Descripción breve

Documento de explicación de las pruebas realizadas al software desarrollado.

PRUEBAS

Videojuego de Combate

José Víctor García Llorente

Alejandro Martín Carrera

Jorge Padilla Rodríguez

Eloy de Sande de las Heras

Olga Somalo Serrano

Contenido

[1. Introducción 5](#_Toc139215436)

[2. App 6](#_Toc139215437)

[2.1. Método “loadRanking” 6](#_Toc139215438)

[2.2. Método “saveRanking” 6](#_Toc139215439)

[2.3. Método “loadEquipos” 7](#_Toc139215440)

[2.4. Método “saveEquipos” 7](#_Toc139215441)

[2.5. Método “loadGenerador” 7](#_Toc139215442)

[2.6. Método “saveGenerador” 7](#_Toc139215443)

[2.7. Método “loadHabilidades” 8](#_Toc139215444)

[2.8. Método “saveHabilidades” 8](#_Toc139215445)

[2.9. Método “run” 8](#_Toc139215446)

[2.9.1. Pruebas no exitosas 9](#_Toc139215447)

[3. Personaje 10](#_Toc139215448)

[3.1. Método AnadirItem 10](#_Toc139215449)

[3.2. Método ModificarOro 10](#_Toc139215450)

[3.3. Método PonerArmaActiva 11](#_Toc139215451)

[3.4. Método OroSuficiente 11](#_Toc139215452)

[3.5. Método AñadirModificador 11](#_Toc139215453)

[3.6. Método AñadirEsbirro 11](#_Toc139215454)

[3.7. Método CalcularSaludEsbirros 12](#_Toc139215455)

[3.8. Método CalcularAtaqueEquipo 12](#_Toc139215456)

[3.9. Método CalcularModificadoresAtaque 12](#_Toc139215457)

[3.10. Método CalcularModificadoresDefensa 12](#_Toc139215458)

[4. Vampiro 13](#_Toc139215459)

[4.1. Método getPotencialAtaque() 13](#_Toc139215460)

[4.2. Método getPotencialDefensa() 13](#_Toc139215461)

[4.3. Método prepararCombate() 13](#_Toc139215462)

[5. Cazador 14](#_Toc139215463)

[5.1. Método getPotencialAtaque() 14](#_Toc139215464)

[5.2. Método getPotencialDefensa() 14](#_Toc139215465)

[5.3. Método prepararCombate() 14](#_Toc139215466)

[6. Licántropo 15](#_Toc139215467)

[6.1. Método getPotencialAtaque() 15](#_Toc139215468)

[6.2. Método getPotencialDefensa() 15](#_Toc139215469)

[6.3. Método prepararCombate() 15](#_Toc139215470)

[7. Cliente 16](#_Toc139215471)

[7.1. Método cambiarPersonaje() 16](#_Toc139215472)

[7.2. Método elegirEquipo() 16](#_Toc139215473)

[7.3. Método desafiar() 17](#_Toc139215474)

[7.4. Método responderDesafios() 17](#_Toc139215475)

[7.5. Método hayDesafios() 17](#_Toc139215476)

[7.6. Método añadirCombate() 17](#_Toc139215477)

[7.7. Método consultaCombates() 18](#_Toc139215478)

[7.8. Método consultaRanking() 18](#_Toc139215479)

[7.9. Método enviarDesafio() 18](#_Toc139215480)

[7.10. Método comprarItem() 18](#_Toc139215481)

[8. Combate 19](#_Toc139215482)

[8.1. Método nuevaRonda() 19](#_Toc139215483)

[8.2. Método desarrolloCombate() 19](#_Toc139215484)

[9. Desafio 20](#_Toc139215485)

[9.1. Método validar() 20](#_Toc139215486)

[9.2. Método aceptar() 20](#_Toc139215487)

[9.3. Método rechazar() 20](#_Toc139215488)

[10. Equipos 21](#_Toc139215489)

[10.1. Método añadirEquipo\_0args() 21](#_Toc139215490)

[10.2. Método anadirEquipo\_Equipo 21](#_Toc139215491)

[10.3. Método modificarEquipo() 21](#_Toc139215492)

[10.4. Método eliminarEquipo() 22](#_Toc139215493)

[11. GeneradorIDs 22](#_Toc139215494)

[11.1. Método getID() 22](#_Toc139215495)

[12. Habilidades 22](#_Toc139215496)

[12.1. Método imprimirInfo() 22](#_Toc139215497)

[12.2. Método mostrarDisciplinas() 23](#_Toc139215498)

[12.3. Método mostrarDones() 23](#_Toc139215499)

[12.4. Método mostrarTalentos() 23](#_Toc139215500)

[12.5. Método añadirHabilidad() 23](#_Toc139215501)

[13. Menu 24](#_Toc139215502)

[13.1. Método inicio() 24](#_Toc139215503)

[13.2. Método pedirAccionCliente() 24](#_Toc139215504)

[13.3. Método pedirAccionOperador() 25](#_Toc139215505)

