Software Design Document

for

TalkBox

Group 15

York University

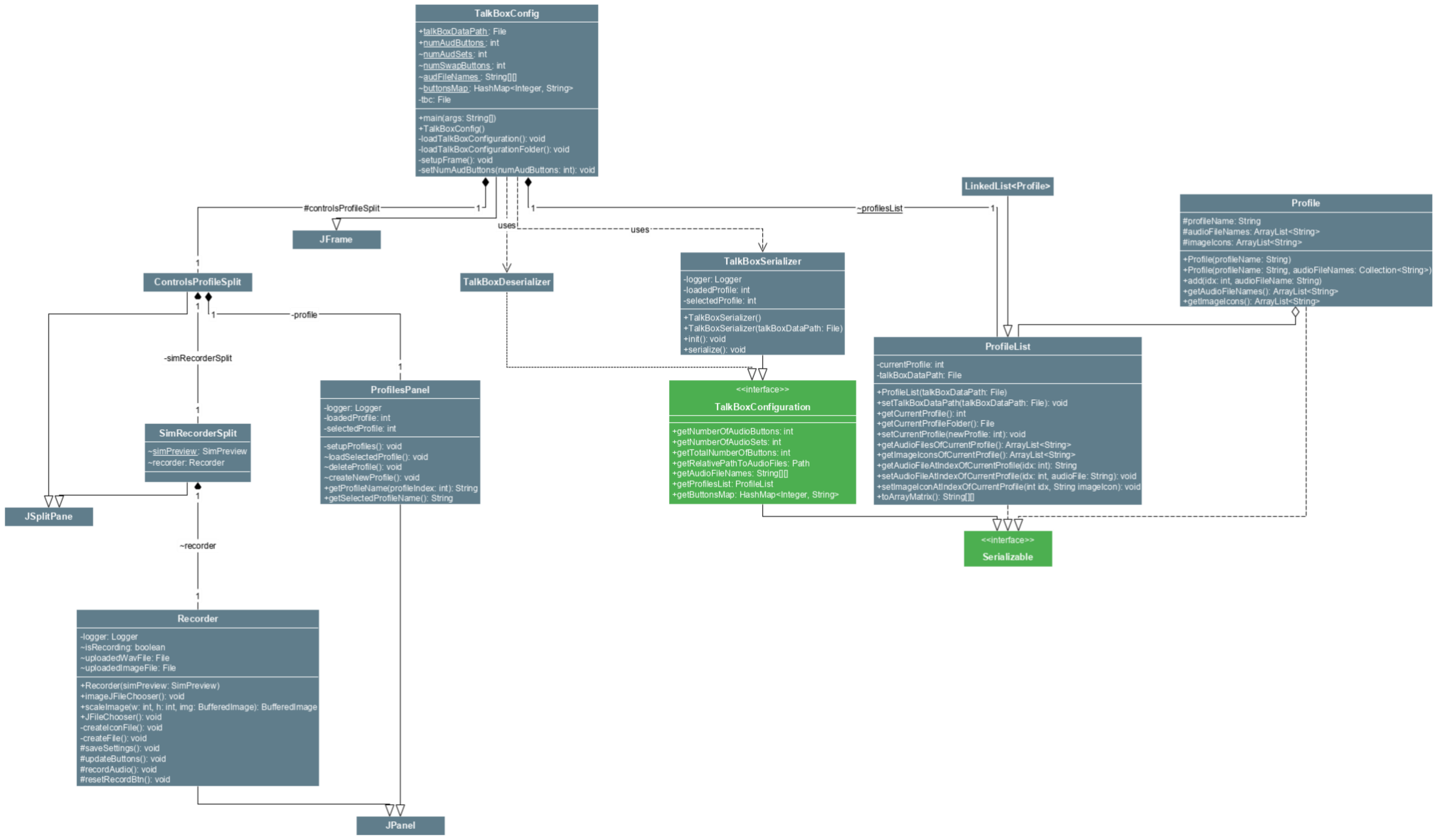
EECS2311 Software Development Project

24. March 2019

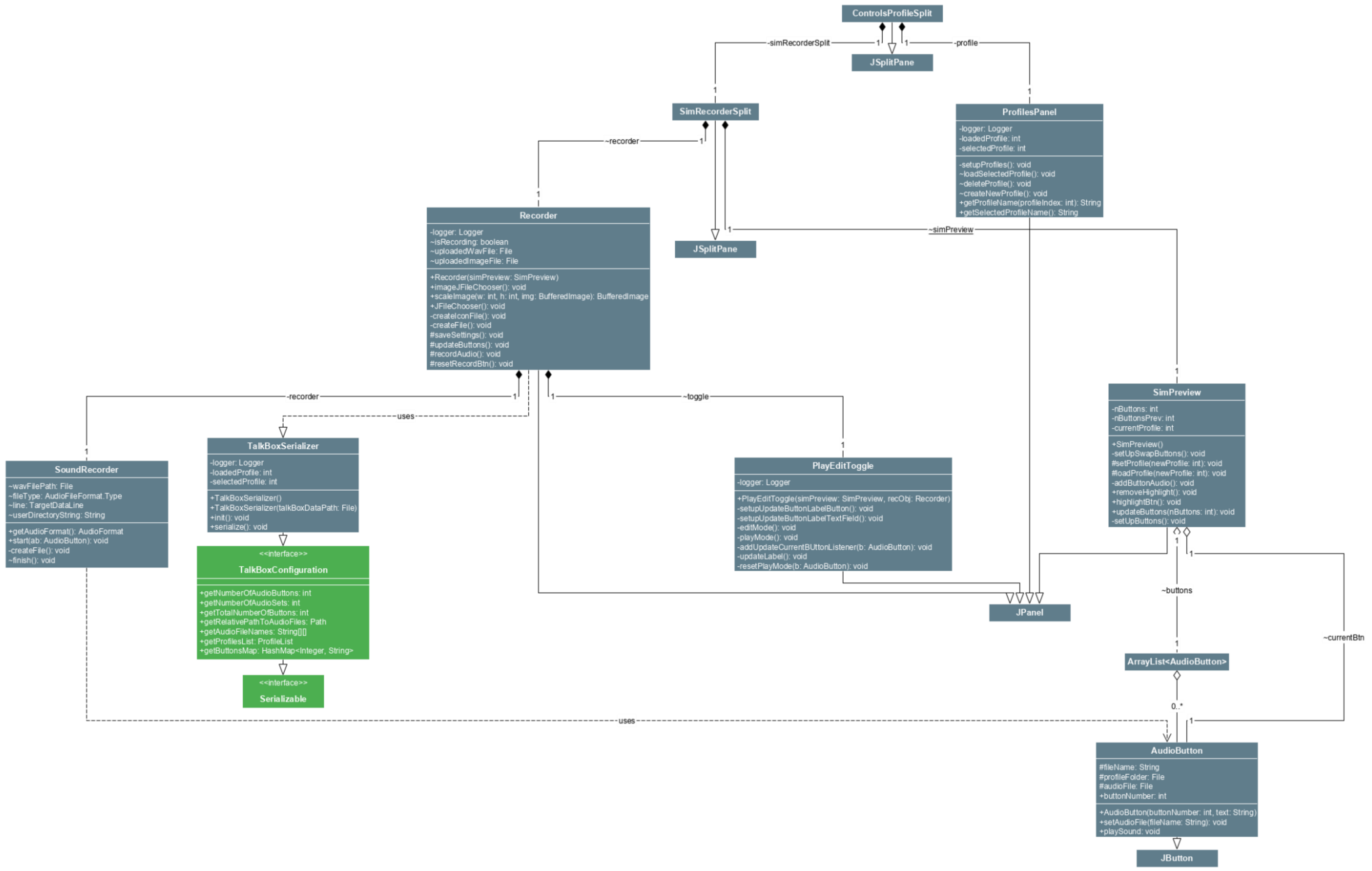
Contents

Class Diagrams

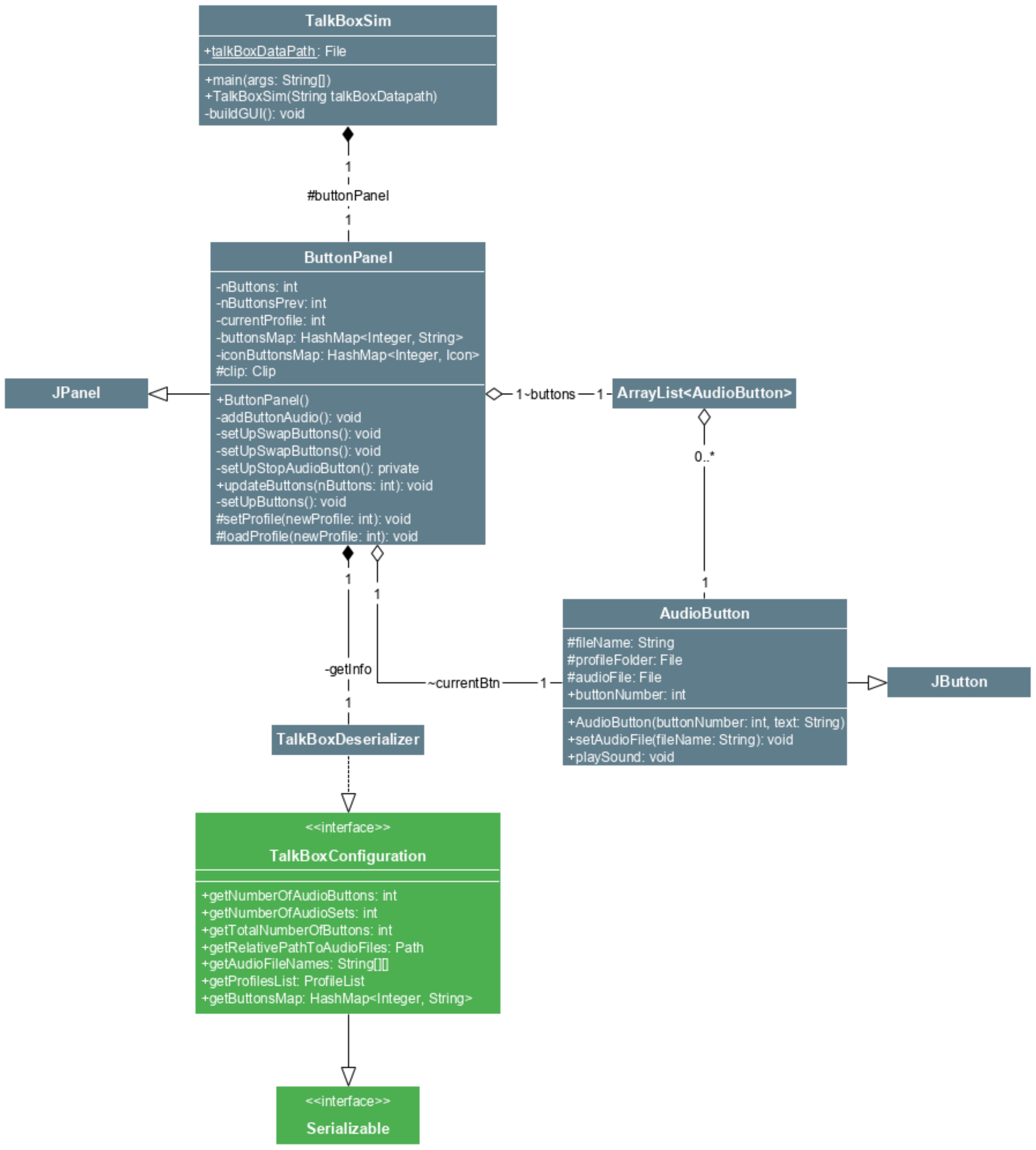
TalkBoxConfig High Level


TalkBoxConfig
+talkBoxDataPath: File
+numAudButtons: int
~numAudSets: int
~numSwapButtons: int
~audFileNames: String[][]
~buttonsMap: HashMap<Integer, String>
-tbc: File
+main(args: String[])
+TalkBoxConfig()
-loadTalkBoxConfiguration(): void
-loadTalkBoxConfigurationFolder(): void
-setupFrame(): void
-setNumAudButtons(numAudButtons: int): void
#controlsProfileSplit
1
1
JFrame
uses
uses
~profilesList
1
1
LinkedList<Profile>
Profile
#profileName: String
#audioFileNames: ArrayList<String>
#imageIcons: ArrayList<String>
+Profile(profileName: String)
+Profile(profileName: String, audioFileNames: Collection<String>)
+add(idx: int, audioFileName: String)
+getAudioFileNames(): ArrayList<String>
+getImageIcons(): ArrayList<String>
ControlsProfileSplit
TalkBoxDeserializer
TalkBoxSerializer
-logger: Logger
-loadedProfile: int
-selectedProfile: int
+TalkBoxSerializer()
+TalkBoxSerializer(talkBoxDataPath: File)
+init(): void
+serialize(): void
1
1
-profile
-simRecorderSplit
1
1
ProfilesPanel
-logger: Logger
-loadedProfile: int
-selectedProfile: int
