# Laboratorio 3: Trabajo Previo

Sobre el trabajo presentado en el segundo laboratorio, incluyendo las modificaciones, se deben agregar funcionalidades que permitan hacer más flexible el programa actual, dejándolo así listo para agregarle una interfaz gráfica más adelante.

Para lograr esta flexibilidad se deberá usar métodos virtuales, interfaces y clases abstractas en los casos en que se indiquen o consideren que es necesario o ventajoso.

# Requerimientos

El grupo debe desarrollar un tercer prototipo del juego tomando en cuenta los siguientes requerimientos:

# Archivos de Configuración

Utilizaremos archivos de configuración, que serán leídos al inicio del programa, para cargar la siguiente personalización del juego:

## Galería Artefactos

Antes de definir la estructura del archivo, y cómo crearemos el mismo, es necesario definir los cambios al programa que efectuaremos. Para esto, consideraremos los siguientes supuestos:

Con respecto a la estructura del archivo podrá suponer que lo que representaremos será la galería de artefactos que pueden existir en el mapa pero en formato **XML**. El archivo tendrá la siguiente estructura, tenga en cuenta que esto puede influir en el diseño de clases realizado previamente:

<Artefactos>

<Artefacto> <!-- primer artefacto -->

<nombre>...</nombre>

<aparicion>...</aparicion> <!--Es la probabilidad de aparición de este

artefacto en un laberinto-->

<nivel>...</nivel> <!--Es el nivel a partir del cual puede aparecer este

artefacto. Puede aparecer en niveles superiores-->

<vida>...</vida> <!--Algunos de los artefactos aumentan la vida del avatar-->

<ataque>...</ataque> <!--Algunos de los artefactos permiten mejorar del

avatar-->

<proteccion>...</proteccion> <!--Algunos de los artefactos protegen al avatar

de los ataques-->

</Artefacto>

...

<Artefacto> <!-- último artefacto -->

<nombre>...</nombre>

<aparicion>...</aparicion>

<nivel>...</nivel>

<vida>...</vida>

<ataque>...</ataque>

<proteccion>...</proteccion>

</Artefacto>

</Artefactos>

Para generar dicha estructura de manera sencilla, utilizaremos la librería XStream, la cual facilita esto en gran medida debido a que automatiza la serialización y deserialización de objetos. La documentación de la librería se encuentra en el siguiente link:

<http://x-stream.github.io/>

Utilizar la librería XStream es sencillo, simplemente debemos agregar las siguientes librerías al proyecto:

* xstream versión 1.4.9
* xpp3\_min versión 1.1.4

La primera librería es la que contiene el objeto XStream que nos permitirá y deserializar nuestros objetos.

La segunda librería es un **parser**, el cual ayuda a XStream en la serialización y deserialización a XML.

Una vez agregadas ambas dependencias, serializar un objeto es muy sencillo, y lo hacemos de la siguiente manera:

1. Creamos el objeto XStream:

XStream xs = new XStream();

1. Llamamos al método **toXML** del objeto XStream que hemos creado, pasándole como parámetro el objeto que deseamos serializar. Esto nos devuelve un String en XML:

String temp = xs.toXML(herramientas);

1. Finalmente, esta cadena de texto la escribimos a un archivo utilizando, por ejemplo, la clase **FileWriter** que es parte de la API de Java:

try {

FileWriter fw = new FileWriter("ejemplo.txt");

fw.write(temp);

fw.close();

} catch (IOException e) {

System.out.println(e.toString());

}

Leer el archive XML generado (en este caso “ejemplo.txt”) es aún más sencillo y requiere un único paso, llamar al método **fromXML** del objeto XStream creado, pasándole como parámetro un Reader hacia el archivo creado, y efectuar el **cast** del objeto devuelto al tipo de dato verdadero (esto debido a que el método mencionado devuelve un **Object**). Ejemplo:

try {

XStream xs = new XStream();

FileReader fr = new FileReader("ejemplo.txt");

herramientas = (ArrayList<Herramienta>)xs.fromXML(fr);

fr.close();

} catch (IOException e) {

System.out.println(e.toString());

}

Puede consultar el Apéndice A del presente documento, en el cual se muestra un ejemplo completo del uso del XStream. Como sugerencia, es más sencillo crear en código los objetos y serializarlos para obtener así el archivo XML, y luego simplemente efectuar la lectura de los mismos.