[13.4. Método pedirString 25](#_Toc139215506)

[13.5. Método 25](#_Toc139215507)

[14. OperadorSistema 25](#_Toc139215508)

[14.1. Método editarPersonaje() 25](#_Toc139215509)

[14.2. Método completarPersonaje() 25](#_Toc139215510)

[14.3. Método validarDesafios() 26](#_Toc139215511)

[14.4. Método banear() 26](#_Toc139215512)

[14.5. Método desbanear() 26](#_Toc139215513)

[14.6. Método añadirItemTienda() 26](#_Toc139215514)

[14.7. Método añadirHabilidad() 27](#_Toc139215515)

[15. Ranking 27](#_Toc139215516)

[16. Ronda 27](#_Toc139215517)

[16.1. Método calculoRonda() 27](#_Toc139215518)

[16.2. Método calculoNumExitosAtaque() 27](#_Toc139215519)

[16.3. Método calculoNumExitosDefensa() 27](#_Toc139215520)

[17. Tienda 28](#_Toc139215521)

[17.1. Método comprar() 28](#_Toc139215522)

[18. Usuario 28](#_Toc139215523)

[18.1. Método darseBaja() 28](#_Toc139215524)

Tabla de ilustraciones

Ilustración 1 Diagrama de la clase App 5

Ilustración 2 Diagrama de la clase Personaje 9

Ilustración 3 Diagrama de la clase Vampiro 12

Ilustración 4 Diagrama de la clase Cazador 13

Ilustración 5 Diagrama de la clase Licantropo 14

Ilustración 6 Diagrama de la clase Cliente 15

Ilustración 7 Diagrama de la clase Combate 17

Ilustración 8 Diagrama de la clase Desafio 18

Ilustración 9 Diagrama de la clase Equipos 19

Ilustración 10 Diagrama de la clase GeneradorIDs 20

# Introducción

En este documento realizaremos una revisión de los casos de prueba realizados en la práctica de la creación de un videojuego para la empresa *MetProg Urjc S.L.*

Para el análisis, lo realizaremos clase por clase, centrándonos en los Test que hemos realizado en cada clase.

De cada Test, explicaremos en qué consiste en si mismo, así como los resultados obtenidos.

# App

Tabla

Descripción generada automáticamente

Ilustración Diagrama de la clase App

Para la clase “App” hemos diseñado diferentes test, cada uno de ellos focalizado en probar/chequear el funcionamiento de cada método.

Como todos ellos necesitan la instantación de app, lo primero que haremos será crear un objeto de la clase app, mediante el comando @Before.

## Método “loadRanking”

Comprobamos que el archivo “ranking.ser” existe y se puede acceder a él.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 | - | La clase App está instanciada | No devuelve excepción (el ranking existe o se ha creado correctamente) | No devuelve excepción | - |

## Método “saveRanking”

Verificamos que se guarda correctamente la estructura ranking en el archivo.

Para ello creamos un ranking, lo guardamos en el fichero y lo leemos para verificar que coinciden.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 | Ranking | La clase App está instanciada y el archivo “ranking.ser” existe | NotNull | NotNull | - |

## Método “loadEquipos”

Comprobamos que el archivo “equipos.ser” existe y se puede acceder a él.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 | - | La clase App está instanciada | NotNull | NotNull | - |

## Método “saveEquipos”

Comprobamos que se puede acceder con permiso de escritura al archivo “equipos.ser”.

Para ello, cargamos en una variable los equipos almacenados en el fichero, añadimos un nuevo equipo a la lista y guardamos los cambios en el fichero.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 | Equipo | La clase App está instanciada y el archivo “equipos.ser” existe | No se produce excepción | No se produce excepción | En caso de que no se guardase correctamente, saltaría una excepción |

## Método “loadGenerador”

Comprobamos que el archivo “generador.ser” existe y se puede acceder a él.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 | - | La clase App está instanciada | NotNull | NotNull | - |

## Método “saveGenerador”

Comprobamos que se puede acceder con permiso de escritura al archivo “generador.ser”.

Para ello, cargamos en una variable los generadores almacenados en el fichero (lectura), añadimos nuevos generadores a la lista y guardamos los cambios en el archivo (escritura).

A continuación leemos de nuevo el fichero (actualizado con los nuevos generadores) y los guardamos en otra variable. Verificamos que los cambios se han realizado correctamente, comprobando la extensión de los generadores actualizos en la variable (sin guardarse en el archivo) y los cargados del fichero. También comprobamos que los generadores añadimos son los mismos.

Para concluir, eliminamos los generadores de prueba añadidos, restaurando el generador.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 | Generador | La clase App está instanciada y el archivo “generador.ser” existe | Equals | Equals | Si el generador actualizado en memoria y el generador creado fueran distintos, su longitud no coincidiría |

## Método “loadHabilidades”

Comprobamos que el archivo “habilidades.ser” existe y se puede acceder a él.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 | - | La clase App está instanciada | NotNull | NotNull | Si el archivo “habilidades.ser” no se encontrase o no se pudiese crear, devolvería Null |

## Método “saveHabilidades”

Comprobamos que se puede acceder con permiso de escritura al archivo “habilidades.ser”.