-setupProfiles(): void
~loadSelectedProfile(): void
~deleteProfile(): void
~createNewProfile(): void
+getProfileName(profileIndex: int): String
+getSelectedProfileName(): String
<<interface>>
TalkBoxConfiguration
+getNumberOfAudioButtons: int
+getNumberOfAudioSets: int
+getTotalNumberOfButtons: int
+getRelativePathToAudioFiles: Path
+getAudioFileNames: String[][]
+getProfilesList: ProfileList
+getButtonsMap: HashMap<Integer, String>
ProfileList
-currentProfile: int
-talkBoxDataPath: File
+ProfileList(talkBoxDataPath: File)
+setTalkBoxDataPath(talkBoxDataPath: File): void
+getCurrentProfile(): int
+getCurrentProfileFolder(): File
+setCurrentProfile(newProfile: int): void
+getAudioFilesOfCurrentProfile(): ArrayList<String>
+getImageIconsOfCurrentProfile(): ArrayList<String>
+getAudioFileAtIndexOfCurrentProfile(idx: int): String
+setAudioFileAtIndexOfCurrentProfile(idx: int, audioFile: String): void
+setImageIconAtIndexOfCurrentProfile(int idx, String imageIcon): void
+toArrayMatrix(): String[][]
SimRecorderSplit
~simPreview: SimPreview
~recorder: Recorder
JSplitPane
1
~recorder
1
<<interface>>
Serializable
Recorder
-logger: Logger
~isRecording: boolean
~uploadedWavFile: File
~uploadedImageFile: File
+Recorder(simPreview: SimPreview)
+imageJFileChooser(): void
+scaleImage(w: int, h: int, img: BufferedImage): BufferedImage
+JFileChooser(): void
