IES Gran Capitán
Módulo: Programación
Ciclo Formativo de Grado Superior "Desarrollo de aplicaciones Web"

Trabajo Final. Anteproyecto: Personajes

Fecha entrega: 08/05/15
Autor: **Estela Muñoz Cordón**

LISTADO DE PERSONAJES LITERARIOS PARA RELATOS O CUALQUIER PROYECTO CREATIVO:

1. INTRODUCCIÓN:

Entre mis aficiones, una de ellas es escribir. Tengo varios relatos empezados, así como comics (también me entretengo dibujando en ocasiones), y un problema con el que me encuentro normalmente es que al retomarlos, como suele pasar bastante tiempo, ya que ando normalmente ocupada con otras cosas, no me acuerdo bien de todo, sobre todo de los datos de los personajes (su nombre, raza, personalidad, reino, etc).

Como consecuencia, pierdo mucho tiempo (y no dispongo de mucho) en intentar recordar todo y organizar mis ideas.

Tengo más de un relato empezado y recordar todos los detalles de todos, es complicado y, usualmente, se me ocurren ideas que añadir de alguno de ellos de vez en cuando, no siempre del mismo. De repente se me ocurre un personaje nuevo para una de mis historias, por ejemplo.

Por todo ello, encontré interesante crear una aplicación para los escritores, donde poder guardar la información de sus personajes inventados, ya que para poder escribir es importante tenerlos muy claros y organizados.

2. ENUMERACIONES:

Las enumeraciones que trataré en mi proyecto son:

- Enumeración Region: para saber en qué zona vive un personaje.
  - Costa (para los que vivan en una bahía o en la playa).
  - Bosque.
  - Montaña.
  - Desierto.
  - Cuevas.
  - Estepa (terreno llano sin apenas árboles.
  - Valle (zona entre dos montañas).
  - Volcánica (zona cercana a un volcán).
  - Río (linde de un río).
  - Lagos.
  - Nevada (zona nevada).
  - Selva.
  - Isla.

- Enumeración Raza: algunos personajes pueden ser de un raza concreta.
  - Elfo.
  - Humano.
  - No muerto (zombi).

- Enumeración Sexo: un personaje puede ser femenino o masculino.
  - Mujer.
  - Hombre.

- Enumeración Zodiaco: un personaje tiene un signo del zodiaco, el cuál suele ser definitorio de su personalidad, ya que según el signo del zodiaco una persona tiene unas características u otras en su personalidad.
  - Aries.
  - Tauro.
  - Geminis.
  - Cáncer.
  - Leo.
  - Virgo.
  - Libra.
  - Escorpio.
  - Sagitario.
  - Capricornio.
  - Acuario.
  - Piscis.

3. CLASES Y HERENCIA:

Vamos a tener las siguientes clases con los siguientes atributos o campos:

- Personaje:
  - Nombre (de tipo String): será único y no podrá haber más de un personaje con el mismo nombre.
  - Apellido: sólo tendrá un apellido, en mi caso no tengo interés en almacenar información acerca del segundo apellido del personaje, ya que se presentan

normalmente por su nombre o primer apellido, que es por el que se identifican socialmente (de tipo String).

- Sexo (del tipo enumeración Sexo).
- Edad (de tipo int).
- Altura (de tipo float).
- Peso (de tipo float).
- Región (denominado region, del tipo enumeración Region).
- Signo del zodiaco (llamado zodiaco, del tipo enumeración Zodiaco).
- Descripción: una breve descripción de cómo es el personaje y datos interesantes sobre él (de tipo String).
- Vida (de tipo int): vitalidad del personaje.
- Danno (de tipo int): daño que puede ocasionar un personaje.
- Patrón para el nombre (patronNombre): para validar el nombre con una expresión regular (del tipo Pattern).
- Patrón para validar el apellido (patronApellido) con una expresión regular (del tipo Pattern).

◆ Arquero: para los personajes que sean de tipo arquero. Como esta clase hereda de la clase Personaje, tiene sus mismos campos, además de los siguientes:

- Raza (del tipo enumeración Raza).
- Habilidad: los arqueros son precisos en sus movimientos (del tipo int).
- Velocidad: los arqueros se caracterizan por ser muy veloces (del tipo int).
- Carcaj: un arquero tiene una cantidad de flechas para disparar con el arco, que no es ilimitada (del tipo int).

Nota: la vida del arquero cambia respecto al resto de personajes, de modo que se le pone una cantidad distinta como valor.

