Configuración de los Escenarios

Nombre	Clase	Escenario
serializedGameManage	GameManager	Crea un archivo con un objeto serializado de la clase GameManager
randomPlayer	GameManager	Un objeto Player
randomLog	Log	un objeto de la clase Log
randomUser	User	Un objeto de la clase User (left = null, right = null, parent = null, username = "Chris",
userABB	GameManager	

Diseño de Casos de prueba

Objetivo: Verificar que el constructor y los métodos getter de la clase GameManager funciona de manera correcta, es decir, que se le asignen de manera correcta los atributos del objeto creado. Adicionalmente, verificar que se mantenga la persistencia del modelo en caso de existir la serialización.

Clase	Método	Escenario	Valores de Entrada	Resultado
GameManager	GameManager()	ninguno	ninguno	Un objeto de la clase GameManager (saves = null, match = null)
GameManager	GameManager()	serializedGameManager	ninguno	Un objeto de la clase GameManager con los atributos del objeto serializado
GameManager	loadGameManag	serializedGameManager	ninguno	jeto GameManager de los archiv
GameManager	loadGameManag	ninguno	ninguno	a null al no haber archivos seria

Objetivo: Verificar que el método newMatch(): void funciona correctamente, es decir, crea un nuevo match

Clase	Método	Escenario	Valores de Entrada	Resultado
GameManager	newMatch(): void	Ninguno	Ninguno	Se le asigna correctamente a la asociación match un objeto de la clase Player

Objetivo: Verificar que el método loadMatch(Player save) : void funciona correctamente, es decir, carga correctamente la partida guardada.

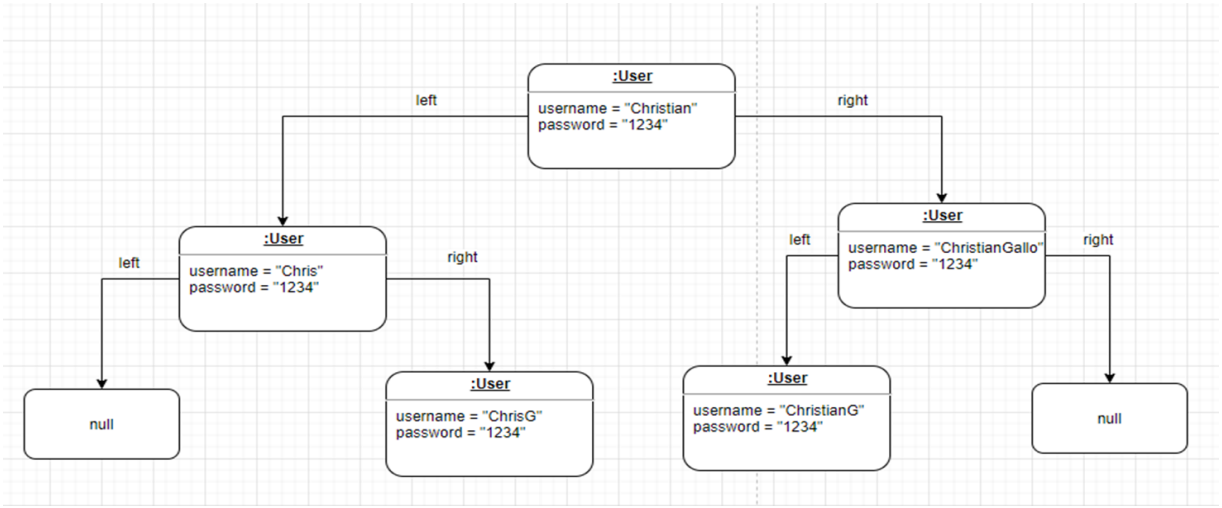
Clase	Método	Escenario	Valores de Entrada	Resultado
GameManager	loadMatch(Player save) : void	randomPlayer	Ninguno	Asigna a la asociación match el objeto Player = randomPlayer
GameManager	latch(Player save)	Ninguno	save = null	

Objetivo: Verificar que se pueden agregar objetos de la clase Log al ABB de logs cumpliento la propiedad de orden por medio del método addLog(double sesssionTime, GregorianCalendar date, User user): void				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
GameManager	addLog(double sesssionTime, GregorianCalendar date, User user): void	randomUser, actualDate	sessionTime = 10	Un objeto de la clase Log (sessionTime = 10, parent = null, left = null, right = null, date = actualDate, user = randomUser) agregado correctamente al ABB de logs,

