[75.07 / 95.02]

Algoritmos y programación III

Trabajo práctico 2: AlgoRoma

Edición: “Gladiadores en fuga”

(trabajo grupal)

Estudiantes:

Nombre	Padrón	Mail

Tutor:

Nota Final:

Índice

1. Objetivo

Desarrollar una aplicación de manera grupal aplicando todos los conceptos vistos en el curso, utilizando un lenguaje de tipado estático (Java) con un diseño del modelo orientado a objetos y trabajando con las técnicas de TDD e Integración Continua.

2. Consigna general

Desarrollar la aplicación completa, incluyendo el modelo de clases, sonidos e interfaz gráfica. La aplicación deberá ser acompañada por pruebas unitarias e integrales y documentación de diseño.

3. Especificación de la aplicación a desarrollar

AlgoRoma es un juego de tablero por turnos entre 2-6 jugadores ambientado en el imperio romano. Cada jugador tomará un gladiador con el objetivo de que escapen del coliseo romano hacia la ciudad de Pompeya. El primero en llegar gana el juego, y su gladiador tendrá como premio una casa al pie del monte Vesubio. Todos los gladiadores parten del mismo punto y van recorriendo el mismo camino lineal en un mapa con obstáculos y recompensas hasta llegar al destino final. En el camino, irán adquiriendo el equipamiento necesario para poder llegar vivos a Pompeya y entrar a la casa.

INFORMACIÓN

El mapa que recorrerán los gladiadores será compartido por la cátedra en formato JSON

Inicio del Juego

Cada gladiador comienza con 20 puntos de energía, sin equipamiento, y partiendo del coliseo romano. Esa energía se modificará durante el transcurso del juego.

Desarrollo

1. Empieza un jugador al azar y sigue en sentido lineal (si hay 4 jugadores y empieza 3, sigue 4, 1,2,)
2. El primer jugador tira un dado de 6 caras y avanza tantas casillas como indica el dado.
3. En la casilla donde cae, puede o no tener un premio o un obstáculo
    a. Si es un premio, puede recibir comida (incrementa energía) ó equipamiento (ver la sección “Premios” a continuación)
    b. Si es un obstáculo, será penalizado de acuerdo al tipo de obstáculo y equipamiento que posea el gladiador en ese momento (ver sección obstáculos).
4. Todos tiran dados y avanzan hasta llegar a Pompeya.
    a. Si el jugador que llegó a Pompeya tiene todos los equipamientos es ganador.
    b. Sino, debe retroceder a la mitad del tablero
5. Si un jugador tiene una energía inferior a 0, ese turno se pasa y se le acreditan 5 puntos extra
6. Hay un máximo de 30 tiradas de dados. Si en 30 tiradas un jugador no llega pierde.

Los gladiadores: Seniority y energía

Seniority	Turnos	Plus de energía por turno
Novato	1-8	0
Semi-Senior	8-12	5
Senior	10 en adelante	10

Los gladiadores tienen un seniority que se incrementa por cantidad de turnos jugados. Al iniciar un turno, los jugadores reciben un plus de energía según la tabla.

Premios

A continuación detallamos los tipos de premios que pueden encontrar en las casillas:

Premio	Qué sucede	Impacto
Comida	El jugador encuentra una comida (usen su creatividad)	La energía se ve incrementada en 15 puntos
Equipamiento	Mejora su equipamiento	(0) Si no tiene nada: Recibe **escudo** (1) Si tiene escudo: Recibe **armadura** (2) Si tiene armadura: Recibe **escudo y espada** (3) Si tiene todo lo anterior: Recibe la **llave** (4) Si tiene la llave: **nada**

Obstáculos

A continuación detallamos los tipos de obstáculos que pueden encontrar en las casillas:

Obstáculo	Qué sucede	Impacto
Asiste a un Bacanal	El jugador tira un dado para determinar la cantidad de copas de vino a tomar (1-6)	Saca 4 puntos de energía por cada a trago tomado
Fiera salvaje hambrienta	Se desata una pelea. Se resta energía según el equipamiento que tenga el gladiador	Energía perdida según equipamiento:: Casco = -15 Casco, Armadura = -10 Casco, Armadura, Escudo y espada = -2 Llave de la casa = 0
Lesión	El gladiador se enoja con la vida, patea una piedra y debe esperar un turno sin avanzar	El turno siguiente no avanza

Jugador

La aplicación se juega entre 2 y 6 jugadores. Al iniciar el juego el sistema debe consultar al usuario su nombre con las validaciones que correspondan.
Validaciones:

- El nombre del jugador debe contener por lo menos 4 caracteres.

