PRÁCTICA METODOLOGÍA DE LA PROGRAMACIÓN

2022-23

CODIFICACIÓN

NOMBRE	APELLIDOS	DNI	GRADO
Lucía	Domínguez Rodrigo	49452469-P	GII+GIS
Ángel	Marqués García	34295108-S	GII+GIS
Marcos	Jiménez Pulido	04860135-M	GII+GIS
Sergio	De Oro Fernández	54712181-B	GII+GIS

LISTA DE DISTRIBUCIÓN DE ROLES

NOMBRE	ROL	ORGANIZAZIÓN
Sergio	Jefe de Proyecto	*URJC*
Lucía	Analista Funcional	*URJC*
Marcos	Analista Programador	*URJC*
Sergio	Ingeniero de Desarrollo	*URJC*
Ángel	Quality Assurance Tester	*URJC*

ÍNDICE

1. INTRODUCCIÓN

A continuación, se describen las principales decisiones tomadas para la resolución de la codificación de la práctica, así como los principales cambios con respecto al diseño inicialmente planteado. También se indicará brevemente para qué se usa cada una de las clases creadas.

2. OBJETIVOS DEL DOCUMENTO

Este documento tiene como objetivo describir los detalles del código hecho para la resolución del proyecto anteriormente diseñado y clarificar la estructura del proyecto, así como de sus diferentes paquetes, y resaltar las partes que consideramos importantes del mismo.

3. ESTRUCTURA DEL CÓDIGO

El código está estructurado en 6 paquetes de clases, cada uno con una funcionalidad determinada y homogénea. Son:

- Paquete "baseDeDatos": Contiene la base de datos de la aplicación. Al inicio de la aplicación contendrá las clases que guardan la información precargada y, según se vaya necesitando almacenar o solicitar información relativa a las diferentes clases, irá interactuando con ellas devolviendo o recogiendo la información que se le indique. Concretamente, este paquete contiene todas las clases "Almacén" que se describen en el diseño. Esta clase cobra gran importancia por la presencia de la clase "Estado", que además de recoger los distintos almacenes, guardará la información relativa al usuario que tenga una sesión activa dentro de la aplicación (es decir, aquel usuario que esté usando la aplicación en ese momento). En la clase "Estado" está recogida también la clase "FabricaPersonajes", por cómo está implementada la creación de los personajes.

  En cuanto a funcionamiento se ha decidido que, para optimizar y reducir las búsquedas dentro de la base de datos, los datos que devuelvan los almacenes y el estado sean principalmente listas y conjuntos de todos los objetos de una clase del juego. En el diseño no se implementa "Estado" como una clase serializable, ya que se pensó que cada almacén podría serializarse de forma independiente, pero esta implementación dio problemas. Además, los métodos y atributos de "Estado" dejan de ser estáticos porque Java no permite serializar propiedades estáticas. También se ha decidido que, para poder conservar la información que se use en una sesión, tanto todos los almacenes como el estado implementen la interfaz Serializable.

- Paquete "clasesDelJuego": Recoge las principales clases que interactúan en el curso normal del juego. En este paquete, con respecto al diseño, se han producido los siguientes cambios importantes a tener en cuenta:
  - Se eliminan las clases Disciplina, Don y Talento, y se añade a la clase HabilidadEspecial un atributo tipo, que tomará los valores "Disciplina", "Don" y "Talento", recogidos en un enumerado TipoHabilidad.
  - Se eliminan las clases Fortaleza y Debilidad, y se añade a la clase Modificador un atributo tipo, que tomará los valores "Fortaleza" y "Debilidad", recogidos en un enumerado TipoModificador.

- Se añaden métodos toString en varias de las clases para mejorar la presentación de los datos por el terminal. Dichos métodos serán creados a partir de sobreescribir el método toString de Java.
- Se añaden valores al enumerado EstadoDesafio, de forma que ahora admita los valores “aceptado”, “rechazado”, “completado” y “cancelado”. Estos valores se utilizan para determinar el proceso por el que pasa un desafío.
- Se añade un atributo claveEncriptacion a la clase Usuario, de forma que se encripte la contraseña que el usuario elija y, mediante la clave de encriptación, comparar las entradas de contraseña que se introduzcan para dar o no acceso al sistema con ese nick único.