Objetivo: Verificar que se agregan objetos de la clase user a al ABB de users cumpliento la propiedad de orden por medio del método addUser(string username, string password) : void				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
GameManager	addUser(string username, string password) : void	Ninguno	username = "Chris", password "admin"	Un objeto de la clase User (left = null, right = null, parent = null, username = "Chris", password = "admin")

Objetivo: Verificar que se agregan objetos de la clase Player a la lista enlazada de saves al final de esta por medio del método saveMatch(Player match) : void				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
GameManager	saveMatch(Player match) : void	Ninguno	username = "Chris", password "admin"	Un objeto de la clase User (left = null, right = null, parent = null, username = "Chris",

userABB


:User
username = "Christian"
password = "1234"
left
right
:User
username = "Chris"
password = "1234"
left
right
:User
username = "ChristianGallo"
password = "1234"
left
right
null
:User
username = "ChrisG"
password = "1234"
:User
username = "ChristianG"
password = "1234"
null

Configuración de los Escenarios

Nombre	Clase	Escenario

Diseño de Casos de prueba

Objetivo: Verificar que el constructor y los métodos getter de la clase User funciona de manera correcta, es decir, que se le asignen de manera correcta los atributos del objeto creado.

Clase	Método	Escenario	Valores de Entrada	Resultado
User	User(string username, string password)	Ninguno	username = "Chris", password "admin"	Un objeto de la clase User (left = null, right = null, parent = null, username = "Chris", password = "admin")

Configuración de los Escenarios

Nombre	Clase	Escenario
randomUser	User	Un objeto de la clase User (left = null, right = null, parent = null, username = "Chris", password = "admin")
actualDate	GregorianCalendar	Un objeto de la clase GregorianCalendar con la fecha actual del sistema.

Diseño de Casos de prueba

Objetivo: Verificar que el constructor y los métodos getter de la clase Log funciona de manera correcta, es decir, que se le asignen de manera correcta los atributos del objeto creado.

Clase	Método	Escenario	Valores de Entrada	Resultado
Log	Log(double sesssionTime, GregorianCalendar date, User user)	randomUser, actualDate	sessionTime = 10	Un objeto de la clase Log (sessionTime = 10, parent = null, left = null, right = null, date = actualDate, user = randomUser)

Configuración de los Escenarios

Nombre	Clase	Escenario
randomUser	User	Un objeto de la clase User (left = null, right = null, parent = null,

Diseño de Casos de prueba

Objetivo: Verificar que el constructor y los métodos getter de la clase Score funciona de manera correcta, es decir, que se le asignen de manera correcta los atributos del objeto creado.

Clase	Método	Escenario	alores de Entrad	Resultado
Score	Score(GregorianCalendar date, int score, double matchDuration)	actualDate	score = 10.2, matchDuration = 15.1	Un objeto de la clase Score (user = randomUser, score = 10.2, matchDuration = 15.1)

Configuración de los Escenarios

Nombre	Clase	Escenario
slimeSprite	AnimatedImage	un objeto de la clase AnimatedImage
slimeMovement	SlimeMovement	Un objeto de la clase SlimeMovement

Diseño de Casos de prueba

Objetivo: Verificar que el constructor de la clase Slime funciona correctamente, es decir, se agregan correctamente los atributos del objeto.				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
Slime	Slime(AnimatedImage sprite, double posX, double posY, double width, double height, Movement movement, double health, double	slimeSprite, slimeMovement	sprite=slimeSprite, posX = 500, posY = 500, width=100, height=100, movement=SlimeMovement, health=100, damage=20	Un objeto de la clase Slime con los atributos sprite=slimeSprite, posX = 500, posY = 500, width=100, height=100, movement=SlimeMovement, health=100, damage=20

Configuración de los Escenarios

Nombre	Clase	Escenario
slimeSprite	AnimatedImage	un objeto de la clase AnimatedImage
slimeMovement	SlimeMovement	Un objeto de la clase SlimeMovement

Diseño de Casos de prueba

Objetivo: Verificar que el constructor de la clase SlimeFactory funciona correctamente

Clase	Método	Escenario	Valores de Entrada	Resultado
Slime	SlimeFactory	ningunoi	ninguno	Un objeto de la clase SlimeFactory

Objetivo: Verificar que el método createMob de la clase Slime funciona correctamente, es decir, que crea Slimes

Clase	Método	Escenario	Valores de Entrada	Resultado
Slime	createMob(): Mob	slimeSprite, slimeMovement	sprite=slimeSprite, posX = 500, posY = 500, width=100, height=100, movement=SlimeMovement, health=100, damage=20	Un objeto de la clase Slime con los atributos sprite=slimeSprite, posX = 500, posY = 500, width=100, height=100, movement=SlimeMovement, health=100, damage=20

Configuración de los Escenarios

Nombre	Clase	Escenario
fireballAttackObject	fireballAttack	un Objeto de la clase FireballAttack

Diseño de Casos de prueba

Objetivo: Verificar que el constructor de la clase FireballAttack y sus getters funciona correctamente

Clase	Método	Escenario	Valores de Entrada	Resultado
Fireball	Fireball()	ninguno	ninguno	Un objeto de la clase Fireball

Objetivo: Verificar que el método attack(Entity entity) : void funciona correctamente, es decir, modifica correctamente el atributo de Health the la entidad.