-createIconFile(): void
-createFile(): void
#saveSettings(): void
#updateButtons(): void
#recordAudio(): void
#resetRecordBtn(): void
JPanel

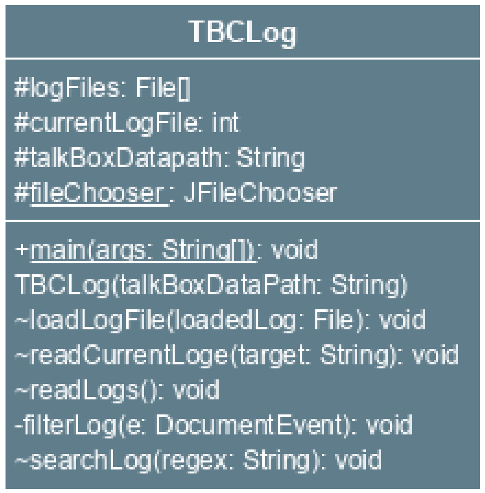
TalkBoxConfig Low Level



TalkBoxSim


TalkBoxSim
+talkBoxDataPath: File
+main(args: String[])
+TalkBoxSim(String talkBoxDatapath)
-buildGUI(): void
1
#buttonPanel
1
ButtonPanel
-nButtons: int
-nButtonsPrev: int
-currentProfile: int
-buttonsMap: HashMap<Integer, String>
-iconButtonsMap: HashMap<Integer, Icon>
#clip: Clip
+ButtonPanel()
-addButtonAudio(): void
-setUpSwapButtons(): void
-setUpSwapButtons(): void
-setUpStopAudioButton(): private
+updateButtons(nButtons: int): void
-setUpButtons(): void
#setProfile(newProfile: int): void
#loadProfile(newProfile: int): void
JPanel
1 ~buttons 1
ArrayList<AudioButton>
0..*
1
1
1
-getInfo
1
~currentBtn
1
AudioButton
#fileName: String
#profileFolder: File
#audioFile: File
+buttonNumber: int
+AudioButton(buttonNumber: int, text: String)
+setAudioFile(fileName: String): void
+playSound: void
JButton
TalkBoxDeserializer
<<interface>>
TalkBoxConfiguration
+getNumberOfAudioButtons: int
+getNumberOfAudioSets: int
+getTotalNumberOfButtons: int
+getRelativePathToAudioFiles: Path
+getAudioFileNames: String[][]
+getProfilesList: ProfileList
+getButtonsMap: HashMap<Integer, String>
<<interface>>
Serializable