Métodos propios: getCarcaj(), getHabilidad(), getRaza(), getVelocidad(), recargarFechas(int), setCarcaj(), setRaza(Raza), tirarFlecha() y su constructor difiere por incluir la raza.

Método implementado de la interfaz Razable: actualizarValores().

◆ Guerrero: para los personajes que sean de tipo guerrero. Como esta clase hereda de la clase Personaje, tiene sus mismos campos, además de los siguientes:

- Raza (del tipo enumeración Raza).
- Fuerza: los guerreros se caracterizan por ser muy fuertes (del tipo int).

- Constitución: los guerreros suelen tener una constitución física robusta y vigorosa que ayuda a que sean más fuertes (del tipo int).
- Defensa: un guerrero tiene una gran resistencia física a golpes y lesiones, más que el resto de personajes menos curtidos en batallas (del tipo int).
- Ira: cuando un guerrero tiene poca vida entra en un estado de furia en el que se vuelve más violento para sobrevivir, aumentando su fuerza (del tipo boolean).

Nota: la vida del guerrero cambia respecto al resto de personajes, de modo que se le pone una cantidad distinta como valor. Habría que sopesar la posibilidad de añadir un campo denominado "filo" para controlar cuando una espada está desafilada para volver a afilarla, ya que si ha perdido su filo, hace menos daño o que,en caso de perderlo, debe volver a afilarla para seguir peleando con ella.

Métodos propios: doblePunnetazo(), embestida(), tajo(), getDefensa(), getFuerza(), getRaza(), isIra(), setFuerza() y su constructor difiere por incluir la raza.

Método implementado de la interfaz Razable: actualizarValores().

- Mago: para los personajes que sean de tipo mago. Como esta clase hereda de la clase Personaje, tiene sus mismos campos, además de los siguientes:
  - Raza (del tipo enumeración Raza).
  - Inteligencia: los magos tienen unos conocimientos propios por los que son capaces de usar su magia; mientras mayores sean mayores son sus posibilidades a la hora de lanzar hechizos más o menos poderosos (del tipo int).
  - Mana: los magos se caracterizan por tener poderes mágicos; a la cantidad de magia de la que disponen se le llama maná. Ésta se agota con cada hechizo (del tipo int).
  - Resistencia: un mago, como dominador de la magia que es, desarrolla una resistencia mágica; es decir, la magia les afecta en menor medida que a otros personajes (del tipo int).

  Nota: la vida del mago cambia respecto al resto de personajes, de modo que se le pone una cantidad distinta como valor. Sopesar la posibilidad de crear un método para recargar el maná.

  Métodos propios: bolaDeFuego(), getInteligencia(), getMana(), getRaza(), getResistencia(), lanzaDeHielo(), rayo(), setRaza() y su constructor difiere por incluir la raza.

  Método implementado de la interfaz Razable: actualizarValores().

- Dragon: los dragones, también, son una raza muy interesante y común a muchas historias, por ello quizá convendría crearla como una subclase directamente. Como esta clase hereda de la clase Personaje, tiene sus mismos campos, además de los siguientes:
  - Fuerza: los dragones al tener un gran tamaño son muy fuertes (de tipo int).
  - Defensa: un dragón es muy resistente a daños físicos (de tipo int).
  - Resistencia: un dragón es más resistente a ataques mágicos que otros personajes, ya que ellos mismos son considerados en muchos casos como seres mágicos (de tipo int).
  - Coraza: cuando un dragón activa su coraza, habilidad proveniente de sus especiales escamas, aumenta su resistencia y defensa (de tipo boolean).
  - Volar: un dragón tiene grandes alas con las que puede emprender el vuelo y surcar los cielos (de tipo boolean).

  Nota: la vida de un dragón cambia respecto al resto de personajes, de modo que se le pone una cantidad distinta como valor. Del mismo modo, su edad cambia con respecto a los demás, ya que un dragón vive más años que otros seres mortales.

  Métodos propios: aliento DeFuego(), getDefensa(), getFuerza(), getResistencia(), isCoraza(), estaVolando(), setCoraza(), setDefensa(int), setResistencia(int), setVolar(boolean), zarpazo().