Comienzo y fin de la partida

La partida comienza con los jugadores en la casilla 0 (Coliseo de Roma). La partida finaliza cuando alguno de los jugadores llega a la casilla final, o bien pasan 30 rondas de turnos y no gana nadie.
La cantidad total de obstáculos, los premios, y su posición en el tablero serán oportunamente entregados por la cátedra en formato **JSON**.

Tablero

El tablero es el lugar donde se lleva a cabo el juego. Consiste en casillas secuenciales (simil carrera de mente, ludo) sobre una grilla. La disposición de casillas del tablero también vendrá dada en formato JSON.

4. Interfaz gráfica

La interacción entre el usuario y la aplicación deberá ser mediante una interfaz gráfica intuitiva. Consistirá en una aplicación de escritorio utilizando JavaFX y se pondrá mucho énfasis y se evaluará como parte de la consigna su usabilidad. (*en el anexo I se explicarán algunas buenas prácticas para armar la interfaz gráfica de usuario o GUI*)

5. Herramientas

1. JDK (Java Development Kit): Versión 1.8 o superior.
2. JavaFX
3. JUnit 5 y Mockito: Frameworks de pruebas unitarias para Java.
4. IDE (Entorno de desarrollo integrado): Su uso es opcional y cada integrante del grupo puede utilizar uno distinto o incluso el editor de texto que más le guste. Lo importante es que el repositorio de las entregas no contenga ningún archivo de ningún IDE y que la construccnos de los IDEs más populares son:ión y ejecución de la aplicación sea totalmente independiente del entorno de desarrollo.
    a. Eclipse
    b. IntelliJ
    c. Netbeans
5. Herramienta de construcción: Se deberán incluir todos los archivos XML necesarios para la compilación y construcción automatizada de la aplicación. El informe deberá contener instrucciones acerca de los comandos necesarios (preferentemente también en el archivo README.md del repositorio). Puede usarse Maven o Apache Ant con Ivy.
6. Repositorio remoto: Todas las entregas deberán ser subidas a un repositorio único en GitHub para todo el grupo en donde quedarán registrados los aportes de cada miembro. El repositorio puede ser público o privado. En caso de ser privado debe agregarse al docente corrector como colaborador del repositorio.
7. Git: Herramienta de control de versiones
8. Herramienta de integración continua: Deberá estar configurada de manera tal que cada *commit* dispare la compilación, construcción y ejecución de las pruebas unitarias automáticamente. Algunas de las más populares son:
    a. Travis-CI
    b. Jenkins
    c. Circle-CI
    d. GitHub Actions (recomendado)

Se recomienda basarse en la estructura del proyecto base armado por la cátedra.

6. Entregables

Para cada entrega se deberá subir lo siguiente al repositorio:

1. Código fuente de la aplicación completa, incluyendo también: código de la prueba, archivos de recursos.
2. Script para compilación y ejecución (Ant o Maven).
3. Informe, acorde a lo especificado en este documento (en las primeras entregas se podrá incluir solamente un enlace a Overleaf o a Google Docs en donde confeccionen el informe e incluir el archivo PDF solamente en la entrega final).

No se deberá incluir ningún archivo compilado (formato .class) ni tampoco aquellos propios de algún IDE (por ejemplo .idea). Tampoco se deberá incluir archivos de diagramas UML propios de alguna herramienta. Todos los diagramas deben ser exportados como imágenes de manera tal que sea transparente la herramienta que hayan utilizado para crearlos.

7. Formas de entrega

Habrá 5 entregas formales que tendrán una calificación de **APROBADO o NO APROBADO** en el momento de la entrega. Además, se contará con una entrega 0 preliminares.

Aquel grupo que acumule 2 no aprobados, quedará **automáticamente desaprobado** con la consiguiente pérdida de regularidad en la materia de todos los integrantes del grupo. *En cada entrega se deberá incluir el informe actualizado*.

8. Evaluación

El día de cada entrega, cada ayudante convocará a los integrantes de su grupo, solicitará el informe correspondiente e iniciará la corrección mediante una entrevista grupal. Es imprescindible la presencia de todos los integrantes del grupo el día de cada corrección.

Se evaluará el trabajo grupal y a cada integrante en forma individual. El objetivo de esto es comprender la dinámica de trabajo del equipo y los roles que ha desempeñado cada integrante del grupo. Para que el alumno apruebe el trabajo práctico debe estar aprobado en los dos aspectos: grupal e individual (se revisarán los commits de cada integrante en el repositorio).

Dentro de los ítems a chequear el ayudante evaluará aspectos formales (como ser la forma de presentación del informe), aspectos funcionales: que se resuelva el problema planteado y aspectos operativos: que el TP funcione integrado.

9. Entregables para cada fecha de entrega

Cada entrega consta de las pruebas + el código que hace pasar dichas pruebas.

