Adrian Arimany Zamora
November 15, 2024

Lab 4 - POO - Polimorfismo a través de Interfaces

Repository at: https://github.com/adrianArimany/Lab4_poo

(Please use the readme.md to run the program).

The ULM can be found on \root\Analysis\ULM.drawio, due to the quantities of classes, the ULM can't be exported to a pdf, so it can only be opened via .drawio which is the application I used to make the ULM.

There is an additional diagram, in \root\Analysis\Program_System_Design_diagram.drawio, it gives a good idea of how the program actually works, it summarises the analysis into a single diagram.

Objective:

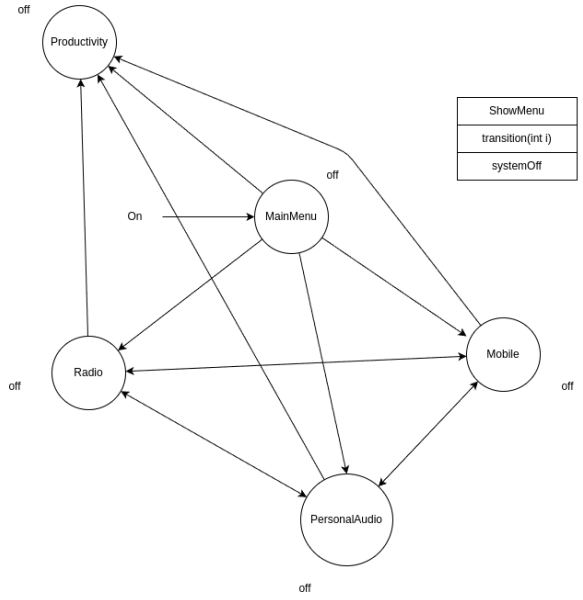
The objective of this laboratory is to construct a sort of car radio with four main interfaces, these are:

1. Radio: is a radio that can change stations.
2. Personal Audio: music that is saved internally in the system.
3. Mobile: to make calls from the user's phone.
4. Productivity: allows the user to start trips.

It is mandatory to implement polymorphism in the program.

To make this program a bit more interesting, I implemented a finite-state machine

Diagram:



So the idea is that the program will transition between 5 different states, one state being the main menu, and the other states being the interfaces mentioned above. Because of

this, it didn't make much sense to have all the system.out.println() just in the App.java, so I do have system.out.println() outside of the main class. Erick did allow for this.

Also because this program simulates a car's radio, the system can't be turned off unless you use CTRL+C, this is because a car's radio can only be fully turned off if the car is turned off.

The program consists of the following classes:

Class App.java

"The main class that runs the application and each state from the finite-state machine"

Attribute:

- running : boolean \ used to stop the while loop.

Methods:

+ main(String[] argos)

—----------Package Estados —----------

Class ManejadorDeEstados:

"ManejadorDeEstados handles the state transitions and the current state of the system."

Attributes:

- currEstado: Estado \ the current state of the machine.
- SystemOn: Boolean \ if false the system is sleeping, if true the system is running.

Methods:

+ ManejardorDeEstados(): constructor
+ showMenu(): String \ Gets the menu options for the current state of the system.
+ getEstado(): Estado \ Retrieves the current state of the system.
+ transition(int action): void \ Handles the state transition based on the given action.
+ systemOff(): void \ enters sleep mode
+ isSystemOn(): boolean \ check if the system is currently running.
+ setSystemOn(boolean systemOn): void \ Sets the system's on/off state.

"SUPER"

Estado "abstract":

" This class represents the state of the system."

Methods:

+ abstract showMenu(): String \ prints the main menu
+ abstract transition(int action): Estado \ switches between the different states.

—----------Package MenuPrincipal ----------

Class MenuPrincipal Extends Estado:

"The MenuPrincipal class represents the main menu of the system."

Methods:

@Override

+ showMenu(): String

@Override

+ transition(int action) : Estado

—---------- Package Radio----------

Class EstadoRadio extends Estado implements IRadio:

"It handles the different stations that the user can listen to."

Attributes:

- Station: float \ The current station of the radio.
- radioData: RadioData \ The data manager for the radio interface.
- stationMap: Map<Float, String> \ The map of available stations.
- favoriteStations: <Map<Float, String> \ The map of favourite stations.