Para ello, cargamos en una variable las habilidades almacenadas en el fichero (lectura), añadimos nuevas habilidades a la lista y guardamos los cambios en el archivo (escritura).

A continuación leemos de nuevo el fichero (actualizado con las nuevas habilidades) y los guardamos en otra variable. Verificamos que los cambios se han realizado correctamente, comprobando la extensión de la lista de habilidades actualiza en la variable (sin guardarse en el archivo) y las cargadas del fichero. También comprobamos que las habilidades que añadimos son las mismas.

Para concluir, eliminamos las habilidades de prueba añadidas, restaurando la lista original.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 | Habilidades | La clase App está instanciada y el archivo “habilidades.ser” existe | True | True | - |
| 2 | Talento | La clase App está instanciada y el archivo “habilidades.ser” existe | True | True | - |
| 3 | Disciplina | La clase App está instanciada y el archivo “habilidades.ser” existe | True | True | - |
| 4 | Don | La clase App está instanciada y el archivo “habilidades.ser” existe | True | True | - |

## Método “run”

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 | -1 | Lorem ipsum | 5 | 5 | Lorem ipsum |
| 2 | 0 | Lorem ipsum | “hola” | “hola” | Lorem ipsum |
| 3 | 1 | Lorem ipsum | true | false | Lorem ipsum |
| 4 | null | Lorem ipsum | NullPointerException | NullPointerException | Lorem ipsum |

@Test

public void testRun() throws Exception {

System.out.println("run");

app.run();

// TODO review the generated test code and remove the default call to fail.

fail("The test case is a prototype.");

}

### Pruebas no exitosas

# Personaje

Diagrama

Descripción generada automáticamente

Ilustración Diagrama de la clase Personaje

## Método AnadirItem

Comprobamos que se puede añadir un Equipo a un personaje. Para ello creamos una instancia de Equipo, en este caso un Arma, y utilizando el método anadirItem se la añadimos a la instancia nueva de Personaje. Luego comprobamos que el nuevo personaje tenga ese arma guardada.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 | Equipo | Las clases Equipo y Personaje se pueden instanciar | True | True | - |

## Método ModificarOro

Para comprobar que el método ModificarOro funciona correctmamente instanciamos la clase Personaje (cabe recordar que el oro inicial es 50), le restamos una cantidad de oro a través del método ModificarOro y comprobamos que el valor modificado se ajusta a la realidad.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 | 5 | La clase Personaje se puede instanciar | True (45) | True (45) |  |

## Método PonerArmaActiva

Instanciamos la clase Arma y Personaje, y añadimos al personaje el arma nueva como arma activa. A continuación, comprobamos que el arma activa del personaje corresponde a la que acabamos de modificar.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 | Arma | Las clases Arma y Personaje se pueden instanciar | True | True | La respuesta es True si el arma activa coincide con la esperada |

## Método OroSuficiente

Instanciamos la clase Personaje, que por defecto tendrá 50 de oro. Llamamos al método OroSuficiente, pasándole difentes valores. Comprobamos que si los valores que le pasamos son menores o iguales al oro que tiene el personaje el método nos debe devolver true. En caso contrario, false.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 | 5 | La clase Personaje se puede instanciar | true | true | - |
| 2 | 50 | La clase Personaje se puede instanciar | true | true |  |
| 3 | 51 | La clase Personaje se puede instanciar | false | false | El valor de entrada es mayor al oro del personaje |

## Método AñadirModificador

Instanciamos las clases Modificador y Personaje. Añadimos al personaje el nuevo modificador con valor de ataque 2. Comprobamos que el potencial de ataque del modificador del personaje coincide con el valor de ataque dado.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 | 2 | Las clases Modificador y Personaje se pueden instanciar | True | True | La respuesta obtendia es true si el valor es 2 |

## Método AñadirEsbirro

Instanciamos las clases Esbirro y Personaje. Añadimos al personaje un nuevo esbirro. Comprobamos que se ha añadido correctamente.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 | Esbirro | Las clases Esbirro y Personaje se pueden instanciar | True | True | La respuesta obtenida es true si el esbirro se ha podido añadir correctamente |

## Método CalcularSaludEsbirros

Instanciamos la clase Personaje y le añadimos un esbirro con salud 1. Llamamos al método CalcularSaludEsbirros y comparamos el valor que nos devuelve con el valor esperado (1).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 | Esbirro | La clase Personaje y Esbirro se pueden instanciar | True | True | La respuesta es true si CalcularSaludEsbirros devuelve un 1 |

## Método CalcularAtaqueEquipo

Instanciamos las clases Personaje y arma. Activamos como arma activa la que acabamos de crear, con valor de ataque 2. Llamamos al método CalcularAtaqueEquipo y comparamos el valor que nos devuelve con el valor esperado (2).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 | - | Las clases Personaje y Arma se pueden instanciar | True | True | La respuesta es true si CalcularAtaqueEquipo devuelve un 1 |

