Universidad del Valle de Guatemala
Programación Orientada a Objetos
Tomás Gálvez
Sección 11
CC2008

Laboratorio IV

Interfaces

Abner Iván García Alegría 21285
José Daniel Gómez Cabrera 21429

Universidad del Valle de Guatemala
Programación Orientada a Objetos
Tomás Gálvez
Sección 11
CC2008

Análisis

1. ¿Qué debe hacer el programa?

El programa debe de realizar una simulación de radios para vehículos de diferente tipo, aprovechando el polimorfismo por interfaces. Cada radio debe de tener propiedades y métodos distintos para satisfacer las necesidades de cada tipo de carro.

Externamente:

- Mostrar menús al usuario para darle información acerca del radio.
- Recibir información del usuario parar determinar acciones, como encender o apagar el radio.
- Utilizar métodos de vista para que el usuario pueda observar el tipo de radio y sus propiedades.

Internamente:

- Utilizar los datos ingresados en vista para modificar las propiedades del programa.
- Utilizar setters y getters para poder manejar el flujo del programa.
- Aprovechar el polimorfismo por medio de las interfaces para poder desarrollar métodos abstractos y también métodos propios de cada clase.

2. ¿Qué clases conformarán el modelo para dar solución a la situación planteada?

Clases Modelo:

1. Radio <<Interface>>: interfaz para desarrollar los diferentes tipos de radio
2. ClaseS: Clase que implementa la interfaz, con métodos abstractos y propios de la clase.
3. ClaseA: Clase que implementa la interfaz, con métodos abstractos y propios de la clase.
4. ClaseC: Clase que implementa la interfaz, con métodos abstractos y propios de la clase.
5. MercedesBenz: Clase que administra el radio local, cambiando el tipo de radio, el estado (encendido/apagado), entre otras propiedades según el usuario lo desee.

Universidad del Valle de Guatemala
Programación Orientada a Objetos
Tomás Gálvez
Sección 11
CC2008

3. ¿Qué propiedades y métodos tendrá cada clase (incluyendo Vista y Controlador)? ¿Qué tipo deben tener las propiedades y métodos de cada clase?

Modelo:

1. Radio <<Interface>>
   - Propiedades:
     - Sin propiedades, debido a que todas las propiedades del radio deben ser alteradas por decisión del usuario, por lo que no se pueden instanciar en una interfaz, debido a que todas las propiedades de una interfaz se hacen finales.

   - Métodos:
     1. set_estado(): void //encendido o apagado
     2. subir_volumen(): void
     3. bajar_volumen(): void
     //Modo Radio
     4. cambiar_fm_am(int): void
     5. cambiar_emisora(): void
     6. guardar_emisora(String): void
     7. cargar_emisora(): void
     8. get_estado_radio(): boolean
     9. get_volumen_radio(): int
     10. to_String(): String[]
     //Modo reproduccion
     11. seleccionar_lista_reproduccion(int): void
     12. cambiar_cancion(): void
     13. escuchar_cancion(): void
     //Modo Telefono
     14. conectar_desconectar_telefono(): void
     15. mostrar_contactos(): String
     16. llamar_contacto(): String
     17. finalizar_llamada(): String

2. ClaseS
   - Propiedades:
     1. estado: boolean //true: encendido, false: apagado
     2. emisoras_guardadas: ArrayList<String>
     3. lista_reproduccion: ArrayList<String>
     4. estado_telefono: boolean
     5. bocinas_o_auriculares: String

   - Métodos:
     // Métodos heredados por interfaz
     1. ClaseS()
     2. cambiar_bocinas_o_auriculares(): void
     3. planificar_viaje(String[]): void

3. ClaseA

- Propiedades:
    1. estado: boolean //true: encendido, false: apagado
    2. emisoras_guardadas: ArrayList<String>
    3. lista_reproduccion: ArrayList<String>
    4. estado_telefono: boolean
    5. ultimo_contacto: String

