[75.07 / 95.02]

Algoritmos y programación III

Trabajo práctico 2: AlgoHoot

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

La aplicación consiste en un juego por turnos, de dos o más jugadores conformado de un panel en el cual se mostrarán preguntas con múltiples opciones de respuesta.

Cada pregunta será mostrada N veces, una vez para cada jugador (al estar jugando en la misma computadora, será responsabilidad de cada jugador no mirar la pantalla mientras el otro responde).

Existen varios tipos de preguntas, que asignan puntaje en forma diferenciada a cada jugador dependiendo de cómo responde cada uno.

También existen opciones como los multiplicadores, la exclusividad de puntaje que cada jugador puede utilizar para mejorar sus oportunidades de obtener puntos y anuladores.

El objetivo del juego es lograr más puntos que el otro jugador respondiendo correctamente las preguntas.

Las preguntas a mostrar serán leídas desde un archivo de texto.

3.1. Elementos

● Jugador: Cada jugador tendrá una oportunidad de responder cada una de las preguntas.

● Panel: Es el panel donde se mostrarán las preguntas junto con las opciones de respuesta, estará dividido en dos partes:

- Sector pregunta: Es el sector en el cual se mostrará el texto de la pregunta.

-Sector respuestas: Es el sector en el cual se presentan las opciones de respuesta, y en el que cada jugador seleccionará la/s opcion/es que considere correctas.

● Preguntas:

- Existen distintos tipos de preguntas:

Verdadero/Falso clásico	
Opciones	2 (Verdadero o Falso)
Cómo funciona	Asigna un punto a cada jugador que responda con la opción correcta.

Verdadero/Falso con penalidad	
Opciones	2 (Verdadero o Falso)
Cómo funciona	Asigna un punto a cada jugador que responda con la opción correcta, y resta un punto a cada jugador que responda en forma incorrecta.

Multiple choice clásico	
Opciones	Desde dos y hasta 5
Cómo funciona	Asigna un punto a cada jugador que acierte TODAS las opciones correctas.

Multiple choice con puntaje parcial	
Opciones	Desde dos y hasta 5
Cómo funciona	Asigna un punto a cada jugador por cada opción correcta que seleccione (aunque no seleccione todas las correctas), siempre y cuando no haya seleccionado ninguna de las opciones incorrectas.

Multiple choice con penalidad	
Opciones	Desde dos y hasta 5
Cómo funciona	Asigna un punto a cada jugador por cada opción correcta que seleccione (aunque no seleccione todas las correctas), y resta un punto a cada jugador por cada opción incorrecta que seleccione.

Ordered choice	
Opciones	Desde dos y hasta 5
Cómo funciona	Se presentan opciones que deben ordenarse. Asigna un punto a cada jugador que coloque las opciones en el orden correcto.

Group choice	
Opciones	Desde dos y hasta 6, a colocar en dos grupos
Cómo funciona	Se presentan dos grupos vacíos, y opciones que deben colocarse en alguno de esos dos grupos. Asigna un punto a cada jugador que coloque las opciones en el grupo correcto.

Además de ser de diferente tipo, las preguntas pueden ser de diferentes temáticas (deporte, historia, ciencia, etc). No deben aparecer dos preguntas de la misma temática seguidas (salvo que sea la última pregunta que queda).

Estas preguntas, se tomarán de un archivo de texto que la cátedra va a proveer. Una vez leído el archivo, estas preguntas deberán ser puestas en un orden aleatorio a los usuarios (de esta forma las partidas serán distintas)