Methods:

+ EstadoRadio: Constructor
+ @Override showMenu(): string
+ @Override transition(int action): Estado
+ @Override cambiarCanalArriba(): String
+ @Override cambiarCanalAbajo(): String
+ @Override cambiarAM(): String
+ @Override cambiarFM(): String
+ @Override elejirFavoritas(): String
+ @Override agregarFavoritas(): String

<Interface> IRadio

"Interface for the EstadoRadio"

Methods:

+ cambiarCanalArriba(): String \ changes the station up
+ cambiarCanalAbajo(): String \ changes the station down
+ cambiarAM(): String \changes to AM
+ cambiarFM(): String \ changes to FM
+ elejirFavoritas(): String \ selects a station that the user likes.
+ agregarFavoritas(): String \ adds the current station to the favourite list from the user.

—---------- Package Productivity----------

Class EstadoProductivity extends Estado implements IProductivity

"It provides options to start a new trip or return to the main menu."

Attributes:

- tripData: TripData \ is an instance of TripData class.
- tripMap: Map<Integer, String> tripMap \ looks for the trip from the tripData.

- currentTrip: String \ is the current trip the user is on.

  Methods:

+ EstadoProdcutivity(): constructor
+ @Override showMenu(): string
+ @Override transition(int action): Estado
+ @Override startTrip(int index): String

<Interface> IProductivity

"Interface for EstadoProductivity"

+ startTrip(int Index) : String \ Starts a trip by selecting the destination based on the provided index

—---------- Package PersonalAudio----------

Class EstadoPersonalAudio extends Radio Implements IPersonalAudio

"It handles the different modes that the user can listen to."

Attributes:

- currentMode: mode \ the user can choose from CD, MP3, or SPOTIFY.
- audioData: PersonalAudioData \ the songs for each mode.
- currentMap: Map<Integer, String> : looks for the mode.
- modeIndices: Map<Integer, String>: looks for the song once the mode is defined.

Methods:

+ EstadoPersonalAudio() Constructor
+ @Override showMenu(): string
+ @Override transition(int action): Estado
+ Interfaces from IpersonalAudio…

<Interface> IPersonalAudio

"The interface for EstadoPersonalAudio

+ typeMode(): String \ switches between the different modes.
+ movoSongUp(): String \ returns the current song and changes the song.
+ moveSongDown(): String \ returns the current song and changes the song.
+ escucharSong(): String \ returns the current song.

—---------- Package Mobile----------

Class EstadoMobile extends Estado Implements IMobile

"It handles the use of the user's phone to make calls."

Attributes:

- audioFormat: String[] \ a list: "Speakers", "Earphones"
- mobileData: MobileData \the contacts from the user's phone.
- contactMap: Map<Integer, String> \ looks for a specific contact from the user.
- isPhoneConnected: boolean \ if the phone is not connected then the interface can't be used.

- currentCallIndex: integer \ the current contact that the user is calling.
- currentAudioFormat: int \ the system has Speakers as default.

Methods:

+ EstadoMobile(): Constructor
+ @Override showMenu(): string
+ @Override transition(int action): Estado
+ Interfaces from IMobile

<Interface> IMobile

+ connectarTelefono(): String \ connects the user's phone returning a string if it achieved the connection.
+ desconectarTelefono(): String \ disconnects the user's phone returning a string if it achieved the disconnection.
+ llamarContact(int index): String \ call a contact from the user's phone, asking the index from the user's phone. Returns a prompt of the contact information.
+ mostrarListaContactos(): String \returns all the contacts from the user's phone.
+ terminarLlamada(): String \ Ends the users call, returning a prompt of the cancellation.
+ cambiarAuriculares: String \ changes the audio format from speakers to earphones and vice versa.

—---------- Package Data----------

All the Data classes use Gson to manage a Json file. They all have a load and save the json file.

Class MobileData:

"This class handles the management of the data for the mobile interface, including the contact list."

Class PersonalAudioData:

"This class handles the management of the data for the personal audio interface, including the song list."

Class RadioData:

"This class handles the management of the data for the radio interface, including the station list and favourite stations."

Class TripData:

"This class handles the management of the data for the trip interface, including the trip list."

—---------- Package JSON----------

CDSONGS.json \these are the songs for the mode CD in interface personal audio.
Favoritreradio.json \ these are the favourite stations from the interface radio.

Mobilecontacts.json \ these are the contacts from the user phone.
MP3SONGS.json \ these are the songs for the mode MP3 in interface personal audio.
SPOTIFYSONGS.json \ these are the songs for the Spotify in interface personal audio.
ttations.json \ these are the stations for the interface radio.
trips.json \ these are the trips the user can choose from in the interface productivity.