IF-1300 INTRODUCCIÓN A LA COMPUTACIÓN E INFORMÁTICA

Proyecto programado

Fecha de entrega final: 27 de junio, 8 am, para poder realizar la entrega final se debe haber realizado: la entrega del UML, la entrega del video, presentar al menos un adelanto (una semana antes de la entrega final) y entregar una versión del proyecto sin errores de compilación.

Objetivo General: aplicar los conocimientos adquiridos en el curso IF-1300 con el fin de demostrar las habilidades alcanzadas en el área de la programación orientada a objetos.

Descripción del problema: Se debe desarrollar un juego **RPG básico** en el que el participante asume roles de personajes en un mundo ficticio. Lucha contra distintos enemigos ficticios y toma decisiones que afectan la narrativa. El juego sigue un sistema de reglas para resolver acciones y conflictos. El juego cuenta con un menú que muestra las siguientes acciones: **explorar, estado del protagonista, comprar y salir del juego**.

Descripción del juego: El juego inicia mostrando las reglas y la mecánica del juego seguido de un mensaje con el **LEMA** del juego. Posteriormente se despliega un menú con las opciones principales del juego (explorar, estado, comprar y salir del juego). La opción **explorar** muestra un submenú que le permite al jugador escoger entre tres lugares posibles (**mares, islas y volcánes**) en cada una de estos lugares existen enemigos con los que el protagonista debe luchar para ganar vitalidad y experiencia. Durante la lucha el protagonista puede elegir entre: luchar, correr o ir a la tienda (comprar). El protagonista continua explorando hasta que decida salir y terminar el juego o sea derrotado cuando se quede sin vitalidad.

Consideraciones:

1. Dentro del proyecto se debe crear un paquete con el nombre **vista**, dentro de este paquete se debe colocar todas las clases que se encargan de crear cualquier tipo de vista (desde la terminal o graficas) en el programa. Por ejemplo: las vistas que se encargan de solicitar datos o mostrar mensajes. Dentro del paquete **vista** se debe crear un paquete con el nombre **terminal** y otro paquete con el nombre **joption.**

    a) Dentro del paquete **terminal** se deben crear las clase: **Escritor** y **Lector**. En la clase **Escritor** se crean todos los métodos requeridos para mostrar los mensajes o resultados necesarios, se debe hacer uso de las instrucciones: **System.out.println o System.out.printf.** Por su parte, en la clase **Lector** se crean todos los métodos requeridos para obtener los datos que sean necesarios y se debe utilizar el paquete **Scanner.** Todas las entradas y salidas del juego deben ser realizadas unicamente mediante el Escritor o el Lector

    b) Dentro del paquete **joption** se deben crear las clases: **Escritor** y la clase **Lector**. En la clase **Escritor** se crean todos los métodos requeridos para mostrar los mensajes o resultados necesarios y se debe hacer uso del paquete **JOptionPane.** Por su parte, en la clase **Lector** se crean todos los métodos requeridos para obtener los datos que sean necesarios y se debe utilizar el paquete **JoptionPane.**

Solo las clases Lector y Escritor puedes ser utilizadas para mostrar entradas y salidas. Importante: los nombres de los métodos que tengan la misma función en ambos lectores y en ambos escritores deben tener el mismo nombre y los mismos parámetros de ser necesario.

2. Dentro del proyecto se debe crear un paquete con el nombre **controlador**, dentro del paquete se debe crear la clase con el nombre: **JuegoRPG** con el método **main**,esta clase es la encargada de iniciar el juego y construir los objetos necesarios para comenzar con el programa del juego, en el método main se debe programar t**oda la lógica del juego**.

3. Dentro del proyecto se debe crear un paquete con el nombre **modelo**, dentro del paquete se deben crear los objetos que contienen los datos del juego como por ejemplo las clases: **Protagonista, Enemigo, Tienda y la Rueda del Destino**.

DETALLES DEL JUEGO

1. **El juego inicia mostrando el lema del juego:** "*Las posibilidades son múltiples; algunas elecciones son sencillas, otras sensatas, unas temerarias... y algunas peligrosas. Eres tú quien debe tomar las decisiones. Puedes jugar muchas veces y obtener resultados diferentes. Recuerda que tú decides la aventura, que tú eres la aventura. Si tomas una decisión imprudente, vuelve al principio y empieza de nuevo. No hay opciones acertadas o erróneas, sino muchas elecciones posibles.* Elige tu propia aventura".

2. Luego se debe crear un jugador (protagonista) al azar entre 4 personajes principales del juego, cada uno con distintos niveles de fuerza y habilidad (los personajes deben ser inventados por los integrantes del proyecto).