TBCLog



TBCLog

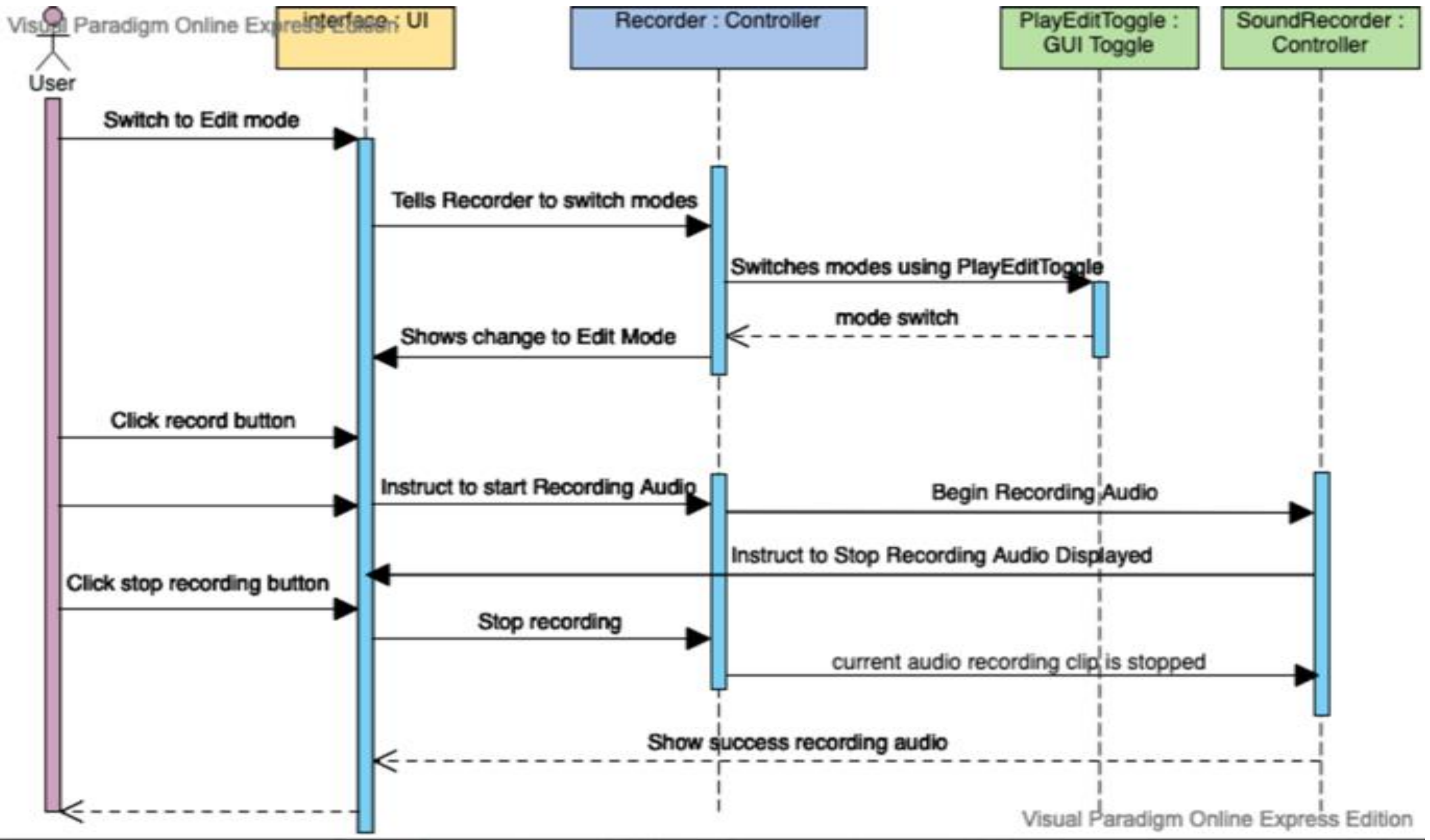
#logFiles: File[]
#currentLogFile: int
#talkBoxDatapath: String
#fileChooser : JFileChooser

+main(args: String[]): void
TBCLog(talkBoxDataPath: String)
~loadLogFile(loadedLog: File): void
~readCurrentLoge(target: String): void
~readLogs(): void
-filterLog(e: DocumentEvent): void
~searchLog(regex: String): void

Sequence Diagrams

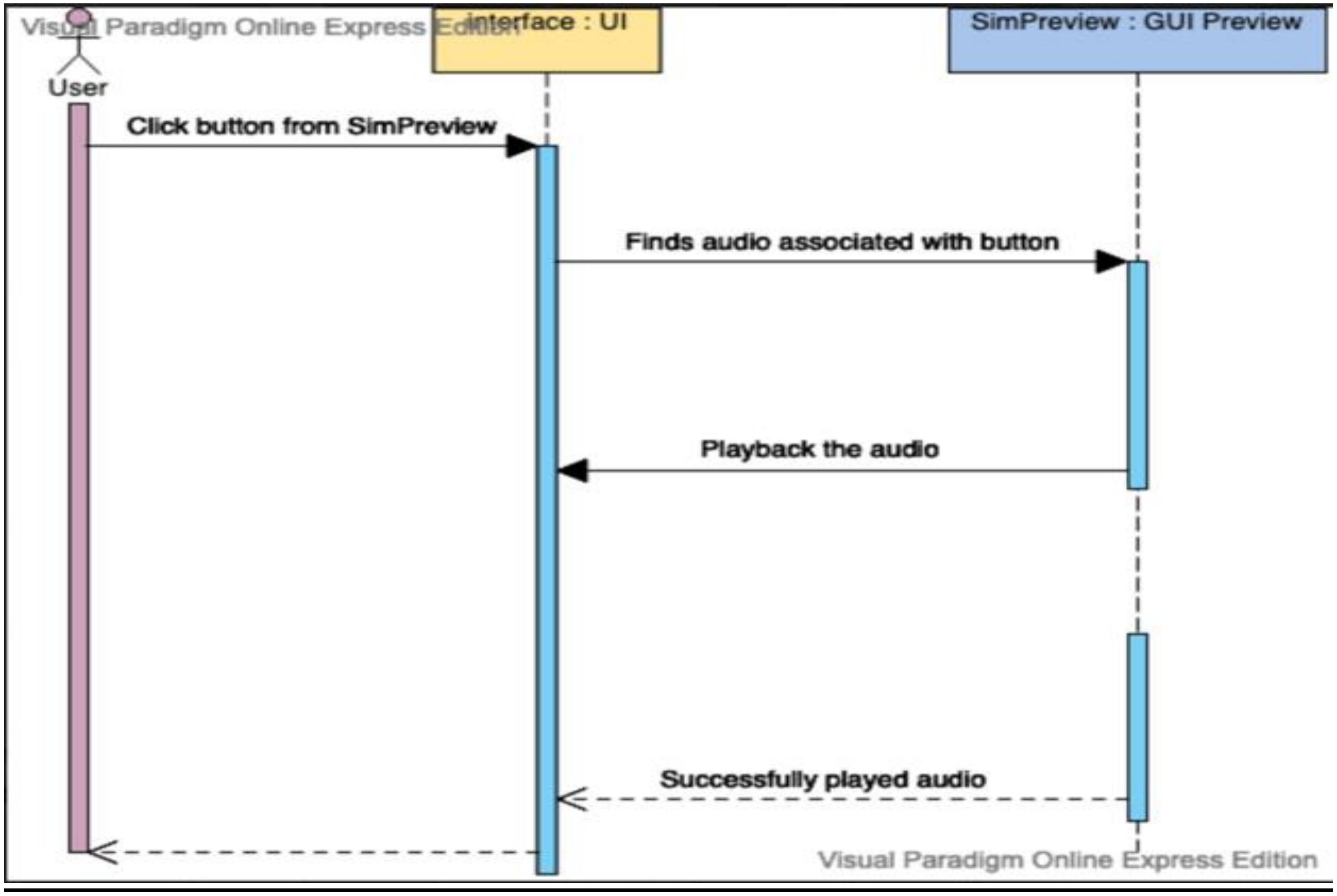
TalkBox Configuration

Record Audio:


Visual Paradigm Online Express Edition
User
interface : UI
Recorder : Controller
PlayEditToggle : GUI Toggle
SoundRecorder : Controller
Switch to Edit mode
Tells Recorder to switch modes
Switches modes using PlayEditToggle
mode switch
Shows change to Edit Mode
Click record button
Instruct to start Recording Audio
Begin Recording Audio
Instruct to Stop Recording Audio Displayed
Click stop recording button
Stop recording
current audio recording clip is stopped
Show success recording audio
Visual Paradigm Online Express Edition

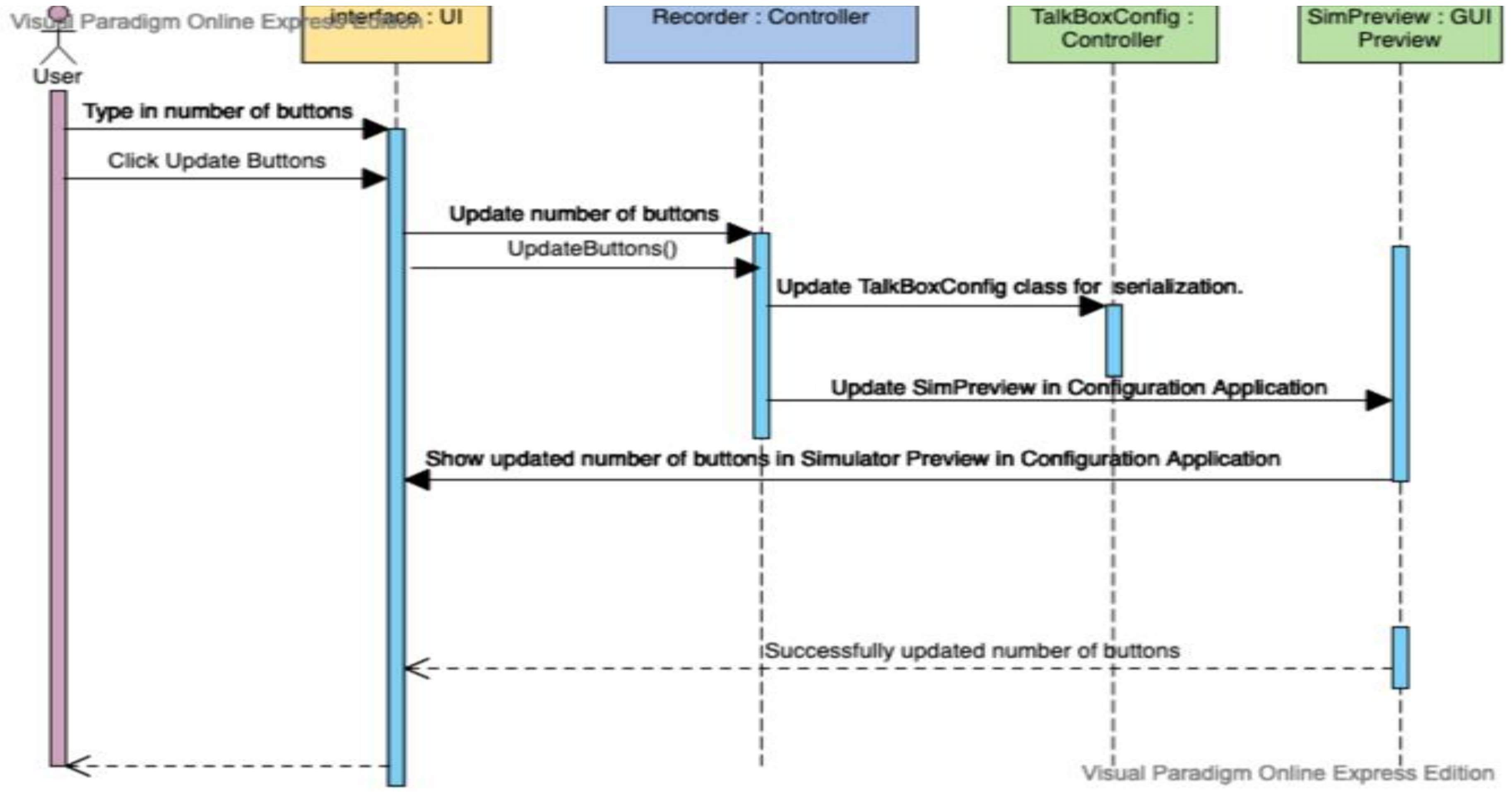
This is the sequence diagram for the record audio functionality of the TalkBox application. This functionality operates as follows: User must switch to edit mode, this tells the recorder class that the mode is being switched and hence a call to the PlayEditToggle class is made, and the mode is changed and shown to user. After this the user clicks the record button and a call to the recorder class is made, which then calls the SoundRecorder class which opens up an input audio stream to record. The user then stops recording and the SoundRecorder is instructed to close the current audio clip input, and recording is stopped. The user sees that the recording has stopped successfully, and that they have recorded audio successfully.