## Método CalcularModificadoresAtaque

Instanciamos la clase Personaje y le añadimos un modificador con valor de ataque 2. Llamamos al método CalcularModificadoresAtaque y comparamos el valor que nos devuelve con el valor esperado (2).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 | - | La clase Personaje se puede instanciar | True | True | La respuesta es true si CacularModificadoresEquipo devuelve un 2 |

## Método CalcularModificadoresDefensa

Instanciamos la clase Personaje y le añadimos un modificador con valor de defensa 2. Llamamos al método CalcularModificadoresDefensa y comparamos el valor que nos devuelve con el valor esperado (2).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 | - | La clase Personaje se puede instanciar | True | True | La respuesta es true si CacularModificadoresDefensa devuelve un 2 |

# Vampiro

Diagrama

Descripción generada automáticamente

Ilustración Diagrama de la clase Vampiro

## Método getPotencialAtaque()

Comprobamos que el método nos devuelve el valor correspondiente. Para ello, creamos una disciplina y con ella un vampiro, y llamamos al método “GetPotencialAtaque”. Finalmente, comprueba que el valor obtenido coincide con el valor esperado.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 | Vampiro | Creamos una disciplina y con ella un vampiro. | 6 | 6 | - |

## Método getPotencialDefensa()

Análogamente al test de “GetPotencialAtaque”, comprobamos que el método funciona correctamente.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 | Vampiro | Creamos una disciplina y con ella un vampiro. | 6 | 6 | - |

## Método prepararCombate()

Comprobamos que el método nos devuelve los valor correspondientes al preparar el combate. Para ello, creamos una disciplina y con ella un vampiro, y llamamos al método “PrepararCombate”. Finalmente, comprueba que los valores guardados (salud, sangre y salud de los esbirros) coinciden con los valores esperados.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 | Vampiro | Creamos una disciplina y con ella un vampiro. | Salud: 5, sangre:0,  salud esbirros:0 | Salud: 5, sangre:0,  salud esbirros:0 | - |

# Cazador

Diagrama

Descripción generada automáticamente

Ilustración Diagrama de la clase Cazador

## Método getPotencialAtaque()

Comprobamos que el método nos devuelve el valor correspondiente. Para ello, creamos e inicialazamos las clases requeridas (talento, personaje, arma, armadura) con valores conocidos y llamamos al método “GetPotencialAtaque”. Comprobamos que ambos valores coinciden.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 | Cazador | Creamos un talento y con él un cazador. Además, le añadimos un arma y una armadura. | 8 | 8 | - |

## Método getPotencialDefensa()

Análogamente al test de “GetPotencialAtaque”, comprobamos que el método funciona correctamente.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 | Cazador | Creamos un talento y con él un cazador. Además, le añadimos un arma y una armadura. | 10 | 10 | - |

## Método prepararCombate()

Comprobamos que el método nos devuelve los valor correspondientes al preparar el combate. Para ello, creamos e inicialazamos las clases requeridas (talento, cazador) con valores conocidos y llamamos al método “PrepararCombate”. Comprobamos que los parámetros de entrada coinciden con los que se ejecutan.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 | Cazador | Creamos un talento y con él un cazador. | Salud: 5  Voluntad: 3  Salud esbirros: 0 | Salud: 5  Voluntad: 3  Salud esbirros: 0 | - |

# Licántropo

Diagrama

Descripción generada automáticamente

Ilustración Diagrama de la clase Licantropo

## Método getPotencialAtaque()

Comprobamos que el método nos devuelve el valor correspondiente. Para ello, creamos un don y con él un licántropo, al que le damos un arma y una amadura. Finalmente, llamamos al método “GetPotencialAtaque” y comprobamos que el valor obtenido coincide con el valor esperado.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 | Licántropo | Creamos un don y con él un licántropo. Además, le añadimos un arma y una armadura. | 4 | 4 | - |

## Método getPotencialDefensa()

Análogamente al test de “GetPotencialAtaque”, comprobamos que el método funciona correctamente.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 | Licántropo | Creamos un don y con él un licántropo. Además, le añadimos un arma y una armadura. | 4 | 4 | - |

## Método prepararCombate()

Comprobamos que el método nos devuelve los valor correspondientes al preparar el combate. Para ello, creamos u n don y con él un licántropo, y llamamos al método “PrepararCombate”. Finalmente, comprueba que los valores guardados (salud, rabia y salud de los esbirros) coinciden con los valores esperados.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 | Vampiro | Creamos una disciplina y con ella un vampiro. | Salud: 5  Rabia:0  Salud esbirros:0 | Salud: 5  Rabia: 0  Salud esbirros: 0 | - |