- Paquete “controladores”: Contiene las diferentes clases controladoras de la aplicación, que se encargan de ejecutar las funciones principales del sistema. Además de las definidas en el diseño, se han tenido que añadir otros nuevos para poder abarcar todas las funcionalidades requeridas, como son:
  - Se añade un nuevo controlador, llamado “ControladorInicioSesion”, que gestionará el inicio de sesión de cada usuario de la aplicación en el sistema.
  - Se implementa “ControladorRegistro”, que se encargará de gestionar el resgistro de los nuevos usuarios dentro del sistema.
  - Se incorpora también “ControladorSeleccionarDesafio”, un controlador que se encargará de las funcionalidades del sistema al elegir desafíos.

  En un principio, la funcionalidad que recogen los controladores mencionados arriba se diseñó para estar recogida en otros controladores existentes, pero se ha decidido fragmentar dicha funcionalidad para mantener el principio de responsabilidad única.

- Paquete “menus”: Contiene todas las vistas del programa. Cada una de las vistas representa una pantalla y, dependiendo de las acciones tomadas por el usuario, se irán actualizando y alternando entre ellas para lograr mostrar la información más relevante en cada momento. En este paquete se incluyen, además de las clases previamente definidas, otras complementarias, como son:
  - **MenuBorrarCuenta:** Se encargará del visionado de mensajes relativos al borrado de una cuenta de usuario.
  - **MenuSeleccionarDesafio:** Permite mostrar una lista de desafíos, de la cual uno será seleccionado.

- Paquete “practicamp”: Tiene asociadas las clases que se encargan del arranque y la precarga de datos en el sistema. En concreto, el programa se inicia con el método main situado en la clase “Main”, y tras esto se llama a una clase Juego que inicializará todos los datos mediante la correspondiente llamada a la base de datos, así como las clases recuperadas de la última ejecución del software. La base de datos tendrá asociados varios archivos en formato .csv, que se leerán y se conservarán en diferentes estructuras de datos dentro del almacén esperando a que alguna clase los necesite.

- Paquete “sistemas”: Se compone de diferentes sistemas complementarios que desarrollan funcionalidades muy concretas. Estos son:
  - **DirectorCombate:** Actúa como juez del combate y, dependiendo de las estadísticas que tenga cada usuario en ese momento, determina resultados del combate, daños a cada personaje, turnos y situación de cada personaje. Se creó para que la ejecución de un combate estuviese recogida de manera independiente.
  - **ExcepcionMalaEntrada:** Es una clase que actúa de forma similar a como lo haría un manejador de excepciones. En este caso concreto, si se produjese cualquier error por cualquier motivo, la ejecución saltaría inmediatamente al código correspondiente a la excepción (similar a una rutina de tratamiento de excepciones). Esta clase se creó, en mayor o menor medida, para recoger los errores cometidos por el usuario al introducir datos por consola. Su funcionamiento se refleja en los controladores de inicio, de inicio de sesión y de registro.
  - **FabricaUsusarios**: Esta clase contiene las construcciones de los distintos objetos de tipo usuario, además de la asignación de su número de registro correspondiente.
  - **FabricaPersonajes**: esta clase contiene la creación de personajes basados en modelos. Estos modelos se inicializan junto a la clase Estado, porque su creación contiene datos pertenecientes a distintos almacenes. Estos almacenes son aquellos que leen de ficheros con formato .csv. Se tomó esta decisión para que la creación de Personajes sea lo más ligera posible (mínimo número de accesos a la base de datos).

4. MODO DE FUNCIONAMIENTO GENERAL

En esta sección se detalla de forma general el proceso de funcionamiento de las funcionalidades más básicas e importantes de la aplicación.

- **Crear una cuenta de usuario:** El proceso seguido es el siguiente:
  1. El controlador de inicio manda mostrar al menú de inicio las diferentes opciones del menú (en este caso iniciar sesión, registrarse y salir), y espera a que el usuario introduzca una entrada.
  2. Una vez conocida la entrada del usuario, el controlador de inicio validará la entrada e iniciará la función correspondiente (en este caso registrarse). Como se solicita registrarse, el controlador de inicio creará y cederá el control al controlador de registro.
  3. El controlador de registro manda al menú de registro mostrar progresivamente cada una de las opciones del formulario de registro, y tras cada mensaje mostrado, guarda cada entrada del usuario y se lo pasa a la fábrica de usuarios para la posterior creación del mismo.
  4. Una vez rellenos todos los campos necesarios, se le dice a la fábrica de usuarios que cree el usuario con los datos que se le han ido pasando (nombre, nick, contraseña y tipo de usuario). La fábrica de usuarios creará entonces el usuario (un jugador o un administrador en función del tipo de usuario especificado), le

pondrá un número de registro en el caso de que el usuario creado sea un jugador, y lo devolverá al controlador de registro.