- Dios: los dioses, también, son un tipo de personaje muy interesante y común a muchas historias, por ello quizá convendría crearla como una subclase directamente. Como esta clase hereda de la clase Personaje, tiene sus mismos campos, y como es un personaje de origen divino, tiene campos de otras clases hijas, conformándolo como un ser prácticamente invencible:
  - Fuerza: son más fuertes que nadie. (de tipo int).
  - Defensa: son más resistentes a daños físicos que nadie (de tipo int).
  - Inteligencia: tienen conocimientos para el uso de poderes mágicos inmensos (de tipo int).
  - Mana: tienen unos poderes mágicos grandiosos (de tipo int).
  - Resistencia: como son seres mágicos, son muy resistentes a los daños originados por la magia (de tipo int).
  - Habilidad: son muy diestros y precisos en todos sus movimientos (de tipo int).
  - Velocidad: son extremadamente rápidos (de tipo int).

  Nota: la vida del dios cambia respecto al resto de personajes, de modo que se le pone una cantidad distinta como valor. Asimismo, no tienen límite de edad, ya que son inmortales. Sopesar la posibilidad de crear un método para crear personaje y otro para eliminarlo.

Métodos propios: getDefensa(), getFuerza(), getHabilidad(), getInteligencia(), getMana(), getResistencia, getVelocidad(), juicioFinal().

4. EXPRESIONES REGULARES:

El **nombre** de un personaje empezará por mayúscula, pudiendo continuar de otra u otras letras minúsculas y un guión seguido de una letra mayúscula y otras letras minúsculas (esto es opcional, no tiene porqué contener un guión seguido de una letra mayúscula). Como mínimo tendrá tres letras y se permiten tildes y la letra ñ (u Ñ). No contendrá números ni otras signos especiales aparte del guión. El patrón para la expresión regular sería:

"^[A-ZÁÉÍÓÚÑ]([a-záéíóúñ]+[-][A-ZÁÉÍÓÚÑ])?[a-záéíóúñ]{2,}$"

Por tanto, permitiríamos nombres como:

- Rock-Naut.
- Von-Set.
- Zían.
- Nora.
- Nel.

Y no permitiríamos nombre como:

- A.
- alan.
- Liun-null.
- Aal.
- Ali4.
- 4Ana.

En cuanto al **apellido** de un personaje tiene que tener como mínimo tres letras (de la “a” a la “z”, incluida la “ñ”) y la primera ha de estar en mayúsculas. Permite tildes, pero no caracteres especiales (guiones, signos de interrogación...) ni números. El patrón para la expresión regular sería:

"^[A-ZÁÉÍÓÚÑ][a-zñáéíóú]{2,}$"

5. FECHAS:

Se pondrá la fecha de creación del programa en sí en opción "Acerca de" del menú "Ayuda" y la fecha de última modificación del listado al guardarlo.

6. INTERFAZ:

Se dispondrá de la interfaz **Razable** con el método abstracto "actualizarValores()" que se implementará en las clases Mago, Arquero y Guerrero, ya que según la raza los valores de sus características cambiarán (por ejemplo, un arquero elfo tiene más habilidad y velocidad que un arquero humano).

También, se usará la interfaz **Serializable** para las clases Personaje y ListadoDePersonajes a través de la cuál podremos almacenarlos en ficheros. Asimismo, existirá la interfaz **Comparable** para poder comparar personajes por el nombre y ordenarlos de forma alfabética.

7. ARRAYLIST:

Los personajes se irán añadiendo a un ArrayList parametrizado de tipo Personaje llamado **"listado"** para su almacenamiento, al que luego podremos acceder para verlos o eliminarlos, en su caso. Éste estará en la clase "ListadoPersonajes".

Podrán mostrarse todos los personajes existentes hasta el momento actual siempre que se desee; así como ver el total. Los personajes podrán buscarse por su nombre y se eliminarán por el mismo.

Nota: Para aspirar a mayor nota, una vez esté todo funcional, cabría la posibilidad de buscar un personaje por su raza (en su caso), por la región en donde viven, por su sexo o por su signo del zodiaco.

También, podríamos contar, por tanto, los ArrayList que se crean para mostrar las búsquedas en sus respectivos métodos (menos la de eliminar).

8. FICHEROS Y FLUJO DE DATOS:

Necesitaremos el uso de **ficheros** y **flujo de datos** para guardar nuestro listado de personajes con los datos de cada uno, pudiendo recuperarlo o modificarlo.

Crearemos una clase **Fichero** para crear un fichero y darle funcionalidad, y la clase **Gestion** para gestionar los ficheros desde ella, controlando cuando se modifica o no el listado para guardar los cambios y almacenarlo en un archivo. La llamaremos desde la interfaz gráfica. Con los flujos de datos podremos abrir o guardar los ficheros que contienen nuestro listado de personajes.