Entrega 0 (Semana 11 del calendario)

- Planteo de modelo tentativo con diagramas de clases.
- Repositorio de código creado según se explica aquí.
- Servidor de integración continua configurado.
- Al menos un commit realizado por cada integrante, actualizando el README.md

Entrega 1 (Semana 12 del calendario)

Pruebas (sin interfaz gráfica)

Caso de uso 1

- Verificar que el jugador empieza con la energía y equipamiento correspondiente.

Caso de uso 2

- Verificar que el jugador salga de la casilla inicial.

Caso de uso 3

- Verificar que un jugador sin energía no pueda jugar el turno.

Caso de uso 4

- Verificar que si recibe comida incrementa energía en 10

Caso de uso 5

- Verificar que si recibe un premio por primera vez obtiene un casco

Caso de uso 6

- Verificar que si recibe un premio por tercera vez obtiene escudo y espada.

Caso de uso 7

- Verificar que si hay un combate con una fiera salvaje y tiene un casco, pierde 10 puntos de energía.

Caso de uso 8

- Verificar que si pasan 8 turnos, el seniority del gladiador pasa de novato a senior y ve su energía incrementada al próximo turno.

Caso de uso 9

- Verificar que si llega a la meta sin la llave en el equipamiento, retrocede a la mitad de las casillas.

Caso de uso 10

- Verificar que si lo ataca una fiera salvaje y posee todo el equipamiento, el daño en energía es 0

Caso de uso 11

- Verificar que si el gladiador tiene la llave y recibe otro premio, no cambia nada

Caso de uso 12

- Verificar que si pasan 30 turnos y nadie llegó a la meta se termina el juego

Entrega 2 (Semana 13 del calendario)

Pruebas (sin interfaz gráfica). Refactor de todas aquellas pruebas de la entrega 1 que hayan sido puros getters para verificar valores y no hayan verificado **comportamiento**.

Caso de uso 13

- Verificar el formato válido del JSON del mapa.

Caso de uso 14

- Verificar el formato válido del JSON de obstáculos y premios.

Caso de uso 15

- Verificar la lectura y posterior conversión a unidades del modelo de dominio del JSON de enemigos

Caso de uso 16

- Verificar la lectura y posterior conversión a unidades del modelo de dominio del JSON del mapa.

Caso de uso 17

- Verificar que el juego se crea acorde a ambos JSON.

Caso de uso 18

- Verificar el sistema de log a utilizar necesario para la entrega 3. El log puede ser una implementación propia, casera y simple del grupo o utilizar alguna librería.

Entrega 3 (Semana 14 del calendario)

Caso de uso 19

- Simular y verificar que el jugador gana una partida.

Caso de uso 20

- Simular y verificar que el jugador pierde una partida.

Interfaz gráfica inicial básica:

- Pantallas inicial de comienzo del juego donde se le pide el nombre al usuario. Validación del mismo y botón para dar inicio al juego.
- Visualización del mapa según JSON.

Finalización del modelo junto a todas las pruebas unitarias y de integración

Se tiene que poder jugar una partida completa a través de las pruebas e ir viendo en la consola todos los eventos que están ocurriendo, por ejemplo:

```
…
Gladiador Tito avanza 2, es atacado por un animal en casilla (X,Y) y pierde energía 10
Gladiador Comodus encuentra un premio y recib casco en casilla (x,y)
Gladiador Tito no tiene energía (-5) y pasa turno
Gladiador Comodus avanza 4, encuentra pizza, energía (10)

…
Jugador Tito Gana la Partida
```

Entrega 4 (Semana 15 del calendario)

Interfaz gráfica completa acorde al enunciado.

Entrega 5 - Final: (Semana 16 del calendario)

Trabajo Práctico completo funcionando, con interfaz gráfica final, **sonidos** e informe completo.

Tiempo total de desarrollo del trabajo práctico:

6 semanas

10. Informe

El informe deberá estar subdividido en las siguientes secciones:

Supuestos

Documentar todos los supuestos hechos sobre el enunciado. Asegurarse de validar con los docentes.

Diagramas de clases

Varios diagramas de clases, mostrando la relación estática entre las clases. Pueden agregar todo el texto necesario para aclarar y explicar su diseño de manera tal que el modelo logre comunicarse de manera efectiva.

Diagramas de secuencia

Varios diagramas de secuencia, mostrando la relación dinámica entre distintos objetos planteando una gran cantidad de escenarios que contemplen las secuencias más interesantes del modelo.

Diagrama de paquetes

Incluir un diagrama de paquetes UML para mostrar el acoplamiento de su trabajo.