Playback Audio from a Button:


Visual Paradigm Online Express Edition
User
interface : UI
SimPreview : GUI Preview
Click button from SimPreview
Finds audio associated with button
Playback the audio
Successfully played audio
Visual Paradigm Online Express Edition

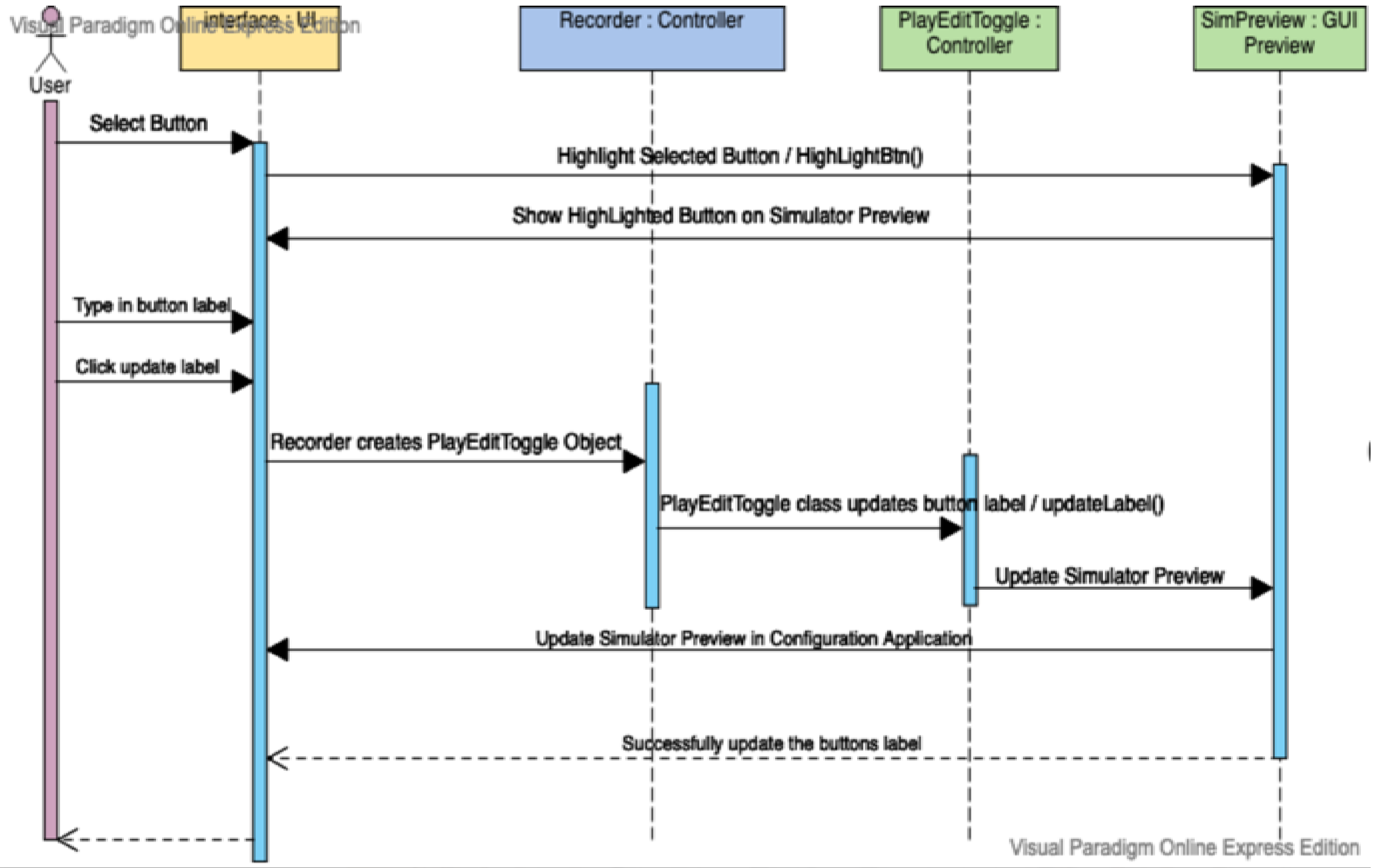
This is the sequence diagram for the user playing back audio from a button in the configuration application. The user first clicks a button that has an associated audio file, after which a call to the SimPreview class is made, which respectively contains the functionality to playback audio associated with a button. The simPreview class is essentially a preview of the simulator inside the configuration application. This completes the playback audio sequence.

Update Number of Buttons:


Visual Paradigm Online Express Edition
User
interface : UI
Recorder : Controller
TalkBoxConfig : Controller
SimPreview : GUI Preview
Type in number of buttons
Click Update Buttons
Update number of buttons
UpdateButtons()
Update TalkBoxConfig class for serialization.
Update SimPreview in Configuration Application
Show updated number of buttons in Simulator Preview in Configuration Application
Successfully updated number of buttons
Visual Paradigm Online Express Edition