# Cliente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Ilustración Diagrama de la clase Cliente

## Método cambiarPersonaje()

Comprobamos que el método cambia el personaje del cliente satisfactoriamente. Para ello creamos la estructura habilidades correctamente y una instancia de Cliente sobre la que ejecutar el método. Finalmente comprobamos que el personaje ha cambiado comprobando que tiene un nombre diferente.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 | Habilidades | Creamos una estructura Habilidades y el cliente nuevo. | True | True | No automatizable pues requiere de input |

## Método elegirEquipo()

Comprobamos que el método elige un arma o armadura activa correctamente. Para ello creamos un cliente nuevo con un arma y una armadura guardada en su personaje pero sin equipo activo. Después ejecutamos el método y comprobamos si tiene un arma o armadura activa.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 |  | Cliente con personaje con un arma y armadura | True | True | No automatizable pues requiere de input |

## Método desafiar()

Comprobamos que el método envía desafios correctamente. Para ello creamos dos clientes en el ranking y ejecutamos el método sobre uno de ellos. Finalmente comprobamos que el desafiado tiene desafíos pendientes guardados.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 | Ranking, GeneradorID | Creamos dos clientes y los añadimos en el ranking | True | True | - |

## Método responderDesafios()

Comprobamos que el método responde a un desafío validado correctamente. Para ello creamos dos clientes con personajes válidos, enviamos un desafío y lo validamos. Finalmente ejecutamos el método y comprobamos que el combate ha tenido lugar viendo que el cliente tiene algún combate reciente guardado.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 | Ranking | Creamos dos clientes con personajes y enviamos un desafío validado | True | True | No automatizable pues requiere de input |

## Método hayDesafios()

Comprobamos que el método devuelve correctamente si hay desafíos pendientes de respuesta. Para ello creamos un cliente nuevo, sin desafíos pendientes, y ejecutamos el método comprobando que obtenemos falso.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 |  | Creamos un cliente nuevo sin desfíos pendientes | False | False | - |

## Método añadirCombate()

Comprobamos que el método añade combates al historial del cliente correctamente. Para ello creamos un cliente y un combate con sus dos personajes. Finalmente ejecutamos el método y comprobamos que el historial de combates no es vacío.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 | Combate | Creamos un cliente, dos personajes y un combate | True | True | - |

## Método consultaCombates()

Comprobamos que el método consulta correctamente los últimos combates. Para ello creamos un cliente y un combate que añadimos al historial del cliente. Finalmente ejecutamos el método y comprobamos que había combates que consultar.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 |  | Creamos un cliente con un combate en su historial | True | True | - |

## Método consultaRanking()

Comprobamos que la consulta del ranking se realiza correctamente. Para ello creamos un cliente y lo añadimos en el ranking. Finalmente ejecutamos comprobando que no ha surgido ninguna excepción.

## Método enviarDesafio()

Comprobamos que el método envía el desafío correctamente. Para ello creamos dos clientes con sus personajes completos, y los añadimos al ranking. Después creamos un desafío y ejecutamos el método. Finalmente comprobamos que el desafiado sí tiene desafíos pendientes guardados.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 | Desafío, Cliente | Creamos dos clientes con sus personajes válidos y un desafío | True | True | - |

## Método comprarItem()

Comprobamos que el método ejecuta la compra de un item de la tienda correctamente. Para ello creamos un cliente con su personaje y la tienda con un artículo. Finalmente ejecutamos el método y comprobamos que el oro del personaje ha disminuido, realizando así la compra.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 | Tienda | Creamos un cliente con personaje y la tienda con un artículo | True | True | - |

# Combate

Imagen que contiene Tabla

Descripción generada automáticamente

Ilustración Diagrama de la clase Combate

## Método nuevaRonda()

Comprobamos que el método crea una nueva ronda del combate correctamente. Para ello creamos un combate nuevo con sus personajes preparados y ejecutamos el método. Finalmente comprobamos si hay alguna ronda guardada en el combate.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 |  | Combate sin rondas con personajes completos | True | True | - |

## Método desarrolloCombate()

Comprobamos que el método desarrolla el combate completo correctamente. Para ello creamos dos personajes completos y el combate para ejecutar el método. Finalmente comprobamos que la salud de al menos uno de los personajes es 0.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 |  | Creamos un combate con dos personajes completos | True | True | - |