Clase	Método	Escenario	Valores de Entrada	Resultado

Configuración de los Escenarios

Nombre	Clase	Escenario
fireballAttackObject	fireballAttack	un Objeto de la clase FireballAttack

Diseño de Casos de prueba

Objetivo: Verificar que el constructor de la clase FireballAttack y sus getters funciona correctamente				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
FireballAttack	FireballAttack()	ninguno	ninguno	Un objeto de la clase FireballAttacj=k

Objetivo: Verificar que el método attack(Entity entity) : void funciona correctamente, es decir, modifica correctamente el atributo de Health the la entidad.				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
FireballAttack	attack(Entity entity) : void	FireballAttackObject	ninguno	Modifica el atributo health de la entidad que se le pasa, si es que

Configuración de los Escenarios

Nombre	Clase	Escenario
slimeAttackObject	slimeAttack	un Objeto de la clase slimeAttack

Diseño de Casos de prueba

Objetivo: Verificar que el constructor de la clase SlimeAttack y sus getters funciona correctamente				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
SlimeAttack	SlimeAttack()	ninguno	ninguno	Un objeto de la clase SlimeAttack

Objetivo: Verificar que el método attack(Entity entity) : void funciona correctamente, es decir, modifica correctamente el atributo de Health the la entidad.				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
SlimeAttack	attack(Entity entity) : void	slimeAttackObject	ninguno	Modifica el atributo health de la entidad que se le pasa, si es que

Configuración de los Escenarios

Nombre	Clase	Escenario
activatePerkAttackObject	ActivatePerkAttack	un Objeto de la clase ActivatePerkAttack

Diseño de Casos de prueba

Objetivo: Verificar que el constructor de la clase ActivatePerkAttack y sus getters funciona correctamente

Clase	Método	Escenario	Valores de Entrada	Resultado
ActivatePerkAttack	ActivatePerkAttack()	ninguno	ninguno	Un objeto de la clase ActivatePerkAttack

Objetivo: Verificar que el método attack(Entity entity) : void funciona correctamente, es decir, modifica correctamente el atributo de Health the la entidad.

Clase	Método	Escenario	Valores de Entrada	Resultado
ActivatePerkAttack	attack(Entity entity) : void	activatePerkAttackObject	ninguno	Modifica el atributo health de la entidad que se le pasa, si es que

Configuración de los Escenarios

Nombre	Clase	Escenario
healthSprite	AnimatedImage	Un objeto de la clase AnimatedImage con los sprites de Health
noMovement	NoMovement	un objeto de la clase NoMovement
activatePerkAttack	ActivatePerkAttac	un objeto de la clase ActivatePerkAttack

Diseño de Casos de prueba

Objetivo: Verificar que el constructor de la clase Health y sus getters funciona correctamente

Clase	Método	Escenario	Valores de Entrada	Resultado
Health	Health()	healthSprite noMovement activatePerkAttack	sprite=healthSprite, posX = 500, posY = 500, width=100, height=100, movement=noMovement,	Un objeto de la clase Health con los atributos sprite=healthSprite, posX = 500, posY = 500, width=100, height=100, movement=noMovement, attack=activatePerkAttack

Configuración de los Escenarios

Nombre	Clase	Escenario
armorSprite	AnimatedImage	Un objeto de la clase AnimatedImage con los sprites de Armor
noMovement	NoMovement	un objeto de la clase NoMovement
activatePerkAttack	ActivatePerkAttack	un objeto de la clase ActivatePerkAttack

Diseño de Casos de prueba

Objetivo: Verificar que el constructor de la clase Armor y sus getters funciona correctamente

Clase	Método	Escenario	Valores de Entrada	Resultado
Armor	Armor()	armorSprite noMovement activatePerkAttack	sprite=armorSprite, posX = 500, posY = 500, width=100, height=100, movement=noMovement, attack=activatePerkAttack	Un objeto de la clase Health con los atributos sprite=healthSprite, posX = 500, posY = 500, width=100, height=100, movement=noMovement, attack=activatePerkAttack