# Enemigos (más) inteligentes

Ahora los enemigos no se moverán aleatoriamente sino que tratarán de acercarse al Avatar de la siguiente manera:

Trazando una línea imaginaria entre el enemigo que se desea mover y el Avatar definiremos en qué cuadrante se encuentra la línea de movimiento del Enemigo (color verde). Los cuadrantes, que hemos definido a consecuencia de este requerimiento, los clasificamos de la siguiente manera:

Cuadrante 1

Cuadrante 2

Cuadrante 3

Cuadrante 4

Mostraremos el procedimiento con el cuadrante 1, debido a que el procedimiento es muy similar en los otros cuadrantes.

Empezamos desde la celda del laberinto en la que se encuentra el enemigo (color verde). Para nuestro ejemplo tenga en cuenta que el Avatar se encuentra en la celda de color rojo.

A partir de este punto evaluamos las celdas ortogonales adyacentes **posibles** a la celda actual que se encuentren dentro del cuadrante definido (en color azul).

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 0 |  |  |  |  |  |  |  |  |  |  | Cuadrante derecho respecto del Enemigo |
| 1 |  |  |  |  |  |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |  |  |  |  |  |
| 5 |  |  |  |  |  |  |  |  |  |  |  |
| 6 |  |  |  |  |  |  |  |  |  |  |  |
| 7 |  |  |  |  |  |  |  |  |  |  |  |
| 8 |  |  |  |  |  |  |  |  |  |  |  |
| 9 |  |  |  |  |  |  |  |  |  |  |  |
| 10 |  |  |  |  |  |  |  |  |  |  |  |

A continuación, verificamos a cuál (sólo será una) de estas celdas adyacentes se moverá el Enemigo. De acuerdo a nuestro algoritmo de generación de laberinto, toda celda del laberinto sólo nos permitirá movernos, como máximo, en dos direcciones posibles; para nuestro ejemplo sólo tenemos una posibilidad marcada de color azul que es la celda a donde se moverá el Enemigo. Esta evaluación se aplicará a cada Enemigo que deba moverse.

Podría darse el caso siguiente en el cual la celda donde se encuentra el Enemigo (color verde) no tiene celdas adyacentes posibles dentro del cuadrante definido. Si se diera esta situación el enemigo se moverá de modo aleatorio hacia cualquiera de sus celdas adyacentes.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 0 |  |  | Cuadrante derecho respecto del Enemigo |  |  |  |  |  |  |  |  |
| 1 |  |  |  |  |  |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |  |  |  |  |  |
| 5 |  |  |  |  |  |  |  |  |  |  |  |
| 6 |  |  |  |  |  |  |  |  |  |  |  |
| 7 |  |  |  |  |  |  |  |  |  |  |  |
| 8 |  |  |  |  |  |  |  |  |  |  |  |
| 9 |  |  |  |  |  |  |  |  |  |  |  |
| 10 |  |  |  |  |  |  |  |  |  |  |  |

# Entregables

Cada grupo deberá subir un sólo archivo comprimido (formato ZIP o RAR) con todos los archivos de su trabajo previo en la Tarea “Laboratorio 2: Trabajo Previo” ubicada en PAIDEIA, antes de las **11:55 p.m.** del día previo al laboratorio.

El nombre del archivo comprimido deberá tener el formato:

<Número del grupo de 2 dígitos> “\_”<Código del integrante 1 de 8 dígitos> “\_”<Código del integrante 2 de 8 dígitos> “\_”<Código del integrante 3 de 8 dígitos> “\_”<Código del integrante 4 de 8 dígitos> “.” <Extensión del archivo>

Como ejemplo, el nombre del archivo del grupo 01 del horario 0784 podría ser “01\_19997626\_19971222\_20037057\_20037087.zip”.

Es COMPLETA RESPONSABILIDAD DE CADA GRUPO el colocar nombres correctos a sus archivos y evitar así confusiones al momento de acceder a ellos durante el laboratorio.

# Temas a evaluar en la sesión de laboratorio

Deben seguir las técnicas de diseño descendente (modularización top-down).

En las modificaciones individuales de sus programas pedidas durante el laboratorio se evaluarán los siguientes temas vistos en clase: Estructuras, Herencia, Polimorfismo, Clases Anidadas, Clases Anónimas, Clases Abstractas, Interfaces, Programación Genérica, RTTI, librerías y Archivos, Flujos y Persistencia de Objetos.

# Apéndice A

RECUERDE QUE ESTE APÉNDICE ES UN EJEMPLO, cada grupo deberá adaptarlo a su estructura de clases y su forma de trabajo para obtener el resultado esperado.