# Desafio

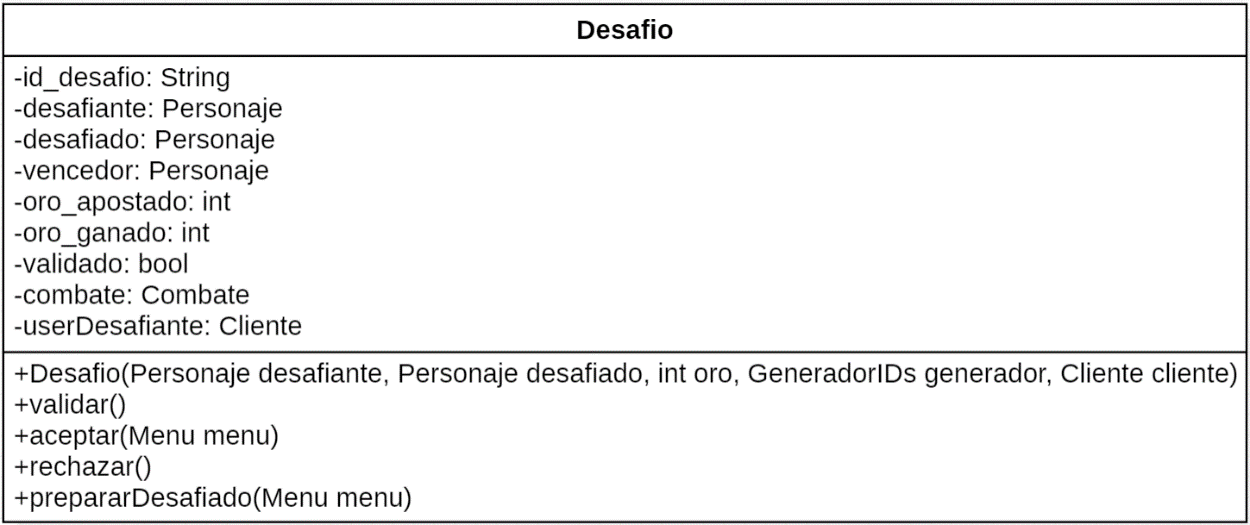


Ilustración Diagrama de la clase Desafio

## Método validar()

Comprobamos que el método valida los desafíos correctamente. Para ello creamos un desafío con dos personajes completos y ejecutamos el método. Finalmente comprobamos que el desafío tiene el parámetro validado a true.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 |  | Desafío sin validar con personajes completos | True | True | - |

## Método aceptar()

Comprobamos que el método acepta y desarrolla el desafío correctamente. Para ello creamos un desafío con dos personajes completos y ejecutamos el método. Finalmente comprobamos que la salud de al menos uno de los personajes es 0.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 | Menu | Creamos dos personajes completos y un desafío | True | True | - |

## Método rechazar()

Comprobamos que el método rechaza el desafío correctamente. Para ello creamos un desafío con dos personajes completos y ejecutamos el método. Finalmente comprobamos que el oro de desafiado se ha reducido.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 |  | Creamos dos personajes completos y un desafío | True | True | - |

# Equipos

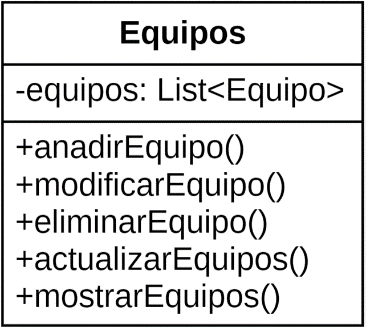


Ilustración Diagrama de la clase Equipos

## Método añadirEquipo\_0args()

Comprobamos que el método añade un item preguntado por pantalla correctamente. Para ello creamos la estructura Equipos vacía y ejecutamos el método. Finalmente comprobamos que hay equipos guardados en la estructura.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 |  | Creamos la estructura Equipos vacía | True | True | No automatizable pues requiere de input |

## Método anadirEquipo\_Equipo

Comprobamos que el método añade el item a la estructura correctamente. Para ello creamos la estructura Equipos vacía y un item. Finalmente ejecutamos el método y comprobamos que la estructura guarda al menos un item.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 | Equipo | Creamos la estructura Equipos vacía y un item | True | True | - |

## Método modificarEquipo()

Comprobamos que el método modifica un item según el input correctamente. Para ello creamos la estructura Equipos con un item añadido. Finalmente ejecutamos el método y comprobamos que el item ha cambiado al menos uno de sus parámetros.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 |  | Creamos la estructura Equipos con un item añadido | True | True | No automatizable pues requiere de input |

## Método eliminarEquipo()

Comprobamos que el método elimina los items correctamente. Para ello creamos la estructura Equipos con un item añadido y ejecutamos el método. Finalmente comprobamos que la estructura no guarda ningún item.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 |  | Creamos la estructura Equipos con un item añadido | True | True | - |

# GeneradorIDs

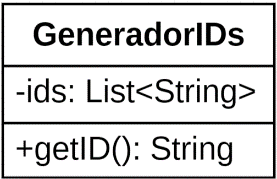


Ilustración Diagrama de la clase GeneradorIDs

## Método getID()

Comprobamos que el método devuelve un ID correctamente. Para ello creamos un generadorIDs nuevo y ejecutamos el método. Finalmente comprobamos que el ID devuelto es el primero posible.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 |  | Creamos un generadorIDs nuevo sin ningún ID guardado | A00AA | A00AA | - |