5. Una vez generado el usuario, el controlador de registro pedirá al Estado que guarde el usuario creado. Después de guardar el usuario, el Estado guarda sus cambios en el fichero binario correspondiente al mismo.

- **Iniciar sesión en el sistema:** Se sigue:
  1. El usuario elige la opción de iniciar sesión en el sistema, por lo que el controlador de inicio creará y cederá el control al controlador de inicio de sesión.
  2. El controlador de inicio de sesión manda al menú de inicio de sesión mostrar progresivamente cada una de las opciones de inicio de sesión, y tras cada mensaje mostrado guarda cada entrada del usuario y comprueba si el nick del usuario pertenece a algún usuario de los guardados en el almacén, y si la contraseña introducida tras aplicarle la clave de encriptación asociada a ese usuario es la misma que la guardada en el usuario con ese nick.
  3. Si se cumplen las condiciones anteriores, se actualiza el usuario activo de la clase Estado, y en función de su tipo (comprobado mediante el número de registro), cederá el controlador de jugador o de administrador que tome el control de la ejecución del programa.

- **Crear un personaje:** Se siguen los siguientes pasos:
  1. El usuario con rol de jugador, estando en el menú de jugador, elige la opción de crear un nuevo personaje, por lo que el controlador de jugador creará y cederá el control al controlador de creación de personajes.
  2. El controlador de crear personajes irá solicitando los campos directamente personalizables del personaje (es decir, el nombre y el tipo de personaje) mediante una sucesión de mensajes, y recibida la información que irá guardando en sus propiedades internas.
  3. Una vez rellenados todos los campos, el controlador de creación de personajes llama a la fábrica de personajes para que cree el personaje pedido, y tras esto este personaje se añade a la lista de personajes presente en el Estado (que automáticamente se encargará de actualizar el archivo serializable), además de asignarse al usuario activo.

- **Crear un desafío:** Su funcionamiento obedece la secuencia de pasos siguiente:
  1. El jugador, estando en el menú de jugador, elige la opción de crear un nuevo desafío. Tras esto, se crea un controlador de creación de desafíos y se cede a él el control de la aplicación.
  2. Mientras que el menú de creación de desafíos va mandando mensajes de texto para guiar al usuario en la inserción de los datos en el sistema, el controlador va guardando el oro apostado y el nick del usuario al que se quiere desafiar en sus propiedades internas. Estas entradas de datos, antes de guardarse se validarán, de forma que se eviten poner cantidades de oro no posibles o crear desafíos dirigidos a usuarios inexistentes.

3. Tras la confirmación previa del usuario, el controlador de creación de desafíos resta la cantidad de oro apostada al jugador activo, crea el nuevo desafío y lo guarda en el almacén de desafíos.

- **Consultar el ranking:** Sigue el siguiente proceso:
  1. El jugador activo elige la opción de consultar el ranking de jugadores del sistema. Como el responsable del ranking de jugadores es el propio controlador de jugador, este será el que le pida al almacén de usuarios del Estado la lista de usuarios.
  2. El Estado devuelve la lista de usuarios solicitada, que ya vendrá ordenada por el propio almacén en función del número de victorias de cada jugador.
  3. El menú del ranking muestra por pantalla la lista de usuarios obtenida, dándole primeramente un formato para poder visualizar la información de mejor forma.

- **Realizar un combate:** El proceso es el siguiente:
  1. El jugador activo, tras iniciar sesión, recibe del controlador de notificaciones un aviso informándole de que ha sido desafiado por otro jugador (previa validación del administrador). Si el jugador lo acepta, el controlador de notificaciones crea un nuevo combate, le resta al jugador desafiado (es decir, al que le llega la notificación) la cantidad de oro acordada, crea el director del combate y le cede el control. Este director de combate se encargará de gestionar el desafío.
  2. El director del combate irá generando y ejecutando rondas hasta que uno de los personajes se quede sin puntos de vida. En ese momento, el director del combate comprobará las estadísticas de los personajes, de forma que:
     - Si ambos jugadores no tienen puntos de vida, entonces se produce un empate y el director de combate no designa ganadores.
     - Si solamente uno de los personajes tiene puntos de vida, entonces el director de combate determinará al personaje que sigue con puntos de vida como ganador del combate. Además, determinará el personaje que sigue con esbirros vivos.

