Universidad Europea

PLAN DE REVISION – TIENDA DE VIDEOJUEGOS

25/05/2023

Autores

Álvaro Dans Montilla
Liam Wittels Beneish

ÍNDICE

Formato

El formato adecuado del código puede mejorar su legibilidad y comprensión. Aquí hay algunos pasos para la optimización del formato:

- Utilizar espacios o tabuladores de manera consistente para identar el código y resaltar la estructura de este.

- Usar saltos de línea para separar bloques lógicos de código, como declaraciones de variables, bucles y estructuras de control.

- Colocar los corchetes de apertura y cierre en líneas separadas para resaltar claramente los bloques de código.

Mantener una alineación coherente para mejorar la legibilidad.

Mensajes de error

Los mensajes de error claros y precisos son fundamentales para identificar y solucionar problemas en el código. Aquí hay algunos pasos para asegurar que los mensajes de error sean adecuados y suficientes:

- Proporcionar mensajes de error descriptivos que indiquen claramente qué salió mal.

- Incluir información relevante en los mensajes de error, como el número de línea o la ubicación del error.

- Manejar y comunicar errores inesperados o excepciones de manera adecuada.

Estructuras de datos

Se debe tener siempre en cuenta el tipo de datos correcto ya que, si se utiliza siempre como recurso la lista convencional, por ejemplo, no se está aprovechando todo el potencial de las estructuras de datos.
En cuanto a este apartado se debe revisar:

- **Tipo de estructura idónea**: por ejemplo, si se quiere almacenar una lista de usuarios y se quiere operar uno en concreto, lo más correcto sería utilizar un mapa, ya que la búsqueda se realizará de forma más ágil.

- **Utilización correcta de estas estructuras**: si se utilizan este tipo de estructuras, debe asegurarse de que se van a usar de la forma correcta si se conocen, para que su uso tenga un sentido y suponga una ventaja real.

Refactorización

La refactorización es uno de los puntos clave de la programación. Mucho código agrupado en una sola función y con una lógica variada puede resultar difícil de comprender, y por lo tanto, difícil de mantener y escalar.
En cuanto a la refactorización se debe vigilar:

- **Refactorización en métodos**: en la medida de posible y para mayor legibilidad, separar la lógica de la aplicación en distintos métodos para poder comprender mejor qué es lo que se está haciendo, es una ventaja a la hora de conseguir un código más limpio y legible. Además, si se localiza un fallo y se tiene que arreglar un método, impactará en menor medida la modificación del código de fuera de esta función, por lo que se mejorara la localización de los errores, y se podrán tratar de forma más aislada y sencilla.
- **Reutilización**: un código reutilizable siempre nos va a ayudar a no sobre cargar nuestra aplicación con código que no necesitamos. Esto nos ahorrará tanto limpieza, como complejidad y efectividad, ya que tener código duplicado no es nada optimo, y puede llevar a repetir fallos en lugares distintos.

Uniformidad

El tema de la uniformidad es muy importante, y sobre todo si se trabaja en equipo. Marcar unas directrices de formato siempre nos va a ayudar a tener un código más limpio y legible. Esto nos puede ayudar sobre todo a la hora de encontrar y arreglar bugs, ya que, si todo sigue un mismo formato, es más fácil de leer y comprender.
Se debe tener en cuenta:

- **Formatos universales**: con esta parte nos referimos a todas esas convenciones universales que se siguen a la hora de programar. Por ejemplo: nombres de las clases que empiezan por mayúscula, las variables que empiecen con minúscula, el conocido camel case, y un largo etcétera.
- **Nomenclatura**: una nomenclatura regular tanto en las variables como en los métodos es muy importante y resulta de gran ayuda. Esto nos va a permitir entender mejor el código a la hora de leerlo, y saber exactamente qué tipo de dato estamos tratando a la hora de programarlo, o que hace el método en cuestión que tenemos que revisar, por ejemplo. Un ejemplo de esto sería utilizar siempre nombres en inglés, o cierto formato en las constantes del programa, o definir un estándar para los nombres de los métodos o clases.

Resumen revisión de código Librería

En relación con el plan propuesto se han encontrado las siguientes mejoras posibles:

- Formato uniforme: puesto que de forma genérica la programación se realiza en inglés, si se quisiese programar en español, sería conveniente que todos los métodos y variables se escribiesen en español, pero que, si se mezclan, se puede complicar la legibilidad del código.
- Error no controlado al añadir libro: este error se produce al añadir un nuevo libro, y meter una letra en el isbn, ocasionando una NumberFormatException. Si bien es cierto que se controla el error al introducir un isbn de un libro con un catch, éste después no devuelve ningún feedback del error. Esto deja con cierta incertidumbre puesto que el usuario no sabría que ha habido un fallo al introducir
- Comentarios innecesarios: se puede apreciar comentarios que no indican mucho más, se pudiesen borrar para poder mantener una estructura limpia y más organizada.
- Error no controlado en reservas: Al modificar una reserva como bibliotecario, es posible introducir un ID de usuario que no existe. Esto se debería controlar ya que puede conllevar un posterior fallo.
- Control en campos introducidos: Al introducir los datos de reserva para modificarla, no se controla que todos los campos estén rellenos para que no falle el programa.