[75.07 / 95.02 / TB025]

Algoritmos y programación III
Paradigmas y programación

Trabajo práctico 2: Gwent

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

Gwent es un juego de cartas rápido por turnos de dos jugadores, representa a dos ejércitos en una batalla, los jugadores siendo sus comandantes y las cartas su ejército.

El juego estará dividido en rondas donde, turnándose, los jugadores juegan sus cartas en el tablero, siendo ganador el mejor de 3 rondas.

Al inicio del juego cada jugador recibe 10 cartas de forma aleatoria de su mazo, se les ofrece la posibilidad de descartar y recibir nuevas cartas. Teniendo en cuenta que estas serán las únicas que reciban, salvo excepciones, durante toda la duración del juego.

Para ganar una ronda el jugador debe superar en puntos a su rival cuidando que le queden cartas suficientes para ganar alguna de las siguientes rondas, dado que al finalizar la ronda las cartas utilizadas pasan a su pila de descarte.

El tablero estará dividido en 6 partes, donde cada jugador contará con una sección para colocar unidades cuerpo a cuerpo, otra para unidades a distancia y una última para unidades de asedio (3 por cada lado).

Existen varios tipos de cartas con sus modificadores correspondientes, pero se pueden agrupar en dos categorías, las unidades y las especiales.

Los modificadores tendrán efecto en distintos elementos de la partida, como unidades, cartas disponibles o puntaje final.

3.1. Elementos

- Jugadores: Cada jugador tendrá la posibilidad de elegir su mazo antes de iniciar el juego, qué cartas jugar y poder pasar.
- Tablero: Es el tablero donde se colocarán las cartas dividido en sus respectivas secciones, además se debe mostrar al jugador:
    - Su mano
    - Su pila de descarte
    - Su número de puntaje y el del rival actualmente
- Mazos: Están conformados por cartas, deben como mínimo tener 15 Unidades y 6 Especiales.
- Cartas: Existen distintos tipos de cartas:
    - Unidades: Son cartas que tienen:
        - Seccion a la que pueden ir (Cuerpo a Cuerpo, Rango o Asedio)
        - Puntos de la carta
        - Modificadores (o no)
    - Especiales: Son cartas que afectan a unidades o al estado de la partida, no poseen puntos y no pueden colocarse como las unidades. Tienen distintos tipos.
        - Tierra arrasada: Quema a las cartas más fuertes del tablero
        - Morale boost: duplica los puntos de todas las cartas de una sección para el jugador que la uso
        - Clima: afecta alguna de las secciones para los dos jugadores, por ejemplo, nieve pone a todas las cartas en cuerpo a cuerpo con puntaje 1 --- Hay otra carta que se puede jugar si algún jugador la tiene que limpia los efectos de clima
- Modificadores:
    - Legendaria: no es afectada por otros modificadores, tienen el puntaje fijo, generalmente valores altos
    - Medicos: te permiten agarrar una carta de la pila de descarte y jugarla en el momento
    - Ágiles: se pueden ubicar en dos o más secciones, por ejemplo, algún hechicero en cuerpo a cuerpo o distancia
    - Cartas unidas: si se ubican varias cartas del mismo tipo se van duplicando sus puntos. Por ejemplo: en asedio Catapulta hace 8, si ubicas otra Catapulta pasa cada una a hacer 16 en vez de 8 (y asi si vas agregando mas)
    - Espías: Los ubicas en la sección del otro jugador (mejorando su puntaje) pero agarras 2 cartas del mazo
    - Suma de valor base: Por ejemplo le sumas +1 a todas las cartas de una

sección
- Puntuación: Es el valor que suman todas las cartas de un Jugador.

3.2. Flujo del programa

- Fase inicial: Cada jugador selecciona su nombre y su mazo.
- Fase de Preparación: Se reparten aleatoriamente 10 cartas a cada jugador de su mazo, se ofrece la posibilidad de descartar hasta 2 cartas y obtener nuevas cartas del mazo, se tira una moneda para elegir quien va a comenzar y se da comienzo a la fase de juego.
- Fase de Juego: siguiendo a la fase inicial, se inicia esta fase, consta de rondas hasta el mejor de 3:
    - Cada jugador en su turno puede:
        - Coloca una carta de unidad
        - Activar una carta especial
        - Pasar, **finalizando toda su participación en la ronda.**

      Una vez hecha su acción se da inicio al turno de su rival.
    - Cuando ambos jugadores pasan, se da por ganador al jugador de mayor puntaje, todas las cartas jugadas pasan a las respectivas pilas de descarte y se da inicio a la siguiente ronda.
- Fase final: El juego termina cuando un jugador gana 2 rondas.

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 GU*I)

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 construcción 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 4 entregas formales que tendrán una calificación de APROBADO o NO APROBADO en el momento de la entrega. Además, se contará con una entrega 0 preliminar.

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.

Cada entrega presupone que los equipos realizarán refactor de su código según crean conveniente.

Entrega 0 (Semana 12 del calendario)

- Planteo de modelo tentativo
- Diagrama de clases general
- Diagrama de secuencia para el caso donde se calcula el puntaje de un jugador basado en las cartas que se encuentran en el tablero.

Entrega 1 (Semana 13 del calendario)

Pruebas (sin interfaz gráfica):

1. Verificar que un jugador posea cartas suficientes para empezar el juego en su mazo.
2. Verificar que a un jugador se le reparten 10 cartas de su mazo.
3. Verificar que un jugador pueda colocar una carta en una sección del tablero
4. Verificar que un jugador juegue una carta de su mazo y tenga un puntaje parcial.
5. Verificar que las cartas pasen a la pila de descarte
6. Verificar que al modificar una carta con una carta unida se cambien sus puntos y se aplique el valor solo a la ronda.
7. Verificar que al utilizar una carta especial de clima se reduzca el valor de las cartas de la sección correspondiente.

Entrega 2 (Semana 14 del calendario)

Pruebas (sin interfaz gráfica):

- Verificar que se pueda eliminar el afecto de clima
- Verificar que si se usa un Tierra arrasada se eliminen las cartas correctamente.
- Verificar que si una carta con modificador espía se ubique en la sección correspondiente y se permita agarrar dos cartas.
- Verificar que una carta ágil se pueda colocar en la sección correspondiente.
- Verificar que si se juega una carta médico se pueda agarrar de la pila de descarte
- Verificar la lectura y posterior conversión a unidades del modelo de dominio del JSON
- Planteo inicial de interfaz gráfica (mockups/dibujos), pantalla donde se muestra una ronda

Entrega 3 (Semana 15 del calendario)

- Modelo del juego terminado
- Interfaz gráfica inicial básica: comienzo del juego y visualización del tablero e interfaz de usuario básica.
- Modelo del manejo de turnos en el juego.

Entrega 4 (Semana 16 del calendario)

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

Tiempo total de desarrollo del trabajo práctico:

5 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.

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.

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.

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