoTest

Id. De la prueba	Objetivo	Entrada	Condiciones de ejecución [1]	Resultado esperado	Resultado obtenido	Comentarios
setUp()	Crear los objetos necesarios para realizar la prueba.			Crear los objetos necesarios satisfactoriamente.	Los objetos se crear correctamente.	
getTamaño()	Obtener los tamaños de los diferentes barcos.			Los tamaños de los barcos son igual al tamaño establecido.	Los barcos tienen el tamaño correcto.	
getPrecioReparacion()	Obtener el precio de reparación de cada casilla.			Los precios de reparación corresponden con los establecidos..	Los precios se establecen correctamente.	
estaDañado()	Obtener el estado del barco.			El estado del será: no dañado.	El estado del barco al crearlo es el establecido.	

[1] En qué estado se encuentra el sistema para que esa entrada cumpla el objetivo de la prueba

DireccionesArmasTest

Id. De la prueba	Objetivo	Entrada	Condiciones de ejecución [1]	Resultado esperado	Resultado obtenido	Comentarios
testGetDIreccion()	Comprobar que las opciones de la clase DireccionesArma se crean correctamente.			Las opciones se crean correctamente.	Las direcciones se crean según lo establecido.	

[1] En qué estado se encuentra el sistema para que esa entrada cumpla el objetivo de la prueba

GestorFicherosTest

Id. De la prueba	Objetivo	Entrada	Condiciones de ejecución[1]	Resultado esperado	Resultado obtenido	Comentarios
lanzarLeerXML()	Leer el fichero para asegurarnos de su correcta lectura	String pDif	El sistema arranca al leer dicho fichero. Nos aseguramos de que el fichero existe en el propio método.	Fichero inicializado	Fichero correctamente inicializado	Este método tiene la etiqueta @BeforeClass que lanzara su ejecución previa al resto de test
getMyGestorFicheros()	Asegurar el patrón Singleton	-	getMyGestorFicheros()	notNull()	notNull()	-
getNumBombas()	Comprobar que se inicializan las bombas	-	El fichero XML ha sido leído previamente	99	99	-
getNumMisiles()	Comprobar que se inicializan los misiles	-	El fichero XML ha sido leído previamente	15	15	-
getNumMisilesDirig()	Comprobar que se inicializan los misiles dirigidos	-	El fichero XML ha sido leído previamente	8	8	-
getNumRadares()	Comprobar que se inicializan las radares	-	El fichero XML ha sido leído previamente	10	10	-

getNumEscudos()	Comprobar que se inicializan los escudos	-	El fichero XML ha sido leído previamente	10	10	-
getPrecioMisiles()	Comprobar que se inicializan los misiles	-	El fichero XML ha sido leído previamente	2500	2500	-
getPrecioMisilesDirig()	Comprobar que se inicializan los misiles dirigidos	-	El fichero XML ha sido leído previamente	8000	8000	-
getPrecioRadares()	Comprobar que se inicializan los radares	-	El fichero XML ha sido leído previamente	5000	5000	-
getPrecioEscudos()	Comprobar que se inicializan los escudos	-	El fichero XML ha sido leído previamente	5000	5000	-
getObtenerPrecioReparacion()	Comprobar que se inicializa el precio de reparación de los barcos	-	El fichero XML ha sido leído previamente	2500	2500	-

[1] En qué estado se encuentra el sistema para que esa entrada cumpla el objetivo de la prueba

HumanoTest

Id. De la prueba	Objetivo	Entrada	Condiciones de ejecución[1]	Resultado esperado	Resultado obtenido	Comentarios
setUp()	Crear los objetos 'ia' para las pruebas	String pDif	El sistema crea el objeto. EL fichero XML ha sido leído previamente.	ia not null	ia not null	Este método tiene la etiqueta @BeforeMethod que lanzara su ejecución previa al cada uno de los test
tearDown()	-	-	El sistema ha finalizado la ejecución de un test unitario	null	null	-
getDificultad()	Comprobar que la IA se crea teniendo en cuenta la dificultad de la partida cargada desde el fichero XML	-	El jugador ha sido previamente creado	"fácil"	"facil"	-

[1] En qué estado se encuentra el sistema para que esa entrada cumpla el objetivo de la prueba

IATest

Id. De la prueba	Objetivo	Entrada	Condiciones de ejecución[1]	Resultado esperado	Resultado obtenido	Comentarios
setUp()	Crear los objetos 'ia' para las pruebas	String pDif	El sistema crea el objeto. EL fichero XML ha sido leído previamente.	ia not null	ia not null	Este método tiene la etiqueta @BeforeMethod que lanzara su ejecución previa al cada uno de los test
tearDown()	-	-	El sistema ha finalizado la ejecución de un test unitario	null	null	-
getDificultad()	Comprobar que la IA se crea teniendo en cuenta la dificultad de la partida cargada desde el fichero XML	-	El jugador ha sido previamente creado	"fácil"	"facil"	-

[1] En qué estado se encuentra el sistema para que esa entrada cumpla el objetivo de la prueba

ListaArmasTest

Id. De la prueba	Objetivo	Entrada	Condiciones de ejecución [1]	Resultado esperado	Resultado obtenido	Comentarios
setUp()	Inicializar un objeto ListaArmas para utilizarlo en el test.			Crear un objeto ListaArmas e inicializarlo correctamente.	El objeto se crea e inicializa según lo previsto.	
testInicializarArmas()	Comprobar que la lista de armas creada en el setUp se crea correctamente.			La lista de armas es diferente de null.	La lista de armas se crea correctamente.	
testGetArma()	Comprobar que podemos obtener un arma de cada tipo.			La lista de armas tiene un campo reservado para cada tipo de arma, y nos devuelve una de cada uno de ellos correctamente.	La lista de armas contiene armas de cada tipo.	
getType()	Comprobar que nos devuelve correctamente el tipo de cada arma.	Arma pArma	El sistema obtiene un arma de cualquiera de los tipos posibles, y nos devolverá el tipo de la misma.	La prueba nos devuelve el tipo del arma introducida.	Obtenemos correctamente el tipo del arma introducida.	

[1] En qué estado se encuentra el sistema para que esa entrada cumpla el objetivo de la prueba