# Habilidades

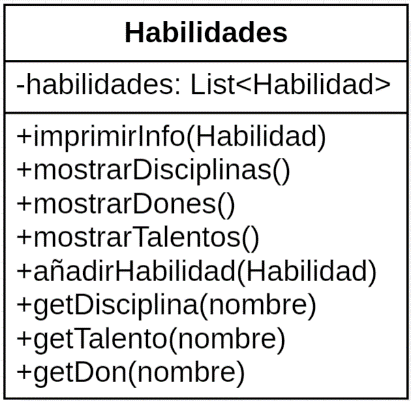


Ilustración Diagrama de la clase Habilidades

## Método imprimirInfo()

Comprobamos que el método muestra la información de una habilidad correctamente. Para ello creamos la estructura Habilidades y una Habilidad nueva a mostrar. Finalmente ejecutamos el método y comprobamos que no ha surgido ninguna excepción en el proceso y que se muestra por pantalla la información esperada.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 | Habilidad | Creamos la estructura Habilidades y una Habilidad |  |  | - |

## Método mostrarDisciplinas()

Comprobamos que el muestra todas las disciplinas guardadas en la estructura correctamente. Para ello creamos la estructura Habilidades añadiendo una Disciplina. Finalmente ejecutamos el método comprobando que no ha surgido ninguna excepción en el proceso y que se muestra por pantalla la información esperada.

## Método mostrarDones()

Análogamente a mostrarDisciplinas(), creamos la estructura con un Don y comprobamos que se muestra la información esperada sin ninguna excepción.

## Método mostrarTalentos()

Análogamente a mostrarDisciplinas() y mostrarDones(), creamos la estructura con un Talentos y comprobamos que se muestra la información esperada sin ninguna excepción.

## Método añadirHabilidad()

Comprobamos que el método añade un Habilidad a la estructura correctamente. Para ello creamos una nueva estructura Habilidades y una Habilidad. Finalmente ejecutamos el método y comprobamos que la estructura guarda por lo menos una habilidad.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 | Habilidad | Creamos la estructura Habilidades vacía y una Habilidad | True | True | - |

# Menu

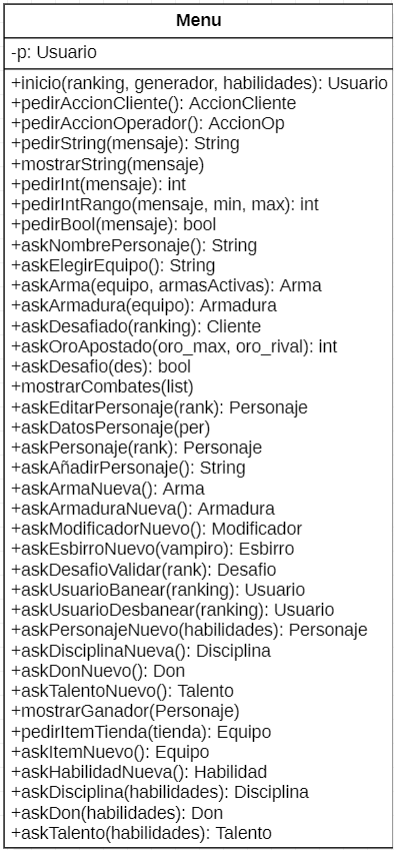


Ilustración Diagrama de la clase Menu

## Método inicio()

Comprobamos que el método realiza un inicio de sesión o registro correctamente. Para ello creamos un Ranking con un usuario y el Menú. Finalmente ejecutamos el programa y comprobamos que hemos obtenido un usuario.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 | Ranking | Ranking con un usuario | True | True | No automatizable pues requiere de input |

## Método pedirAccionCliente()

Comprobamos que el método pide correctamente una acción de cliente. Para ello ejecutamos el método comprobando que devuelve una acción.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 |  | Creamos un menú | True | True | No automatizable pues requiere de input |

## Método pedirAccionOperador()

Análogamente a pedirAccionCliente(), comprobamos que el método devuelve una acción de operador válida.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 |  | Creamos un menú |  |  | No automatizable pues requiere de input |

## Método pedirString()

Comprobamos que el método pide un String al usuario correctamente.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 |  | Creamos un menú |  |  | No automatizable pues requiere de input |

## Método mostrarString()

Método askNombrePersonaje

Método askElegirEquipo

Método askArma

Método askArmadura

Método askDesafiado

Método askOroApostado

Método askDesafio

Método MostrarCombates

Método askEditarPersonaje

Método askDatosPersonaje

Método askPersonaje

Método askAñadirPersonaje

Método askArmaNueva

Método askArmaduraNueva

Método askModificadorNuevo

Método askEsbirroNuevo

Método askDesafioValidar

Método askUsuarioBanear

Método askUsuarioDesbanear

Método askPersonajeNuevo

Método askDisciplinaNueva

Método askDonNuevo

Método askTalentoNuevo

Método PedirItemTienda

Método askItemNuevo

Método askHabilidadNueva

Método askDisciplina

Método askDon

Método askTalento

# OperadorSistema

Imagen que contiene Tabla

Descripción generada automáticamente

Ilustración Diagrama de la clase OperadorSistema

## Método editarPersonaje()

Comprobamos que el método editar el personaje que un cliente del ranking correctamente. Para ello creamos un ranking con un cliente con personaje y un OperadorSistema. Finalmente ejecutamos el método y comprobamos que el nombre del personaje ha cambiado.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 | Ranking | Creamos un Ranking con un cliente con personaje completo y un OperadorSistema | True | True | No automatizable pues requiere de input |