- Métodos:
    //Métodos hederados por la interfaz
    1. ClaseA()
    2. llamar_ultimo_contacto(): String
    3. ver_tarjetas_presentacion(): String

4. ClaseC

- Propiedades:
    1. estado: boolean //true: encendido, false: apagado
    2. emisoras_guardadas: ArrayList<String>
    3. lista_reproduccion: ArrayList<String>
    4. estado_telefono: boolean
    5. pronostico_del_tiempo: String

- Métodos:
    //Métodos hederados por la interfaz
    1. ClaseC()
    2. cambiar_llamada_en_espera(): String
    3. ver_pronostico_del_tiempo(): String

5. MercedesBenz

- Propiedades:
    1. radio: Radio (Polimórfico)
    2. radio_actual: String

- Métodos:
    1. MercedesBenz()
    2. encerder_apagar_radio(): void
    3. cambiar_tipo_radio(): void
    4. subir_volumen(): void
    5. bajar_volumen(): void
    6. acciones(int): void
    7. agregar_emisora(String): void

3. Vista:

- Propiedades:
    1. Scanner: scan
- Métodos:
    1. bienvenida()
    2. obtener_string(): String
    3. obtener_int(): int
    4. mostrar_menu_radio_encendido(): int //Menu para cuando el radio esté encendido
    5. mostrar_menu_radio_apagado(): int //Menu para cuando el radio esté encendido
    6. cambiando_fm_am(): void
    7. cambiando_emisora(): void //±0.5
    8. guardar_emisora(): String
    9. cargando_emisora(): void
    10. seleccionar_lista_reproduccion(): int
    11. mostrar_listas_reproduccion(): void
    12. cambiando_cancion(): void
    13. mostrar_cancion(): void
    14. mostrar_contactos(): void
    15. llamar(): void
    16. finalizar_llamada(): void
    17. planificando_viaje(): String[]
    18. mostrar_pronostico_tiempo(): void
    19. mensaje_error(): void
    20. despedida(): void

Controlador:

7. Controlador
    - Propiedades: Sin propiedades.

    - Métodos:
        1. SimuladorMercedesBenz(): void

8. MAIN
    - Propiedades: Sin Propiedades.

    - Métodos: Método Main (String[] args).

Universidad del Valle de Guatemala
Programación Orientada a Objetos
Tomás Gálvez
Sección 11
CC2008

4. ¿Cuáles deben ser los modificadores de visibilidad, tipos y descripción de los miembros en cada clase?

Clase	Propiedad/Método	Visibil.	Descripción
1. Radio <<Interface>>	- Propiedades		
	//Sin propiedades, debido a que todas las propiedades del radio deben ser alteradas por decisión del usuario, por lo que no se pueden instanciar en una interfaz, debido a que todas las propiedades de una interfaz se hacen finales.		
	- Métodos (parámetro)		

	1. set_estado(): void //encendido o apagado
	2. subir_volumen(): void
	3. bajar_volumen(): void
	//Modo Radio
	4. cambiar_fm_am(int): void
	5. cambiar_emisora(): void
	6. guardar_emisora(String): void
	7. cargar_emisora(): void
	8. get_estado_radio(): boolean
	9. get_volumen_radio(): int
	10. to_String(): String[]
	//Modo reproduccion
	11. seleccionar_lista_reproduccion(int): void
	12. cambiar_cancion(): void
	13. escuchar_cancion(): void
	//Modo Telefono
	14. conectar_desconectar_telefono(): void
	15. mostrar_contactos(): String
	16. llamar_contacto(): String
	17. finalizar_llamada(): String
		Protected
		Protected
		Protected
		Protected
		
		Protected
		Protected
		Protected
		Protected
		Protected
		Protected
		Protected
		Protected
		
		Protected
		Protected
		Protected
		Protected
		
		Protected
		Protected
		Protected
		Protected
		Protected
			Método para cambiar el estado del radio.
			Método para subir el volumen del radio.
			Método para bajar el volumen del radio.
			