  Después de esto, se asignarán las recompensas correspondientes a cada usuario.

- **Editar personajes:** Se sigue el siguiente proceso:
  1. El administrador, tras haber iniciado sesión, elige en el su correspondiente menú la opción de Editar personaje, la cual hace que se le pase el control a el controlador de seleccionar personaje.
  2. Dicho controlador mostrará el menú de selección de personajes (que mostrará los personajes disponibles), y, después de que el administrador proporcione una selección válida, creará un controlador de editar personajes con el personaje elegido y le cederá la ejecución al mismo.
  3. El controlador de edición de personajes mostrará el de detalles del personaje. Una vez mostradas dicha vista, se quedará a la espera de un input válido por parte del usuario, que determinará el atributo del personaje que quiere editar.

4. Si el administrador ha seleccionado un campo libre (nombre, vida, descripción y poder) se mostrará el menú campos libres, que permitirá al administrador introducir valores en el rango determinado de cada propiedad.
5. Si el administrador ha seleccionado un campo fijo (habilidad, armas disponibles, armaduras disponibles, modificadores y esbirros), se mostrará el menú de campos fijos. Esta muestra los valores seleccionables (previamente obtenidos desde la base de datos) para ese atributo, de los cuales el usuario podrá escoger la cantidad que desee.
6. Para el resto de los atributos (armas activas y armadura activas) se pasa el control al controlador de cambiar equipo activo.

- **Validar desafío:** Se sigue la siguiente secuencia:
  1. El administrador, tras haber iniciado sesión, elige en el su correspondiente menú la opción de Seleccionar desafío, la cual hace que se le pase el control a el controlador de validar desafío.
  2. El menú mostrará una lista de desafíos pendientes de validar, y después de que el administrador proporcione una entrada válida (entendiendo entrada válida como la selección de uno de los desafíos mostrados por pantalla), se preguntará al administrador si desea validar el desafío o cancelarlo.
  3. Si decide cancelarlo, se le preguntará por qué quiere cancelarlo, teniendo como opciones: “me parece injusto” e “incumple las normas”. Si selecciona que el desafío incumple las normas, además de cancelar el desafío se baneará al usuario.
  4. Si decide aceptarlo, se mostrará el menú validar desafío. Este menú mostrará las debilidades y fortalezas del personaje desafiante y el personaje desafiado, en este orden. El administrador podrá escoger los modificadores que considere correspondientes.

- **Banear usuario:** Se sigue la siguiente secuencia:
  1. El administrador, tras haber iniciado sesión, elige en el su correspondiente menú la opción de banear usuario, la cual hace que se le pase el control a el controlador de baneos.
  2. El controlador de baneos pedirá al menú de baneos que muestre los jugaores que no están baneados (dicha lista se obtiene de la base de datos), y después de esperar que el administrador proporcione una entrada válida (entendiendo válida como la selección de uno de los jugadores mostrados), muestra un mensaje de confirmación al administrador. En caso de que dicha confirmación sea positiva, el jugador quedará baneado del sistema, no dejándole iniciar sesión.

Este proceso se ejecutará de la misma manera para desbanear usuarios.

5. PERSISTENCIA

En cuanto a la forma de preservar la información que es usada por la aplicación entre diferentes ejecuciones se utilizan 2 métodos distintos, que son:

- Archivos de texto en formato .csv: Para cada grupo de elementos que deben estar precargados a la hora de iniciar la aplicación (esbirros, modificadores, habilidades especiales, armas y armaduras) se añade un archivo de datos con extensión .csv, que guardará las características principales de cada elemento inicial. A partir de esta información, dentro de la aplicación se implementa un lector de datos que leerá cada uno de los archivos y creará cada objeto, añadiéndolo a la lista de cada tipo de objeto situada dentro de su almacén correspondiente.
- Archivos binarios de clases: Durante la ejecución de la aplicación, para hacer posible la persistencia de los datos se serializa la clase Estado en un archivo binarios, de forma que cuando se inicie de nuevo la aplicación sea posible mantener los datos de los personajes, usuarios y desafíos entre ejecuciones de la aplicación.

5. DETALLES A TENER EN CUENTA

Se deberá disponer de la versión del JDK 17 en adelante para el correcto funcionamiento de la aplicación.

El IDE utilizado ha sido NetBeans, por lo que deberá ejecutarse en dicho IDE.