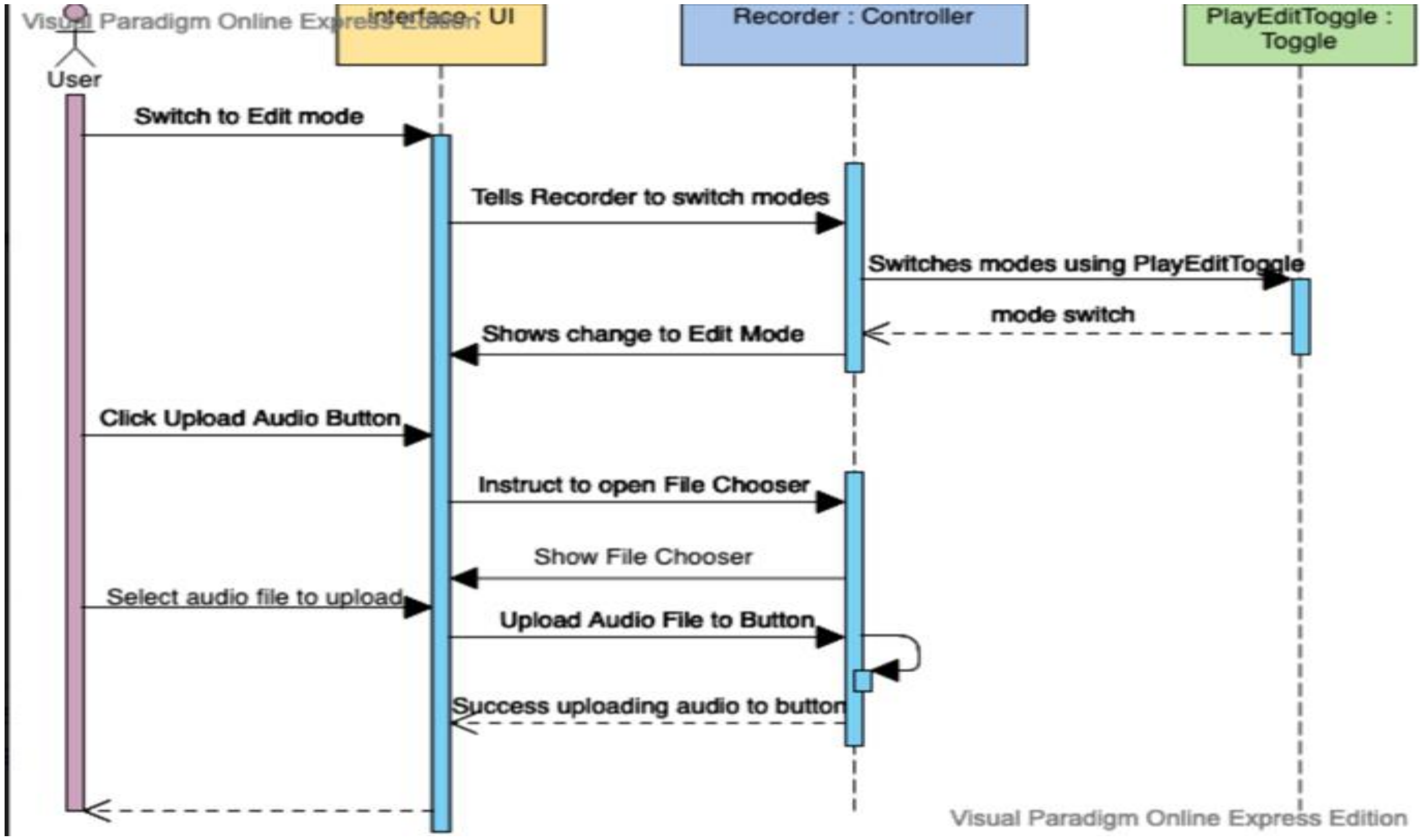
This is the sequence diagram for updating the number of buttons on the simulator through the configuration application. The user first types in a number of buttons and either hits enter on their keyboard, or clicks the update buttons button. This then calls the recorder controller class which updated a field inside the TalkBoxConfig main class that contains the number of buttons for the simulator. This field is used for serialization for the simulator application. Alongside this, the recorder also updated the simPreview class by updating the number of buttons of the simulator preview, and applying this change to the GUI in the configuration application. After the update to SimPreview, the user is shown the updated number of buttons successfully.

Update Button Label:


User
interface : UI
Recorder : Controller
PlayEditToggle : Controller
SimPreview : GUI Preview
Select Button
Highlight Selected Button / HighLightBtn()
Show HighLighted Button on Simulator Preview
Type in button label
Click update label
Recorder creates PlayEditToggle Object
PlayEditToggle class updates button label / updateLabel()
Update Simulator Preview
Update Simulator Preview in Configuration Application
Successfully update the buttons label
Visual Paradigm Online Express Edition

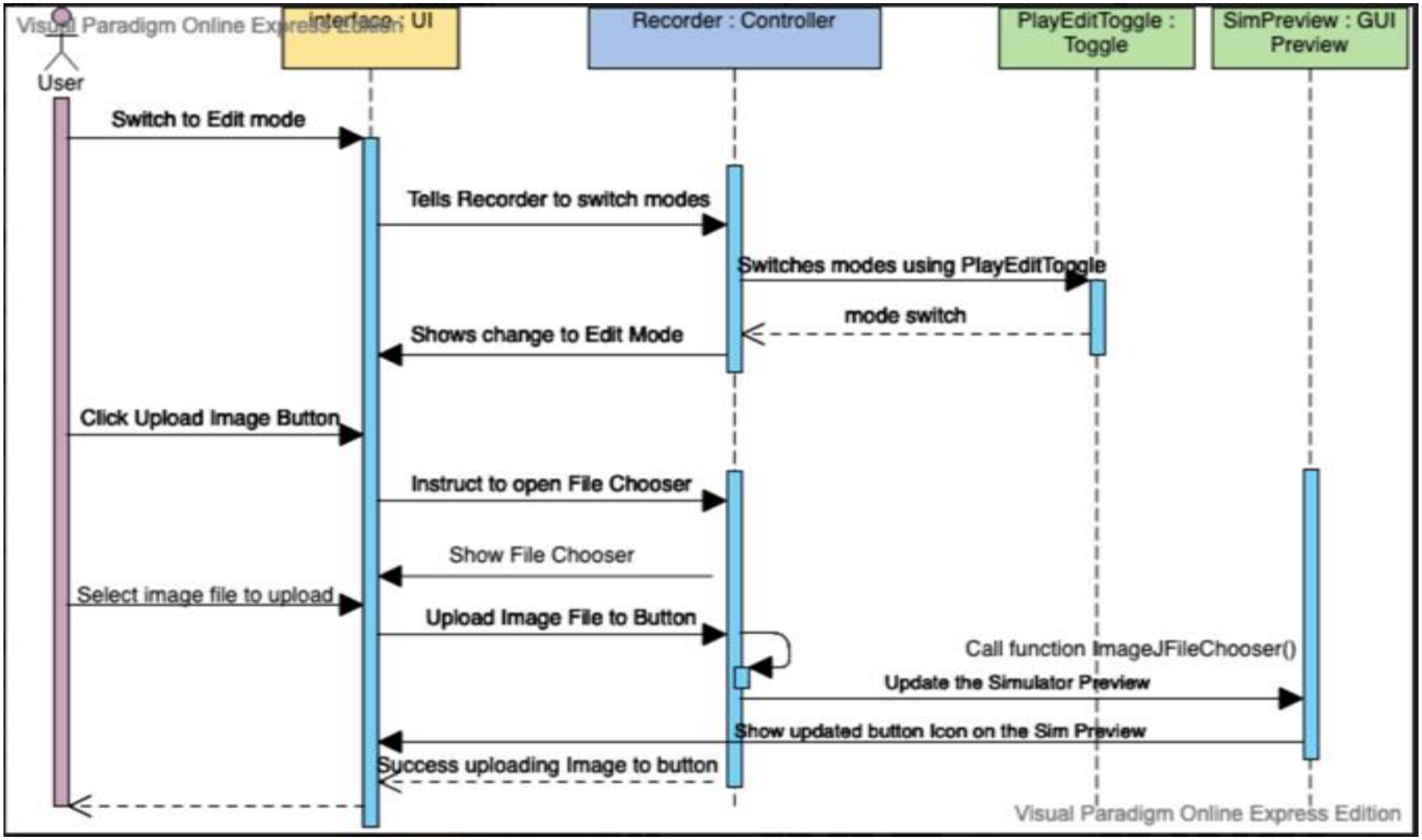
This is the sequence diagram to update the label on each button. The user first selects a button and the button is highlighted using a call to the SimPreview class. The user then types in the label they wish to have on the button, and this instructs the recorder to create a PlayEditToggle object which allows mode switches as well as updating of the button label. After the label is updated within the PlayEditToggle class, an update is done to the SimPreview class to allow the user to see the respective change of the label. The label is then updated and successfully shown to the user.

