Ejercicio 6 – Análisis

Josué Marroquin 22397

Competencias a desarrollar:

- Identifica los requisitos funcionales del sistema a desarrollar a partir del problema planteado.
- Elabora el análisis y el diseño del programa utilizando un diagrama de clases en UML.
- Diseña la interface que necesitará para generalizar su programa.

Problema a resolver:

De la empresa Apple se le ha pedido que realice una simulación de su nuevo iPod. Este iPod consta de control de volumen, control de listado general de canciones, canciones favoritas, un botón de encendido/apagado y una opción para bloquear los controles. Por ser una simulación deberá incluir una opción que permita ver el estado del iPod en cualquier instante.

Las características específicas del iPod son:

- Control de volumen: Tiene un rango de 0 – 10 y el cambio se hace en pasos de 0.5.
- Listado de canciones:
    - Recorrido de listado general: Se recorre de una en una cada canción, ascendente (NEXT) o descendentemente (PREVIOUS). Tiene una capacidad máxima de 50 canciones.
    - Guardar canción en favoritas: La canción actual se guarda en el lugar elegido por el usuario. Se tienen 10 espacios para favoritas (TOP TEN). Al inicio de la simulación debe estar vacía.
    - Ir a una canción guardada en favoritas: Se escoge alguna posición de la lista y se va directo a esa canción.
- Encendido/Apagado: Cambia de encendido a apagado y viceversa
- Bloqueado/Desbloqueado: Bloquea o desbloquea los controles del iPod

Para el estado del iPod debe mostrar las características actuales del mismo. Si está encendido o no, bloqueado o no, el nivel de volumen, que canción está seleccionada en ese momento, que canciones tiene almacenadas en favoritas, etc.

Observaciones y excepciones:

- Si el iPod está apagado no se debe permitir ningún cambio de estado del iPod, únicamente encender
- Si el iPod está bloqueado no se debe permitir ningún cambio de estado del iPod, únicamente desbloquear y ver estado.
- Al escoger una canción favorita debe verificar que seleccione una posición válida (1 al 10)

- La lista general es una secuencia circular, es decir, la siguiente canción de 50 es 1 y la anterior a la primera canción es la que está en la posición 50.
- Muestre mensajes de error descriptivos cuando sea necesario.

Tareas:

Tarea #1. Realice el análisis del problema planteado. Tenga en cuenta que parte de su programa va a ser utilizado por otro compañero de clases, por lo que debe hacerlo lo más general y claro posible.

Tarea #2. Elabore el diagrama de clases (UML) del sistema.

Tarea #3. Diseñe la interface que se usará para intercambiar la GUI, o la clase que interactúa con el usuario.

Tarea #4. Intercambie su clase de interacción con el usuario con otro compañero. A la clase que reciba solo puede cambiarle **una línea de código** para que funcione con su programa.

Tarea #5. Describa el proceso de diseño que tuvo que hacer el equipo para poder cambiar solo una línea de código con el otro compañero. Describa los tropiezos que tuvo y los cambios que tuvo que hacer para poder usar la clase intercambiada con su sistema.

Material a entregar y fechas de entrega:

- **Lunes 7 de noviembre a las 9:20**.
  - Archivo .pdf con el análisis del sistema
  - Archivo de imagen (.png, .jpg) con el diagrama de clases diseñado
  - Archivos .java con la implementación del sistema.
- **Jueves 9 de noviembre a las 9:20**.
  - Archivo .pdf con la reflexión y la descripción de los cambios que hay que realizarle al sistema para que funcione con otra interfaz
  - Archivos .java con el sistema incluyendo la clase de interacción del usuario del grupo con el que le tocó intercambiar.

Evaluación:

Requisitos Funcionales **(5 puntos)**:

- Identifica correctamente todo lo que debe hacer el programa según la situación planteada.

Análisis **(24 puntos)**:

- Identificación de Clases, atributos y métodos (**9 puntos**):
  - (**3 puntos**) Se identifican correctamente las clases que se necesitan para resolver el problema. El número de clases identificadas es suficiente para darle solución a la situación planteada.

    - Clase IpodSiumator y Canción

- (**3 puntos**) Se identifica correctamente la jerarquía de clases (herencia) que se presenta en el problema planteado.
- (**3 puntos**) Se explica correctamente y de forma lógica el propósito de cada una de las clases.
  La clase IpodSimulator contiene todos los métodos que le perimitaran funcionar así como sus atributos

- Atributos de las clases (**10 puntos**):
  - (**2 puntos**) Se identifican correctamente todos los atributos de cada una de las clases seleccionadas. Son los necesarios para resolver el problema planteado.

    Para canción
    private String _title;
    private String _artist;
    private String _album;
    private String _duration;
    private int _id;

    Para el ipod
    private float volume;
    private boolean actual_state;
    private int actual_index;
    private boolean actual_locked_state;

Diseño **(18 puntos)**:

- Diagrama de Clases en UML
  - (**5 puntos**) Se representa correctamente la jerarquía de clases identificadas, utilizando las relaciones correctas.

    UMLEJ6.PNG incluida en zip y repositorio.

  - (**4 puntos**) Se representan correctamente el resto de las relaciones entre las clases identificadas.
  -
  - (**5 puntos**) Todos los atributos y métodos tienen correctamente representada la visibilidad que poseen.
  - (**4 puntos**) Los métodos tienen todos los parámetros y valor de retorno correctamente representados.
  -

Implementación **(53 puntos)**

- (**20 puntos**) Están correctamente implementados los requisitos funcionales identificados. Se da solución con ellos al problema planteado.
- (**10 puntos**) El sistema es completamente reactivo a entradas incorrectas, mostrando errores amigables con el usuario.
- (**23 puntos**) Está correctamente implementado el polimorfismo basado en interfaces. Se logró cumplir el objetivo de intercambiar con éxito la clase de interacción con el usuario cambiando solo una línea de código.
- (**5 puntos)** Está totalmente documentado usando el estándar JavaDoc.

Nota: Para que sea calificada la implementación deben cumplirse los siguientes requisitos:

- El sistema no tiene errores de sintaxis ni en tiempo de compilación que impidan la revisión del mismo.
- La implementación debe coincidir en un 95% con el diseño. Los cambios que surjan deben ser mínimos y deben estar documentados en el código.