- Multiplicadores de puntaje:
  - ➔ Cada jugador tendrá dos multiplicadores (un multiplicador “x2” y un multiplicador “x3”) que podrá elegir utilizar en cualquiera de las preguntas “con penalidad”.
  - ➔ El jugador podrá asignar el multiplicador a la pregunta cuando le llegue el turno de responder.
  - ➔ Al calcular los puntos, se multiplicará x2 o x3 tanto los puntos por opción correcta como la penalidad, solo para el jugador que usó el multiplicador.
  - ➔ Al mostrar los resultados (después de que respondieron los jugadores), se deberá indicar que se usó este modificador
- Exclusividad de puntaje:
  - ➔ Cada jugador tendrá dos opciones de “exclusividad de puntaje” que podrá elegir utilizar en cualquiera de las preguntas que no tienen “penalidad”.
  - ➔ El jugador podrá asignar la exclusividad de puntaje a la pregunta cuando le llegue el turno de responder.
  - ➔ Al calcular los puntos, se asignará el doble del puntaje, pero solo en caso de que solo uno de los jugadores haya realizado la opción correcta. Es decir, si los N jugadores eligen la opción correcta, no se asignará puntaje a ninguno. Si uno de los jugadores eligió la opción correcta, ese jugador conseguirá el doble del puntaje.
  - ➔ Alcanza con que uno de los jugadores asigne la exclusividad de puntaje para que la regla afecte a todos los jugadores en esa pregunta.
  - ➔ Si por lo menos 2 jugadores asignan exclusividad de puntaje, entonces el efecto se multiplica por la cantidad de jugadores que la asignaron y se asignará el puntaje al jugador que elija la opción correcta (sólo si ningún otro jugador no eligió la opción correcta a su vez).
  - ➔ Al mostrar los resultados (después de que respondieron los jugadores), se deberá indicar que se usó este modificador

● Anulador de puntaje

- ➔ Cada jugador tendrá la opción de usar por única vez un “anulador de puntaje” contra los otros jugadores. Al usarlo, si otro jugador acierta la pregunta no recibirá los puntos (Aplica a todos los otros jugadores)
- ➔ Si otro jugador no acierta, no tendrá impacto en sus puntos
- ➔ Ejemplo de un caso de uso: Si los N jugadores usaron “”Anulador de puntaje”, ninguno va a recibir puntos por esa pregunta, acierten o no.
- ➔ Al mostrar los resultados (después de que respondieron los jugadores), se deberá indicar que se usó este modificador

3.2. Flujo del programa

● Fase inicial (una por cada jugador): Cada jugador selecciona su nombre.

● Fase de juego: Cuando ambos jugadores finalizaron su Fase inicial, se inicia esta fase, la cual consiste en mostrar a cada jugador las preguntas con sus opciones. Consta de un turno por vez por usuario..

● Fin: Se llega al límite de preguntas preestablecido, no hay más preguntas o se llega a X cantidad de puntos. Siendo el límite y X, valores que pueden ingresarse al iniciar la partida.

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 y diagrama de secuencia para el caso donde se pide a una pregunta de verdadero/falso clásico que evalúe una respuesta de un jugador y le asigne puntaje.

Entrega 1 (Semana 13 del calendario)

Pruebas (sin interfaz gráfica):

- Una Pregunta de Verdadero/Falso clásico recibe una lista de respuestas y asigna correctamente puntos a los jugadores que respondieron correctamente ✅.
- Una Pregunta de Verdadero/Falso clásico recibe una lista de respuestas y asigna correctamente puntos a los jugadores que respondieron de forma incorrecta ❌.
- Una Pregunta de Múltiple Choice clásico recibe una lista de respuesta de un jugador y asigna correctamente puntos a los jugadores que respondieron correctamente ✅.
- Una Pregunta de Múltiple Choice clásico recibe una lista de respuesta de un jugador y asigna correctamente puntos a los jugadores que respondieron de forma incorrecta ❌.
- Una Pregunta de Verdadero/Falso con penalidad recibe una lista de respuestas y asigna correctamente puntos a los jugadores que respondieron correctamente ✅.
- Una Pregunta de Verdadero/Falso con penalidad recibe una lista de respuestas y asigna correctamente puntos a los jugadores que respondieron de forma incorrecta ❌.
- Una Pregunta de Múltiple Choice con penalidad recibe una lista de respuesta de un jugador y asigna correctamente puntos a los jugadores que respondieron correctamente ✅.
- Una Pregunta de Múltiple Choice con penalidad recibe una lista de respuesta de un jugador y asigna correctamente puntos a los jugadores que respondieron de forma incorrecta ❌.

Entrega 2 (Semana 14 del calendario)

Pruebas (sin interfaz gráfica):

- Agregar pruebas, para el tipo restante de preguntas, donde cada jugador responde correcta o incorrectamente y su puntaje se ve incrementado o reducido (sin incluir "anulador de puntaje" o "exclusividad de puntaje")
- Agregado de multiplicadores a cada tipo de pregunta, verificando que cumplan con el efecto indicado (x2 o x3)
- Agregar "anulador de puntaje" para los jugadores.
- Agregar "exclusividad de puntaje" para los jugadores.
- Planteo inicial de interfaz gráfica, pantalla donde se muestra una pregunta con sus opciones

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