9. EXCEPCIONES:

Para añadir un personaje al listado todos sus campos han de ser válidos, lo que controlaremos con excepciones:

- NombreNoValido: en caso de que el nombre no sea válido (debe coincidir con un patrón preestablecido).
- ApellidoNoValido: en caso de que los apellidos no sean válidos (ajustándose a un patrón preestablecido)
- EdadNoValidaException: en caso de que la edad no sea válida (habrá personajes inmortales y otros con una edad más limitada que otros).
- AlturaNoValidaException: en caso de que la altura no sea válida (hay personajes con alturas diferentes, dentro de unos límites).
- PesoNoValidoException: cuando el peso no es válido (hay personas con diferente peso, dentro de unos límites).
- ManaInsuficienteException: para cuando se agote el maná de los personajes que usan magia.
- CarcajVacioException: cuando el arquero se queda sin flechas.
- CarcajLlenoException: cuando el arquero ya no puede recargar más flechas porque el carcaj ya está lleno.
- PersonajeNoExisteException: cuando el personaje no existe y no puede ser eliminado, por consiguiente.
- PersonajeYaExisteException: en caso de que el personaje ya exista y no pueda añadirse al listado.

Además, usaremos las excepciones para el control de ficheros (para almacenar y recuperar nuestro listado):

- FicheroCorrupto: en caso de que el archivo esté dañado.
- IOException: en caso de fallo o interrupción en la entrada y salida de datos.
- FileNotFoundException: en caso de que no se encuentre el archivo.
- ClassNotFoundException: en caso de que no se encuentre la clase.

10. INTERFAZ GRÁFICA:

Requeriremos de la interfaz gráfica de usuario (GUI) para que éste interaccione con la aplicación y pueda gestionar el listado de personajes a su antojo.

Crearemos una ventana principal con la clase **Principal** que tendrá los menús **“Ficheros”**, **“Personaje”**, **“Mago”**, **“Arquero”**, **“Guerrero”**, **“Dragón”**, **“Dios”** y **“Ayuda”**.

Nota: cabría la posibilidad de modificar los menús según lo requiera la funcionalidad del programa (si alguno es prescindible o modificable por otro, etc).

En "Ayuda" encontraremos información básica del programa en el submenú "**Acerca de"** (nombre, versión, creador, una descripción...) y de su utilización (instrucciones) en el submenú **"Ayuda"**.

En "Personaje" podremos acceder a las opciones: **Añadir**, **Eliminar**, **Modificar**, **Mostrar** (todos los personajes), **Buscar por nombre** y **Total** (cantidad de personajes del listado). Esto nos permitirá añadir un personaje al listado, eliminarlo o mostrarlo (uno concreto o todos los existentes). Cada una de estas opciones nos llevará a otra ventana de diálogo con la que el usuario podrá interaccionar.

Nota: cabría la posibilidad de incorporar los submenús Buscar por sexo, Buscar por raza, Buscar por región y Buscar por signo del zodiaco, en caso de que, una vez estuviera todo el programa funcional, hubiera tiempo suficiente antes de la entrega.

En "Mago", "Arquero", "Guerrero", "Dragón" y "Dios" podremos acceder a las mismas funciones, pero concretando para ese tipo de personajes, con sus peculiaridades.

Al menos habrá en uno de estos personajes (subclases) los ataques disponibles para éste y la posibilidad de poder ver las estadísticas del que se seleccione (daño tal, maná tanto, ira tanto...).

En "Ficheros" abriremos un archivo guardado anteriormente (un listado de personajes, por así decirlo), crearemos uno desde cero o modificaremos el ya existente. También, nos permitirá salir de la aplicación. Todo ello con los submenús **"Nuevo"**, **"Abrir"**, **"Guardar"**, **"Guardar como"** y **"Salir"**.

A los menús y submenús se podrá acceder con mnemonics y aceleradores.

Por tanto, dispondremos de las clases AcercaDe, Annadir, Ayuda, BuscarPorNombre, Eliminar, Mostrar, Modificar y VentanaPadre (de la que heredarán todas las anteriores menos Total, Ayuda y AcercaDe).

Nota: añadir BuscarPorRaza, BuscarPorSexo, BuscarPorRegion, BuscarPorZodiaco en caso de posible ampliación si hubiera tiempo suficiente antes de la entrega final.