## Método completarPersonaje()

Comprobamos que el método completa el personaje de un cliente del ranking correctamente. Para ello creamos un ranking con un peradorSistema. Finalmente ejecutamos el método y comprobamos que el personaje tiene al menos un item nuevo.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 | Ranking, Equipos | Ranking con un cliente con personaje | True | True | No automatizable pues requiere de input |

## Método validarDesafios()

Comprobamos que el método valida desafíos pendientes de clientes del ranking correctamente. Para ello creamos un ranking con dos clientes y enviamos un desafío. Finalmente ejecutamos el método y comprobamos que el desafío ha sido validado.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 | Ranking | Ranking con dos clientes desafiados | True | True | No automatizable pues requiere de input |

## Método banear()

Comprobamos que el método banea correctamente a usuarios del ranking. Para ello creamos un ranking con un cliente y ejecutamos el método. Finalmente comprobamos que el cliente ha sido marcado como baneado.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 | Ranking | Ranking con un cliente | True | True | No automatizable pues requiere de input |

## Método desbanear()

Comprobamos que el método desbanea correctamente a usuarios del ranking. Para ello creamos un ranking con un cliente baneado y ejecutamos el método. Finalmente comprobamos que el cliente ha sido desbaneado.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 | Ranking | Ranking con un cliente desbaneado | True | True | No automatizable pues requiere de input |

## Método añadirItemTienda()

Comprobamos que el método añade un item nuevo con el input a la tienda correctamente. Para ello creamos la estructura Equipos vacía y ejecutamos el método. Finalmente comprobamos que la estructura Equipos guarda al menos un item.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 | Equipos | Estrucutra Equipos vacía | True | True | No automatizable pues requiere de input |

## Método añadirHabilidad()

Comprobamos que el método añade una habilidad nuevo según input a la estructura Habilidades correctamente. Para ello creamos la estructura Habilidades vacía y ejecutamos el método. Finalmente comprobamos que la estructura Habilidades guarda al menos un elemento.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 | Habilidades | Estructura Habilidades vacía | True | True | No automatizable pues requiere de input |

# Ranking

15.1 Método

# Ronda

## Método calculoRonda()

Comprobamos que el método ejecuta la ronda del combate correctamente. Para ello creamos dos personajes completos y creamos una Ronda. Finalmente ejecutamos el método y comprobamos que no ha surgido ninguna excepción en la ejecución.

## Método calculoNumExitosAtaque()

Comprobamos que el método realiza correctamente el cálculo de los puntos de ataque de un personaje en la ronda. Para ello creamos una ronda y un personaje completo. Finalmente ejecutamos el método y comprobamos que ha devuelto un valor positivo, que está sujeto al azar.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 | Personaje | Creamos un personaje completo y una ronda | True | True | - |

## Método calculoNumExitosDefensa()

Análogamente a calculoNumExitosAtaque(), comprobamos que se realiza el cálculo de los puntos de defensa correctamente. Para ello creamos un personaje completo y una ronda, y comprobamos que el método devuelve un valor positivo.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 | Personaje | Creamos un personaje completo y una ronda | True | True | - |

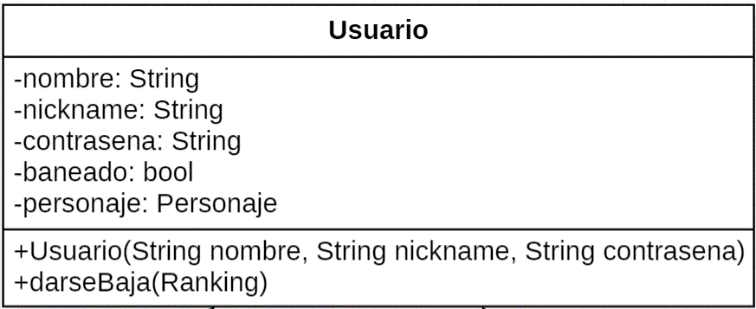
# Tienda

## Método comprar()

Comprobamos que el método permite realizar una compra en la tienda correctamente.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 | Personaje | Creamos un personaje y lo compramos. | True | True | - |

# Usuario



## Método darseBaja()

Comprobamos que el método da de baja los usuarios correctamente. Para ello, creamos un usuario, para lo cual creamos antes un ranking, un menú y un generadorIDs. Ejecutamos el método y comprobamos que el usuario se ha borrado, para ello usamos el método getUsuario de Ranking y vemos si nos devuelve null.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Entrada | Condiciones de ejecución | Respuesta Esperada | Respuesta Obtenida | Comentarios |
| 1 | Cliente | Creamos un cliente y lo damos de baja. | null | null | - |