

[75.07 / 95.02 / TB025]

Algoritmos y programación III

Paradigmas y programación

Trabajo práctico 2: Balatro

Estudiantes:

Nombre	Padrón	Mail

Tutor:

Nota Final:

Índice

1. Objetivo	3
2. Consigna general	3
3. Motivación / Background	3
4. Especificación de la aplicación a desarrollar	4
5. Interfaz gráfica	8
6. Herramientas	9
7. Entregables	10
8. Formas de entrega	10
9. Evaluación	10
10. Entregables para cada fecha de entrega	11
Entrega 0 (Semana 12 del calendario)	11
Entrega 1 (Semana 13 del calendario)	11
Entrega 2 (Semana 14 del calendario)	12
Entrega 3 (Semana 15 del calendario)	12
Entrega 4 (Semana 16 del calendario)	12
11. Informe	13
Supuestos	13
Diagramas de clases	13
Diagramas de secuencia	13
Diagrama de paquetes	13
Detalles de implementación	13
Excepciones	13
Anexo 0: ¿Cómo empezar?	14
Requisitos, análisis	14
Anexo I: Buenas prácticas en la interfaz gráfica	14
Prototipo	14
JavaFX	14
Recomendaciones visuales	15
Tamaño de elementos	15
Contraste	15
Uso del color	15
Tipografía	15
Recomendaciones de interacción	16
Manejo de errores	16
Confirmaciones	16
Visibilidad del estado y otros	16

1. Objetivo

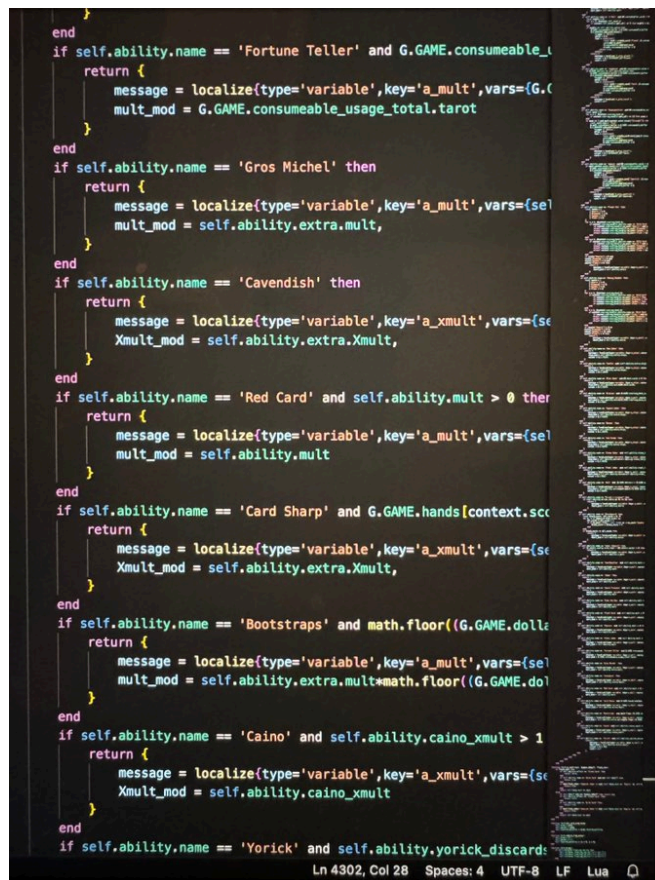
Desarrollar una aplicación de manera grupal aplicando todos los conceptos vistos en el curso, utilizando un lenguaje de tipado estático (Java) con un diseño del modelo orientado a objetos y trabajando con las técnicas de TDD e Integración Continua.

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. Motivación / Background

Balatro es un juego [“deck-builder”](#) hecho en [lua](#) que se [encuentra disponible en steam](#). Un usuario difundió el código fuente del mismo y compartió que en un archivo `card.lua` había una función de 4717 líneas de código con la implementación interna de las modificaciones que hacen todas las cartas del juego.



```
end
if self.ability.name == 'Fortune Teller' and G.GAME.consumeable_u
    return {
        message = localize(type='variable',key='a_mult',vars={G.C
        mult_mod = G.GAME.consumeable_usage_total.tarot
    }
end
if self.ability.name == 'Gros Michel' then
    return {
        message = localize(type='variable',key='a_mult',vars={sel
        mult_mod = self.ability.extra.mult,
    }
end
if self.ability.name == 'Cavendish' then
    return {
        message = localize(type='variable',key='a_xmult',vars={se
        Xmult_mod = self.ability.extra.Xmult,
    }
end
if self.ability.name == 'Red Card' and self.ability.mult > 0 then
    return {
        message = localize(type='variable',key='a_mult',vars={sel
        mult_mod = self.ability.mult
    }
end
if self.ability.name == 'Card Sharp' and G.GAME.hands[context.scc
    return {
        message = localize(type='variable',key='a_xmult',vars={se
        Xmult_mod = self.ability.extra.Xmult,
    }
end
if self.ability.name == 'Bootstraps' and math.floor((G.GAME.dolla
    return {
        message = localize(type='variable',key='a_mult',vars={sel
        mult_mod = self.ability.extra.mult+math.floor((G.GAME.dol
    }
end
if self.ability.name == 'Caino' and self.ability.caino_xmult > 1
    return {
        message = localize(type='variable',key='a_xmult',vars={se
        Xmult_mod = self.ability.caino_xmult
    }
end
if self.ability.name == 'Yorick' and self.ability.yorick_discard:
```

Figura 1. ifs de card.lua

Como ya vimos, real devs don't use ifs, nuestra propuesta es que ustedes lo mejoren.

