Command pattern program

Program description

The program stores a list of media devices (TVs/radios) and allows the user to add new media devices, individually turn them on or off, change their volume, or turn off all the stored media devices via the command design pattern.

Object description

1. Command – interface that defines encapsulated commands in the program

* Execute() – method to used to execute the encapsulated command

1. ElectronicDevice – interface that defines the methods available to electronic devices (TVs/radios) in the program

* On() – method to turn on the device
* Off() – method to turn off the device
* volumeUp() – method to turn the device’s volume up
* volumeDown() – method to turn the device’s volume down
* getStatus() – method that returns the device’s status (on/off) as a Boolean
* getStatusString() – method that returns the device’s overall status (on/off, volume if on) as a string

1. Radio – class supposed to correspond to a radio and implements the ElectronicDevice interface

* Volume – stores the device’s volume as an integer ranging from 0 to 100
* Status – Boolean corresponding to the device’s status (on/off)
* statusString – string describing the device’s overall status (on/off, volume if on)
* on() – updates the class’ *status* and *statusString* variables and returns *statusString*
* off() – updates the class’ *status* and *statusString* variables and returns *statusString*
* volumeUp() – if *volume* is below 100, increases *volume* by 1 and updates and returns *statusString*; otherwise displays a message telling the user *volume* can’t be bigger than 100
* volumeDown() – if *volume* is above 0, decreases *volume* by 1 and updates and returns *statusString*; otherwise displays a message telling the user *volume* can’t be lower than 0
* getStatus() – returns *status*
* getStatusString() – returns *statusString*

1. Television – class supposed to correspond to a TV and implements the ElectronicDevice interface

* Volume – stores the device’s volume as an integer ranging from 0 to 100
* Status – Boolean corresponding to the device’s status (on/off); off by default
* statusString – string describing the device’s overall status (on/off, volume if on)
* on() – updates the class’ *status* and *statusString* variables and returns *statusString*
* off() – updates the class’ *status* and *statusString* variables and returns *statusString*
* volumeUp() – if *volume* is below 100, increases *volume* by 1 and updates and returns *statusString*; otherwise displays a message telling the user *volume* can’t be bigger than 100
* volumeDown() – if *volume* is above 0, decreases *volume* by 1 and updates and returns *statusString*; otherwise displays a message telling the user *volume* can’t be lower than 0
* getStatus() – returns *status*
* getStatusString() – returns *statusString*

1. TurnTVOn – class used to encapsulate the *on()* command of classes that implement the *ElectronicDevice* interface

* theDevice – used to store the *ElectronicDevice* the specific *TurnTVOn* instance refers to
* TurnTVOn(ElectronicDevice newDevice) – class constructor that stores *newDevice* as *theDevice*
* Execute() – method that encapsulates the *on()* method of the *ElectronicDevice* object stored in *theDevice*

1. TurnTVOff – class used to encapsulate the *off()* command of classes that implement the *ElectronicDevice* interface

* theDevice – used to store the *ElectronicDevice* the specific *TurnTVOff* instance refers to
* TurnTVOff(ElectronicDevice newDevice) – class constructor that stores *newDevice* as *theDevice*
* Execute() – method that encapsulates the *off()* method of the *ElectronicDevice* object stored in *theDevice*

1. TurnTVUp – class used to encapsulate the *volumeUp()* command of classes that implement the *ElectronicDevice* interface

* theDevice – used to store the *ElectronicDevice* the specific *TurnTVUp* instance refers to
* TurnTVUp(ElectronicDevice newDevice) – class constructor that stores *newDevice* as *theDevice*
* Execute() – method that encapsulates the *volumeUp()* method of the *ElectronicDevice* object stored in *theDevice*

1. TurnTVDown – class used to encapsulate the *volumeDown()* command of classes that implement the *ElectronicDevice* interface

* theDevice – used to store the *ElectronicDevice* the specific *TurnTVDown* instance refers to
* TurnTVDown(ElectronicDevice newDevice) – class constructor that stores *newDevice* as *theDevice*
* Execute() – method that encapsulates the *volumeDown()* method of the *ElectronicDevice* object stored in *theDevice*

1. TurnAllOff – class used to encapsulate and run the *off()* command on a list class instances that implement the *ElectronicDevice* interface

* theDevices – used to store a list of objects that implement the *ElectronicDevice interface*
* TurnAllOff(List<ElectronicDevice> newDevices) – class constructor that stores the *ElectronicDevice* list *newDevices* as *theDevices*
* Execute() – method that encapsulates the *off()* method of all *ElectronicDevice* objects stored in *theDevices*

1. Form1 – Main class/form of the program, used to store the list of all currently added *ElectronicDevice* objects and execute the commands associated with them

* *currentDevices* – list of *ElectronicDevice* objects used to store the currently added devices
* addTVBtn\_Click() – creates a new *Television* object, adds it to *currentDevices* and displays its status in a list box in the form
* addRadioBtn\_Click() – creates a new *Radio* object, adds it to *currentDevices* and displays its status in a list box in the form
* TurnOnBtn\_Click() – fetches the currently selected *ElectronicDevice* from the list box, encapsulates its *on()* method using the *TurnTVOn* class and executes it, then updates the device’s status in the list box; Displays a message and stops if there is no device selected or the device is already on
* TurnOffBtn\_Click() – fetches the currently selected *ElectronicDevice* from the list box, encapsulates its *off()* method using the *TurnTVOff* class and executes it, then updates the device’s status in the list box; Displays a message and stops if there is no device selected or the device is already off
* VolumeUpBtn\_Click() – fetches the currently selected *ElectronicDevice* from the list box, encapsulates its *volumeUp()* method using the *TurnTVUp* class and executes it, then updates the device’s status in the list box; Displays a message and stops if there is no device selected or the device is off
* VolumeDownBtn\_Click() – fetches the currently selected *ElectronicDevice* from the list box, encapsulates its *volumeDown( )*method using the *TurnTVDown* class and executes it, then updates the device’s status in the list box; Displays a message and stops if there is no device selected or the device is off
* TurnAllOffBtn\_Click() – encapsulates the *off()* method of *currentDevices* using the *TurnAllOff* class and then executes the group call; Updates the entire list box after finishing

3 problems

Reusability – Reusability would prove difficult since the command pattern usually produces a multitude of little classes that handle a singular function

Maintainability – Due to the small and readable nature of the classes handling the encapsulation, maintainability should be easy, provided proper naming is used; If classes are named in a vague and unclear way though, maintainability would be very hard due to how many classes the pattern produces

Extendibility – Adding encapsulation for any other commands added in the future should be easy, considering that the classes handling the actual encapsulation are very small