Diagramas de estado

Incluir diagramas de estados, mostrando tanto los estados como las distintas transiciones para varias entidades del modelo.

Detalles de implementación

Deben detallar/explicar qué estrategias utilizaron para resolver todos los puntos más conflictivos del trabajo práctico. Justificar el uso de herencia vs. delegación, mencionar que principio de diseño aplicaron en qué caso y mencionar qué patrones de diseño fueron utilizados y por qué motivos.

IMPORTANTE

No describir el concepto de herencia, delegación, principio de diseño o patrón de diseño. Solo justificar su utilización.

Excepciones

Explicar las excepciones creadas, con qué fin fueron creadas y cómo y dónde se las atrapa explicando qué acciones se toman al respecto una vez capturadas.

Anexo 0: ¿Cómo empezar?

Requisitos, análisis

¿Cómo se empieza? La respuesta es lápiz y papel. ¡No código!

1. Entiendan el dominio del problema. Definir y utilizar un lenguaje común que todo el equipo entiende y comparte. Ej.: Si hablamos de “X entidad”, todos entienden que es algo ... Si los conceptos son ambiguos nunca podrán crear un modelo congruente.
2. Compartan entre el grupo, pidan opiniones y refinen la idea en papel antes de sentarse a programar. Pueden escribir en ayudante-virtual si tienen dudas

Anexo I: Buenas prácticas en la interfaz gráfica

El objetivo de esta sección es recopilar algunas recomendaciones en la creación de interfaces gráficas de usuario o GUI. La idea no es exigir un diseño refinado y prolijo, sino que pueda ser usado por el grupo de docentes de Algo3 sin impedimentos “lo mejor posible”.

Prototipo

Venimos escribiendo código, integrales y UML todo el cuatrimestre. Me están pidiendo una interfaz gráfica. ¿Cómo se empieza? La respuesta es lápiz y papel. No código!.

1. Armen un dibujo o prototipo en papel (pueden usar google docs o cualquier herramienta también) de todas las “pantallas”. No tiene que ser perfecto.
2. Compartan entre el grupo, pidan opiniones y refinen la idea en papel antes de sentarse a programar. Pueden escribir en ayudante-virtual si tienen dudas

JavaFX

Entender cómo funciona JavaFX es clave para implementar la GUI correctamente. No subestimen el tiempo que lleva implementar y modificar la UI o interfaz de usuario. Lean todo lo que ofrece y las buenas prácticas de la tecnología. Hay plugins específicos para el IDE, pero por más que usen herramientas WYSIWYG, siempre conviene entender la API para mejorar el código autogenerado que casi nunca es óptimo.

Recomendaciones visuales

Tamaño de elementos

- Se recomienda un tamaño de tipografía de al menos a 10 puntos al mayor contraste negro contra blanco para asegurar la legibilidad, o bien 12 puntos.
- Para los botones o elementos interactivos, el tamaño mínimo del área debería ser de 18x18.
- Extra: Los elementos más importantes deberían ser más grandes y estar en posiciones más accesibles (poner ejemplos)

Contraste

- Aseguren que el texto y los elementos tengan buen contraste y se puedan leer bien
- Se sugiere un contraste cercano a 3 entre textos y fondos para texto grande e imágenes.
- Herramientas para verificar contraste: https://colourcontrast.cc/ https://contrast-grid.eightshapes.com

Uso del color

- La recomendación es no utilizar más de 3 colores para la UI y los elementos
- En el enunciado se ejemplifica con una paleta accesible, pero pueden usar cualquiera para las fichas
- Accesibilidad: Si usan para las fichas rojo y verde, o verde y azul, aseguren que las personas con daltonismo puedan distinguirlas usando letras o símbolos sobre las mismas.
- Dudas eligiendo paletas? https://color.adobe.com

Tipografía

- No se recomienda usar más de 2 tipografías para toda la aplicación
- Asegurarse de que esas tipografías se exporten correctamente en en TP
- Evitar tipografías “artísticas” para texto, menú y botones, ya que dificultan la lectura

Recomendaciones de interacción

Manejo de errores

- No escalar excepciones a la GUI. Es un No absoluto. Enviar a la consola.
- Si muestran errores al usuario, el mensaje de error debe estar escrito sin jerga técnica y permitir al usuario continuar y entender lo que está pasando
- Siempre optar por validar y prevenir errores, a dejar que el usuario ejecute la acción y falle.

Confirmaciones

- Antes de cerrar o ejecutar cualquier operación terminal, una buena práctica es pedir confirmación al usuario (para el alcance del tp no sería necesario)

Visibilidad del estado y otros

- Mostrar el estado en el cual está el juego. Siempre debería estar accesible.
- Permitir al usuario terminar o cerrar en cualquier momento.