El presente programa consta de cuatro archivos y clases, todos en el paquete lablp2xmltest.

El programa crea una tienda virtual y luego la serializa y deserializa.

El programa requiere las librerías de XStream mencionadas en la sección 1.

## Clase: Main

/\*

\* Esta es la clase principal, y crea los armas y armaduras, las agrega a

\* la TiendaVirtual, y finalmente las serializa y escribe, y deserializa y lee

\* desde el archivo “ejemplo.txt”, utilizando la librería XStream.

\*/

package lablp2xmltest;

import com.thoughtworks.xstream.XStream;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

public class Main {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

//Armas

Arma arma1 = new Arma();

arma1.setNombre("AK-47");

arma1.setPotencia(100f);

arma1.setPrecio(1500);

arma1.setRango(600f);

arma1.setTipo(1);

Arma arma2 = new Arma();

arma2.setNombre("M-61");

arma2.setPotencia(95.5f);

arma2.setPrecio(1750);

arma2.setRango(700f);

arma2.setTipo(1);

Arma arma3 = new Arma();

arma3.setNombre("HK-906");

arma3.setPotencia(30f);

arma3.setPrecio(550);

arma3.setRango(400f);

arma3.setTipo(2);

//Armaduras

Armadura armadura1 = new Armadura();

armadura1.setNombre("Kevlar + Casco");

armadura1.setPotencia(50f);

armadura1.setPrecio(1000);

Armadura armadura2 = new Armadura();

armadura2.setNombre("Kevlar + Casco");

armadura2.setPotencia(50f);

armadura2.setPrecio(1000);

TiendaVirtual tv = new TiendaVirtual();

tv.setCredito(1000);

tv.getMercancia().add(arma1);

tv.getMercancia().add(arma2);

tv.getMercancia().add(arma3);

tv.getMercancia().add(armadura1);

try {

XStream xs = new XStream();

configureXStream(xs);

//1. Escribir el archivo

FileWriter fw = new FileWriter("ejemplo.txt");

fw.write(xs.toXML(tv));

fw.close();

//2. Leer el archivo

FileReader fr = new FileReader("ejemplo.txt");

tv = (TiendaVirtual)xs.fromXML(fr);

fr.close();

} catch (IOException e) {

System.out.println(e.toString());

}

}

private static void configureXStream(XStream xs) {

xs.alias("Arma", Arma.class);

xs.alias("Armadura", Armadura.class);

xs.alias("TiendaVirtual", TiendaVirtual.class);

}

}

## Clase: TiendaVirtual

/\*

\* Esta clase representa la Tienda Virtual.

\*/

package lablp2xmltest;

import java.util.ArrayList;

public class TiendaVirtual {

private ArrayList mercancia = new ArrayList();

private long credito = 0;

public ArrayList getMercancia() {

return mercancia;

}

public void setMercancia(ArrayList mercancia) {

this.mercancia = mercancia;

}

public long getCredito() {

return credito;

}

public void setCredito(long credito) {

this.credito = credito;

}

}

## Clase: Arma

/\*

\* Esta clase representa las armas de las unidades.

\*/

package lablp2xmltest;

/\*\*

\*

\* @author PlataformaCvLac

\*/

public class Arma {

private String nombre;

private int tipo;

private Float potencia;

private Float rango;

private long precio;

public String getNombre() {

return nombre;

}

public void setNombre(String nombre) {

this.nombre = nombre;

}

public int getTipo() {

return tipo;

}

public void setTipo(int tipo) {

this.tipo = tipo;

}

public Float getPotencia() {

return potencia;

}

public void setPotencia(Float potencia) {

this.potencia = potencia;

}

public Float getRango() {

return rango;

}

public void setRango(Float rango) {

this.rango = rango;

}

public long getPrecio() {

return precio;

}

public void setPrecio(long precio) {

this.precio = precio;

}

}

## Clase: Armadura

/\*

\* Esta clase representa las armaduras de las unidades.

\*/

package lablp2xmltest;

/\*\*

\*

\* @author PlataformaCvLac

\*/

public class Armadura {

private String nombre;

private Float potencia;

private long precio;

public String getNombre() {

return nombre;

}

public void setNombre(String nombre) {

this.nombre = nombre;

}

public Float getPotencia() {

return potencia;

}

public void setPotencia(Float potencia) {

this.potencia = potencia;

}

public long getPrecio() {

return precio;

}

public void setPrecio(long precio) {

this.precio = precio;

}

}