			Método para cambiar la frecuencia del radio de am a fm o viceversa.
			Método para cambiar de emisora.
			Método para guardar emisoras.
			Método para obtener una emisora en específico. (getter)
			Getter de estado del radio para saber si esta encendido o apagado.
			Getter del volumen del radio.
			Método para obtener diferente información del radio.
			
			Método para seleccionar la lista de reproducción del radio.
			Método para cambiar de canción.
			Método para escuchar o pausar canción.
			
			Método para conectar o desconectar teléfono.
			Método para obtener un contacto.
			Método para llamar a un contacto existente.
			Método para finalizar la llamada con el contacto.

Universidad del Valle de Guatemala
Programación Orientada a Objetos
Tomás Gálvez
Sección 11
CC2008

2.ClaseS	- Propiedades		
	1. estado: boolean //true:encendido,false: apagado 2. emisoras_guardadas: ArrayList<String> 3. lista_reproduccion: ArrayList<String> 4. estado_telefono: boolean 5. bocinas_o_auriculares: String	Private Private Private Private Private	Propiedad booleana para saber el estado del radio. (encendido/apagado) Arraylists de emisoras guardadas. Arraylists de listas de reproducción de música. Propiedad booleana del estado de teléfono. Propiedad String de los auriculares o bocinas para ClaseS.
	- Métodos (parámetro)		
	// Métodos heredados por interfaz 1. ClaseS() 2. cambiar_bocinas_o_auriculares(): void 3. planificar_viaje(String[]): void	 Public Public Public	 Constructor. Método para cambiar la propiedad de bocinas o auriculares. Método para planificar un viaje.
3. ClaseA	- Propiedades		
	1. estado: boolean //true: encendido, false: apagado 2. emisoras_guardadas: ArrayList<String> 3. lista_reproduccion: ArrayList<String> 4. estado_telefono: boolean 5. ultimo_contacto: String	Private Private Private Private Private	Propiedad booleana para saber el estado del radio. (encendido/apagado) Arraylists de emisoras guardadas. Arraylists de listas de reproducción de música. Propiedad booleana del estado de teléfono. Propiedad String de el último contacto para ClaseA.
	- Métodos (parámetro)		
	//Métodos hederados por la interfaz 1. ClaseA() 2. llamar_ultimo_contacto(): String 3. ver_tarjetas_presentacion(): String	 Public Public Public	 Constructor. Método para llamar al último contacto. Método para ver una tarjeta de presentación.
4. ClaseC	- Propiedades		
	1. estado: boolean //true: encendido, false: apagado 2. emisoras_guardadas: ArrayList<String> 3. lista_reproduccion: ArrayList<String> 4. estado_telefono: boolean 5. pronostico_del_tiempo: String	Private Private Private Private Private	Propiedad booleana para saber el estado del radio. (encendido/apagado) Arraylists de emisoras guardadas. Arraylists de listas de reproducción de música. Propiedad booleana del estado de teléfono. Propiedad String del pronóstico del tiempo para ClaseS.
	- Métodos (parámetro)		
	//Métodos hederados por la interfaz 1. ClaseC() 2. cambiar_llamada_en_espera(): String 3. ver_pronostico_del_tiempo(): String	 Public Public Public	 Constructor. Método para cambiar la llamada a llamada en espera. Método para obtener el pronóstico del tiempo.

Universidad del Valle de Guatemala
Programación Orientada a Objetos
Tomás Gálvez
Sección 11
CC2008