4. Especificación de la aplicación a desarrollar

La aplicación consiste en un juego por turnos, “roguelike”, **de un solo jugador** que consiste en ir perfeccionando un mazo de cartas, sumando puntos por manos jugadas.

El juego estará dividido en hasta N rondas donde en cada ronda el jugador se le reparte una mano al azar de su mazo y el jugador debe conformar manos de poker para sumar puntos

Cada ronda posee un puntaje mínimo que el jugador debe anotar para poder seguir jugando, tiene hasta cinco turnos para alcanzar la puntuación en donde en cada turno puede o jugar una mano o descartar cartas.

Existen varios tipos de cartas, que cambian su forma de calcular el puntaje o poseen efectos especiales.

También existen cartas que están por fuera del mazo, entre ellos los comodines que tienen efectos especiales sobre la forma de calcular el puntaje y pueden combinarse unos con otros.

Además de también existir las cartas de tarot que pueden modificar a las cartas del mazo.

El objetivo del juego es crear un mazo que permita sobrevivir la mayor cantidad de rondas siendo esta la puntuación final.

4.0 Reglas

El juego de cartas a desarrollar va a seguir aproximadamente las reglas del poker con [baraja inglesa](#), brevemente dejamos una explicación de las mismas.

Con las cartas se pueden realizar las siguientes manos, con valor de mayor a menor.

- **Royal flush:** A, K, Q, J, 10 – Todas del mismo palo
- **Straight flush:** Combinación de cinco cartas consecutivas del mismo palo. Por ejemplo 9, 8, 7, 6, 5
- **Four of a kind:** Combinación de cuatro cartas del mismo valor y una carta cualquiera. Ej. A, A, A, A, 8
- **Full house:** Combinación de tres cartas del mismo valor más una pareja distinta. Ej. Q, Q, Q, 4, 4
- **Flush:** Cualquier combinación de 5 cartas del mismo palo
- **Straight:** Combinación de cinco cartas consecutivas, no necesariamente del mismo palo
- **Three of a kind:** Combinación de tres cartas del mismo valor y otras dos cartas distintas al trío y distintas entre sí.
- **Two pair:** Combinación de dos pares diferentes
- **One Pair:** Combinación de un par del mismo palo
- **High card:** Sucede cuando las cinco cartas no interactúan entre sí

El valor viene dado por la probabilidad de que ocurran estas combinaciones al distribuir las cartas:

POKER HAND RANKINGS

PROBABILITY OF MAKING A SPECIFIC HAND

Royal flush ~0.0001%



Straight flush ~0.0013%



Four-of-a-kind ~0.0240%



Full house ~0.1440%



Flush ~0.1965%



Straight ~0.3924%



Three-of-a-kind ~2.1128%



Two pair ~4.7539%



Pair ~42.2569%



High card ~50.1177%



Se recomienda leer las reglas del [poker](#) para tener más detalle. *Recordar que el juego que vamos a realizar es para un solo jugador, todo lo que es comparación de manos de un mismo tipo no aplica a nuestro juego.*

4.1. Elementos

- **Jugador:** El jugador tendrá la posibilidad de elegir qué mano jugar, que cartas comprar y en qué orden usarlas.
- **Panel:** Es el panel donde se mostrará información de la partida:
 - El número de puntaje a alcanzar y el obtenido actualmente
 - El puntaje de la mano y el multiplicador junto con el resultado
 - La cantidad de manos restantes y las jugadas
 - La cantidad de descartes restantes
- **Cartas:** Existen distintos tipos de cartas:
 - Cartas normales de poker: Son aquellas que pertenecen al mazo tradicional de poker y pueden ser añadidas al mazo del jugador. El jugador selecciona las mismas para hacer [manos de poker](#).
 - Comodines: Son cartas que no van en el mazo y que tienen efectos especiales que afectan a la puntuación o a la partida, siempre y cuando el jugador las posea. Solo puede poseer 5 comodines a menos que se especifique lo contrario. **Importa el orden** ya que se aplican de izquierda a derecha.
 - Tarot: Son cartas que no van en el mazo, pero que pueden usarse para modificar los puntos y/o el multiplicador de las cartas normales o de la mano, el jugador solo puede llevar 2 a menos que se especifique lo contrario y cuando lo desee puede usarlas para modificar una carta de su mano.