3. Después se debe mostrar el menú principal del juego: explorar, estado, comprar y salir del juego. La **opción salir,** termina el juego, la opción estado, muestra todos los datos del protagonista (nombre, vitalidad, entre otros), la opción tienda le permite al protagonista adquirir distintos tipos de ítem (pociones) y la opción **explorar** genera un enemigo al azar de alguno de los tres lugares (**mares, islas y volcanes**) con distintos niveles de vitalidad y poder. Luego muestra un submenu con las opciones: **luchar, correr o tienda.** La opción **tienda** le permite al jugador adquirir un item, la opción correr muestra un mensaje por ejemplo: "*el protagonista se retira del combate contra _nombre del enemigo_*" y el juego regresa al menú principal. Y la opción **luchar** permite generar valores al azar entre daño recibido y daño generado (los valores se deben calcular dependiente del poder, la vitalidad u otras características que el protagonista y el enemigo posean).

4. Si el protagonista pierde toda su vitalidad pierde el combate pero puede seguir jugando. Si el enemigo pierde toda su vitalidad el protagonista gana el combate y se le aumenta la cantidad de dinero y su vitalidad. El protagonista puede seguir luchando o jugando todas las veces que desee.

DETALLES DE LOS OBJETOS MÍNIMOS NECESARIOS

1. **Objeto Protagonista:** representa a un jugador y su rol: se requiere tener entre 3 o 4 nombres de héroes (personajes), preestablecidos y que cuando se crea una instancia de un protagonista se seleccione a la azar uno de ellos y se le asigne un nivel de vitalidad al azar. También se pueden solicitar al usuario que digite un nombre para crear al protagonista. Los **atributos mínimos de este objeto son: nombre, vitalidad (100), daño de ataque (50), nombre del ataque, poción de vitalidad (0), poción de intercambio (0), dinero (100), victorias y derrotas totales.**

5. **Objeto Enemigo**: se requiere tener 6 nombres de enemigos preestablecidos (2 enemigos por cada lugar de exploración (**mares, islas y volcanes**), los enemigos de los mares son de nivel débil (vitalidad y ataque mínimos), los de las islas intermedio y los volcanes fuerte (los integrantes del grupo deben escoger sus valores). Cuando se crea una instancia de un **Enemigo** se selecciona a la azar uno de ellos y se le asigne un nivel de vitalidad y daño de ataque al azar. Los atributos mínimos son: **nombre, vitalidad máxima, daño de ataque y nombre del ataque.**

6. **Objeto Tienda:** representa una tienda en donde se pueden comprar distintos tipos de ítem como: pociones de vitalidad y pociones de intercambio. La poción de vitalidad genera un numero al azar y le aumenta esta cantidad de vitalidad al protagonista. La poción de intercambio de vitalidad (intercambia la vitalidad entre el enemigo y el protagonista). Cada vez que un protagonista toma una poción de la tienda, el inventario debe actualizarse (los integrantes del grupo deben establecer la cantidad y el precio de cada poción disponible, si se acaban las pociones de la tienda el protagonista no podrá comprar o si el protagonista no tiene dinero suficiente).

7. **Objeto Rueda del destino**: se utiliza para obtener al azar un valor entre 1 y el valor máximo que se le asigne. Debe tener un método que permite retornar un valor generado al azar entre 1 y un valor máximo que recibe como parámetro. Este objeto es el único que se puede utilizar para generar cualquier valor al azar (no se puede utilizar el método Match.random en ningún otro lugar solo en este objeto).

8. **Objeto JuegoRPG**: La clase juego es la única que tiene el método main y se encarga de crear los objetos necesarios para cumplir con la lógica y las reglas del juego.

DETALLES FINALES:

1. **Diagrama de clases UML**: se debe entregar el **diagrama UML** de todas las clases como una primera versión de la solución propuesta del problema. No se reciben trabajo ni adelantos si no se cuenta con un diagrama UML.

2. **Control de errores**: se debe controlar que los datos solicitados al usuario sean correctos (números o textos según sea el caso). No se deben utilizar **excepciones de java** en su lugar se debe mostrar un mensaje de error y mostrar nuevamente el menú.

3. Las clases y los métodos mencionados anteriormente son los **mínimos** requeridos, pero los integrantes del proyecto pueden agregar más clases o métodos según sean necesarios para la solución del problema.