5. MercedesBenz	- Propiedades		
	1. radio: Radio (Polimórfico) 2. radio_actual: String	Private Private	Método que almacena el objeto polimórfico de tipo Radio. Método para determinar el radio actual de manera más fácil.
	- Métodos (parámetro)		
	1. MercedesBenz() 2. encerder_apagar_radio(): void 3. cambiar_tipo_radio(): void 4. subir_volumen(): void 5. bajar_volumen(): void 6. acciones(int): void 7. agregar_emisora(String): void	Public Public Public Public Public Public Public	Constructor. Método para encender o apagar el radio desde el simulador. Método para cambiar el tipo de radio desde el simulador. Método para subir volumen desde el simulador. Método para bajar volumen desde el simulador. Método que administra las acciones del radio. Método para agregar una emisora al radio.
6. Vista	- Propiedades		
	1. Scanner: scan	Private	Scanner para obtener datos.
	- Métodos (parámetro)		
	1. bienvenida() 2. obtener_string(): String 3. obtener_int(): int 4. mostrar_menu_radio_encendido(): int //Menu para cuando el radio esté encendido 5. mostrar_menu_radio_apagado(): int //Menu para cuando el radio esté encendido 6. cambiando_fm_am(): void 7. cambiando_emisora(): void //+0.5 8. guardar_emisora(): String 9. cargando_emisora(): void 10. seleccionar_lista_reproduccion(): int 11. mostrar_listas_reproduccion(): void 12. cambiando_cancion(): void 13. mostrar_cancion(): void 14. mostrar_contactos(): void 15. llamar(): void 16. finalizar_llamada(): void 17. planificando_viaje(): String[] 18. mostrar_pronostico_tiempo(): void 19. mensaje_error(): void	Public Private Private Public Public Public Public Public Public Public Public Public Public Public Public Public Public Public Public	Bienvenida para el usuario. Método para obtener un String de parte del usuario. Método para obtener un entero de parte del usuario. Menú cuando el radio está encendido. Menú cuando el radio está apagado. Método de mensaje para el usuario. Método de mensaje para el usuario. Método para agregar una emisora. Método de mensaje para el usuario. Método para seleccionar una lista de reproducción. Método de mensaje para el usuario. Método de mensaje para el usuario. Método de mensaje para el usuario. Método de mensaje para el usuario. Método de mensaje para el usuario. Método de mensaje para el usuario. Método que obtiene los datos para planificar un viaje. Método para mostrar el pronóstico del tiempo. Método de mensaje error para el usuario.

	20. despedida(): void	Public	Método de mensaje de despedida para el usuario.
7. Controlador	- Propiedades		
	Sin propiedades.		
	- Métodos(Parámetros)		
	SimuladorMercedesBenz(): void	Public	Método para ejecutar el simulador.
8. MAIN	- Propiedades		
	Sin Propiedades		
	- Métodos (parámetro)		
	1 Main(String[] args)	Public	Método Main con la instancia del simulador de MercedesBenz.

Universidad del Valle de Guatemala
Programación Orientada a Objetos
Tomás Gálvez
Sección 11
CC2008

Posibles fallos y Exceptions:

1. RunTimeException:

   Este es el error más común. Lo ideal es que se maneje con Try Catch, y mensajes para identificar la clase y método donde se ubica el error. Esto puede lograrse de muy buena manera con el método Throw. En la cual podemos adjuntar un mensaje personalizado para que se concatene al mensaje de la excepción que Java genere.

2. InputMissMatchException:

   Esta excepción se podría identificar en un ingreso de datos al sistema por parte del usuario. Es muy importante manejarla con cuidado, ya que podría afectar a todo el programa y ocasionar un reinicio.

   Para poder controlar este error, cubriré cada entrada del usuario con un While y un mensaje al usuario, para obligar al usuario a ingresar el dato correctamente, para poder obtener el dato correcto, mandárselo al Controlador, y del Controlador al Modelo.

3. IndexOutOfBoundsException:

   Esta excepción se ocasiona muy seguido cuando se trabaja con Arrays de tamaño estático. Este error ocurre cuando tratamos de acceder u obtener un valor fuera del rango del arreglo estático. Por lógica, también podría ocurrir con un arrgelo dinámico, si se trata de acceder a un valor de índice mayor al del arreglo dinámico.

   Para manejar esta excepción, trabajaré siempre utilizando los métodos lenght() para Arrays y size() para ArrayLists. De esta manera, siempre se sabrá exactamente el tamaño de cada arreglo, y se evitará un índice fuera de bandas.