Por ejemplo: Se utiliza un tarot que modifica los puntos en 10 en la carta 3 de trébol, la carta anteriormente debía puntuar como un 3, pero posterior a la modificación ahora puntúa en 10.

Otro ejemplo: Se usa un tarot que modifica la multiplicación en un x3 sobre la mano "par". Anteriormente por jugar un "par" sea cual sea, tenía una puntuación base de 10 puntos x 2 multiplicador, ahora con la modificación sería 10 puntos x 3 multiplicador.
- **Efectos de los comodines:**
 - Al Puntaje: Ya sea afectando al multiplicador o a los puntos
 - Bonus por Mano Jugada: Se modifica los puntos y/o el multiplicador dependiendo de qué mano se jugó.

- Bonus por Descarte: Se modifica los puntos y/o el multiplicador dependiendo si se realizo un descarte.
 - Chance de activarse aleatoriamente: Se aplica el efecto de forma aleatoria.
 - Combinación: Cualquier combinación de efectos anteriores.
-
- Puntuación: Se va conformando por varias etapas, una vez finalizadas se calcula como su valor como Puntos por Multiplicador.
 - Por Carta de Izquierda a Derecha: Se suma el valor numérico de la carta salvo que haya sido modificada por un tarot, en ese caso será el valor que aplique el tarot.
 - Por Mano: Dependiendo de la mano que utilizo tiene una puntuación base por mano:
 - Carta alta: 5 puntos x 1 multiplicador
 - Par: 10 puntos x 2 multiplicador
 - Doble Par: 20 puntos x 2 multiplicador
 - Trio: 30 puntos x 3 multiplicador
 - Escalera: 30 puntos x 4 multiplicador
 - Color: 35 puntos x 4 multiplicador
 - Full House: 40 puntos x 4 multiplicador
 - Poker: 60 puntos x 7 multiplicador
 - Escalera de Color: 100 puntos x 8 multiplicador
 - Escalera Real: 100 puntos x 8 multiplicador
 - Por Comodines: Por ultimo se van aplicando los comodines en orden aplicando sus efectos.

3.2. Flujo del programa

- Fase inicial: Cada jugador selecciona su nombre.
 - Fase de Juego: siguiendo a la fase inicial, se inicia esta fase, consta de 8 rondas en donde en cada ronda se inician dos fases:
 - Fase de Preparación: Se muestra una tienda donde el jugador se le muestran aleatoriamente 4 cartas.
 - Fase de ronda: La ronda muestra la cantidad de puntaje que debe alcanzar como mínimo el jugador para continuar en el juego donde dispone de 5 turnos. En cada turno se le reparten aleatoriamente 8 cartas del mazo a la mano del jugador. El jugador debe seleccionar sus cartas y puede:
 - jugarlas tratando de armar una mano de poker.
 - aplicar algún modificador usando un tarot.
 - o descartar las que quiera hasta tres veces, donde debe recibir del mazo las cartas suficientes para completar 8 .
- Una vez que jugó su mano, se puntúa utilizando los comodines de izquierda a derecha y se suma el puntaje y se avanza al siguiente turno.
- Una vez finalizados los turnos y sumando la cantidad suficiente para pasar de ronda, se vuelve a la fase de preparación.
- Fase final: El juego termina en dos instancias:
 - Caso positivo: el jugador supera las 8 rondas.
 - Caso negativo: cuando el jugador no logra sumar los puntos necesarios para pasar de ronda.

5. 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)*

6. 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.

7. 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.

8. Formas de entrega

Habrán 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.

9. 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.

10. 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 pide a una mano de póquer que se evalúe y se le asigne puntaje a un jugador.

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 8 cartas de su mazo.
3. Verificar que se puede jugar una mano de un mazo.
4. Verificar que al jugar una mano, se aplique el valor correspondiente.
5. Verificar que importe el orden en la puntuación de las cartas.
6. Verificar que al modificar una carta al utilizar un tarot que cambia sus puntos por 10, se aplique el puntaje correcto en el mazo.
7. Verificar que al modificar una carta utilizando un tarot que cambia su multiplicador a un x6 se aplique el valor correspondiente.

Entrega 2 (Semana 14 del calendario)

Pruebas (sin interfaz gráfica):

- Verificar que al tener un comodín que suma 8 al multiplicador se aplique correctamente
- Verificar que el jugador recibe un aumento correspondiente si tiene el comodín que aumenta el multiplicador por 3 si juega una escalera
- Verificar que el jugador si posee un comodin que suma 10 puntos por descarte, al descartar suma la cantidad correcta.
- Verificar que si el jugador posee un comodin que tiene chance 1 sobre 1000 de romperse se rompa correctamente.
- El jugador activa un comodín con una combinación de efectos bonus de mano jugada + puntaje aumentado + activación aleatoria
- 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

11. 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 refinan 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 refinan 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