NOTA: La información anterior es una guiá para solucionar el ejercicio, pero los integrantes del proyecto pueden proponer también su propia y original solución al problema. Siempre y cuando se cumplan las reglas, los contenidos y las teorías del paradigma de programación orientada a objetos. **Se espera que los integrantes del proyecto necesiten realizar al menos un 25% de investigación individual en la solución del proyecto programado.**

Evaluación
Tome en cuenta que las convenciones y buenas prácticas serán contadas dentro de los siguientes rubros.

Evaluación	**Valor**	**Obtenido**
Estándares y buenas prácticas de programación orientada a objetos	5	
*Diagrama de clases	5	
Clases Escritor y Lector	5	
Clase Protagonista	6	
Clase Enemigo	4	
Clase Tienda	6	
Clase Rueda del destino	4	
Clase JuegoRPG	5	
Solución apropiada y completa	15	
*Video defensa del programa	5	
Resultado:	**60pts**	

Nota* El UML y el video de la defensa se debe entregar de forma obligatoria para realizar la evaluación y la calificación del código fuente de lo contrario no se aceptará la entrega del proyecto.

Aspectos de evaluación:

- El proyecto deberá ser realizado en grupos de 2 o 3 personas.
- La entrega deberá tener el siguiente formato: un único archivo del proyecto completo comprimido .zip. Cada clase debe tener en las primeras lineas el nombre y el carné de todos los integrantes del proyecto. Además, deberá venir un archivo llamado LEERME.txt en el cual se debe incluir: Universidad, carrera, curso, asignación, profesor, carné, nombre de los estudiantes y fecha de entrega de la tarea. Es responsabilidad del estudiante verificar que vengan los archivos **.java.** Deberá ser entregado en el sitio de mediación virtual del curso antes de la fecha final de entrega.
- Es responsabilidad de los estudiantes mantener en todo momento una versión de la aplicación que compile sin errores. El trabajo entregado debe cumplir con esta característica, de lo contrario será calificada con nota 0.
- La nota final estará sujeta a la defensa del código fuente por parte INDIVIDUAL de cada miembro del grupo. Cada estudiante debe tener conocimiento de la totalidad del programa y en caso de que alguno de los miembros no responda adecuadamente a lo solicitado, esto podrá afectar la nota individual. **Para la defensa del proyecto se debe realizar un video tutorial donde se explique primero el diagrama UML, luego la ejecución del programa y por último el código fuente. Las explicaciones de las clases deben ser detallada utilizando un lenguaje técnico (variables de instancias, clases, constantes, objetos etc) el código fuente debe ser defendido**

por todos los integrantes del grupo. El video debe durar entre 5 y 10 minutos aproximadamente.

- No está permitido tomar código con soluciones a problemas generadas por IA o bajadas de Internet o copiadas de otro grupo o compañero, en tal caso la nota de las tareas de los involucrados será automáticamente de 0 y se tomarán las medidas disciplinarias del caso, de acuerdo con el reglamento universitario.
- Si un grupo decide separarse debe primero hablar con el profesor para dialogar sobre la situación. El profesor es quién decide si el grupo se separa e indica el proceso de separación. Si un miembro de un grupo decide separarse deberá terminar el proyecto de forma individual. No se permite entregar dos trabajos por separado para un grupo que no ha pedido al profesor separarse, en este caso no se recibe ninguna de las dos proyectos programadas realizadas individualmente, y la nota será de cero.

Convenciones de programación

El programa debe contar con al menos las siguientes convenciones de programación y serán evaluadas dentro de cada uno de los rubros de evaluación mencionados anteriormente.

- Nombres de clases iniciando en mayúscula y representativo. Por ejemplo: Estudiante.
- Nombres de variables y métodos en minúscula. Por ejemplo: saldoDeCuenta, cobrar()
- Nombres de constantes en mayúscula. Por ejemplo: CANTIDAD_MAXIMA_DE_USUARIOS.
- Nombres significativos. Por ejemplo: saldoCuentaCorriente, en lugar de la palabra dinero.
- Iniciar cada palabra significativa con mayúscula cuando se tengan nombres compuestos por varias palabras (excepto la primera), o separados por el carácter “_” en caso de constantes. Por ejemplo: saldoDeCuentaCorriente o bien CANTIDAD_MAXIMA_DE_USUARIOS.
- Identación de forma adecuada y legible: Alineamiento apropiado entre las líneas de código que se encuentran a cada nivel de anidamiento. (Algunos se refieren a esto como Tabulación). Espaciado adecuado entre llaves, operadores, paréntesis y métodos.
- Separación de declaraciones de variables en diferentes líneas cada una.
- Uso correcto de constantes (final).
- Uso correcto de los modificadores de acceso public, private, etc.
- Despliegue adecuado de hileras y ortografía en las mismas.
- Uso de modularización (objetos) en el programa.