Instituto tecnológico de Costa Rica

Unidad de Computación

Trivia Quirk

Adrian Jose Villalobos Peraza, Isaac Ramírez Rojas

Sede San Carlos

11/15/2023

Introducción

Este proyecto es realizado como una asignación del curso Programación Orientada a Objetos, el objetivo principal de este es familiarizar al estudiante con distintos conceptos de dicha rama de la programación, además, de tener un acercamiento con elementos clave en el desarrollo de software, como interfaz gráfica de usuario y pruebas unitarias. Además, en este proyecto también se promueve el trabajo en equipo, ya que era necesario incluir secciones creadas por otros grupos de trabajo.

Los conceptos mencionados anteriormente son sumamente importantes para la formación de los ingenieros en computación, esto se puede apreciar en el caso de las pruebas unitarias, ya que, según Durga Prasad, mediante el uso de estas pruebas, se puede examinar cada sección del código aisladamente, lo cual permite mejorar la calidad, detectar errores tempranos, agilizar el proceso y reducir costos (Prasad, 2023).

El proyecto consiste en la creación de una trivia que logre abarcar una serie de puntos, por ejemplo, crear categorías de preguntas, gestionar usuarios, manejo de archivos, trato de excepciones y manejo de errores, crear una interfaz gráfica agradable, entre otros puntos clave para el adecuado funcionamiento de esta.

Análisis del problema

El problema planteado, como se mencionó anteriormente, consiste en la elaboración de una trivia, la cual debe tener ciertas categorías y preguntas correspondientes. Para abarcar dicho punto, se pueden crear distintas clases del tipo Enum en java, que permitirán almacenar las preguntas de manera que el acceso a estas se pueda realizar de forma eficiente. Por otro lado, se puede crear una única clase para las categorías, la cual puede implementar el patrón singleton, de modo que se asegure que no se crucen las categorías, generando así errores.

Por otro lado, las preguntas mencionadas con anterioridad deben apegarse a cierto esquema para la distribución del puntaje, bajo ciertos parámetros, se podría decir que cuantas más veces se falle la pregunta, más puntos valdrá esta. Para manejar esto, se puede hacer uso de archivos, donde se almacene la cantidad de aciertos y desaciertos, de modo que el sistema consulte estos archivos a la hora de asignarle el puntaje a los jugadores.

En cuanto a las partidas se refiere, cada partida está dada por una categoría seleccionada, por lo tanto, se debe asegurar que a cada partida se le asigne una categoría, además, la partida puede ser terminada en cualquier momento (o cuando se acaben las preguntas), por lo que se deben manejar estos escenarios al momento de elaborar la interfaz de usuario.

Respecto a la interfaz de usuario, estaba debe tener un menú principal, donde se pueda tocar un botón para iniciar una partida, durante la creación de usuarios, es necesario que el sistema tenga la capacidad de consultar cuantos jugadores jugaran la partida, así como sus nombres y la categoría seleccionada.

Por otro lado, en la trivia es necesario que se muestren preguntas obtenidas de manera aleatoria, según corresponda la categoría seleccionada, además, se deben mostrar otros parámetros como el tiempo restante, la puntuación y el turno actual del jugador.

Solución el problema

Para abarcar los puntos anteriores, cabe destacar que se implementó un paquete Jar proporcionado por el profesor, que contiene las interfaces y excepciones necesarias para la elaboración del proyecto, así como una categoría externa proporcionado por un grupo de estudiantes.

Para empezar, las preguntas fueron elaboradas en una clase enum, que luego es consultada por la clase de categoría. Dicha clase puede adoptar cuatro nombres (uno para cada tipo de pregunta a jugar), de modo que este nombre luego es almacenado en la partida correspondiente, con el objetivo de que se puedan estar jugando varias partidas al mismo tiempo, incluso si son de temas distintos.

Para la elaboración de la partida, se gestionó mediante la clase categoría, la cual contiene un método para crear una partida cuando se desee, de modo que se pueda tener en una lista, todas las partidas actuales, las cuales contienen a los jugadore y otros parámetros necesarios para llevar a cabo el juego.

En cuanto a la interfaz gráfica, se manejaron tres clases según cada escenario, una para el menú principal, otra para el registro de usuario y otra para la trivia como tal, de modo que se pueden estar realizando distintas acciones de manera simultánea, por ejemplo, jugar una partida mientras se registran los datos necesarios para empezar otra, o jugar varias partidas al mismo tiempo. Cabe destacar que el marcador de puntaje, a pesar de estar presente en la interfaz gráfica de la trivia, se manejó como una clase aparte, para implementar el patrón de diseño Observer.

Para finalizar, para las pruebas unitarias se hizo uso de la librería JUNIT, la cual permite realizar distintas pruebas individuales a cada método de cada clase. El código creado logro pasar estas pruebas satisfactoriamente.

Análisis de resultados

TAREA	***ESTADO***	***OBSERVACIONES***
El usuario puede crear partidas y para estas seleccionar las categorías de preguntas.	*Completo*	
Las puntuaciones se establecerán por categoría de pregunta.	*Completo*	
Cada partida deberá reflejarse en una ventana independiente y se pueden estar jugando múltiples partidas a la vez.	*Completo*	
Las partidas terminan cuando los jugadores completen las 30 preguntas o cuando los mismos definan terminarlo. Al finalizar las partidas se mostrarán las puntuaciones finales en un reporte ordenado por puntuación y por categoría y deberán guardarse	*Completo*	

automáticamente las puntuaciones globales en archivos.		
Cada pregunta tendrá 20 segundos para ser respondida, de no contestar en el tiempo establecido se registrará como fallida la respuesta.	*Completo*	
En el modo multijugador se alternará pregunta a pregunta por cada jugador y las preguntas no se repetirán entre jugadores.	*Completo*	
Implementación de los patrones singleton y observer	*Incompleto*	*Solo se implementó singleton.*
Tanto las puntuaciones, así como la estadística de acierto de las preguntas deberá ser almacenado en archivos.	*Completo*	
Fuertemente manejo excepciones de todo el juego.	*Completo*	
A cada pregunta contestada - en modo global para todas las partidas- se registrar la cantidad de veces que se haya contestado de manera correcta o incorrecta.	*Completo*	
En todo momento de las partidas se mostrará la puntuación actual de cada jugador conforme avancen las preguntas.	*Completo*	
Cada acierto de pregunta tiene una puntuación que se rige por ciertos parámetros.	*Completo*	
Documentación adecuada según JavaDoc.	*Completo*	
Utilización del Jar proporcionado por el profesor.	*Completo*	
Creación de al menos 3 categorías y una cuarta implementada desde otro proyecto.	*Completo*	
Elaboración de pruebas unitarias.	*Completo*	

Conclusiones

En este proyecto de Programación Orientada a Objetos, se logró desarrollar una trivia funcional que aborda diversos aspectos cruciales en el ámbito de la programación. A través de la implementación

de conceptos como pruebas unitarias, interfaces gráficas de usuario, y trabajo en equipo, se ha proporcionado a los estudiantes una experiencia integral en el desarrollo de software.

La creación de la trivia involucró la resolución de problemas complejos, desde la gestión de preguntas y categorías hasta la implementación de partidas concurrentes y la manipulación de archivos para almacenar puntuaciones. La adopción de patrones de diseño como el Singleton y el Observer contribuyó a la eficiencia y modularidad del sistema.

En términos de resultados, el proyecto cumplió con la mayoría de las tareas propuestas. Se destacan logros como la creación de partidas con selección de categorías, establecimiento de puntuaciones por categoría, gestión de múltiples partidas simultáneas y finalización ordenada con informes de puntuación.

Además, se logró la implementación de mecanismos de tiempo para responder preguntas, alternancia en el modo multijugador, almacenamiento de datos en archivos y manejo efectivo de excepciones en todo el juego. Aunque se logró implementar el patrón Singleton, la aplicación del patrón Observer quedó incompleta, lo que podría haber mejorado aún más la estructura del código.

No obstante, la utilización del paquete Jar proporcionado por el profesor, la documentación adecuada según JavaDoc, la creación de categorías y la implementación de pruebas unitarias se llevaron a cabo de manera exitosa, demostrando una comprensión sólida de los conceptos y herramientas utilizados.

Recomendaciones

Para ampliar el alcance de este proyecto, se podrían buscar medios para generar interfaces de usuario más agradables, aparte de swing. Por ejemplo, se puede hacer esta trivia, pero en una aplicación web, con el objetivo de utilizar nuevas tecnologías de un ámbito completamente nuevo, así como que la aplicación sea más accesible para todo el mundo.

Por otro lado, se podría generalizar un poco más el proyecto, de modo que resulte más sencillo implementar las categorías de otros grupos, o incluso dividir el proyecto en grupos, por ejemplo, que un grupo trabaje toda la interfaz gráfica y otro la lógica detrás del programa, lo cual se asemeja más a cómo trabaja una empresa y se aseguraría un mejor trabajo.

Referencias bibliográficas

Prasard, D. (2023). Explicación de las pruebas unitarias: qué es, por qué es importante y cómo empezar. GEEKFLARE. https://geekflare.com/es/unit-testing-guide/