Upload Audio to Buttons:


Visual Paradigm Online Express Edition
User
interface : UI
Recorder : Controller
PlayEditToggle : Toggle
Switch to Edit mode
Tells Recorder to switch modes
Switches modes using PlayEditToggle
mode switch
Shows change to Edit Mode
Click Upload Audio Button
Instruct to open File Chooser
Show File Chooser
Select audio file to upload
Upload Audio File to Button
Success uploading audio to button
Visual Paradigm Online Express Edition

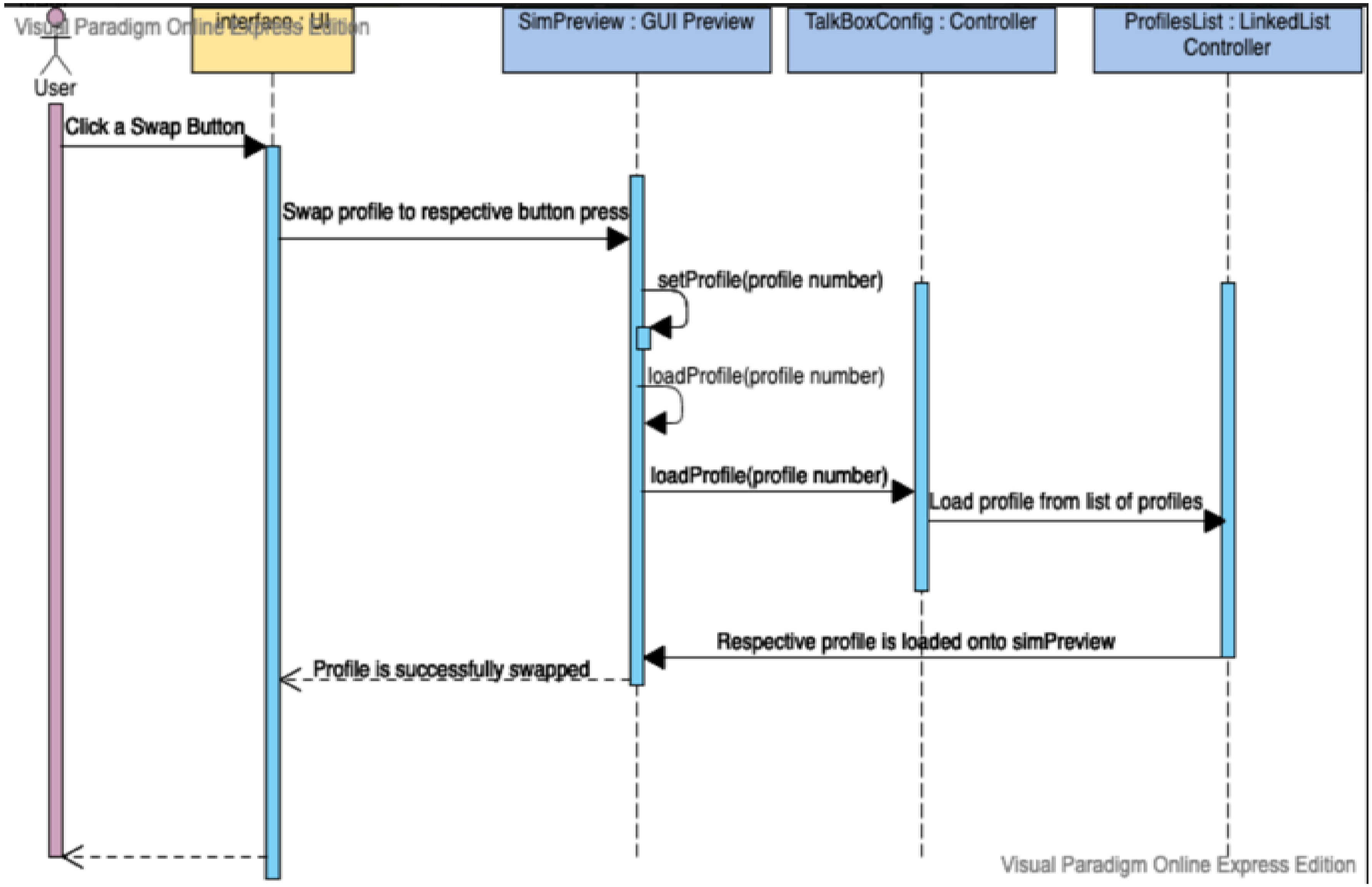
This is a sequence diagram showing the upload audio to a button procedure within the program. User switches to edit mode, this tells the recorder class that the mode is being switched and hence a call to the PlayEditToggle class is made and the mode is changed and shown to user. The user then clicks the update audio button, and a call is made to the recorder class to allow the sequence to progress. User is shown a file chooser and asked to select an audio file (.wav). After the user has selected an audio file, the recorder class within itself, calls method to add the audio to the associated button. Once this is complete the sequence is successful and the audio is uploaded to the button, and can now be played back by the user.

Upload Images to Buttons:


User
interface : UI
Recorder : Controller
PlayEditToggle : Toggle
SimPreview : GUI Preview
Switch to Edit mode
Tells Recorder to switch modes
Switches modes using PlayEditToggle
mode switch
Shows change to Edit Mode
Click Upload Image Button
Instruct to open File Chooser
Show File Chooser
Select image file to upload
Upload Image File to Button
Call function ImageJFileChooser()
Update the Simulator Preview
Show updated button Icon on the Sim Preview
Success uploading Image to button
Visual Paradigm Online Express Edition

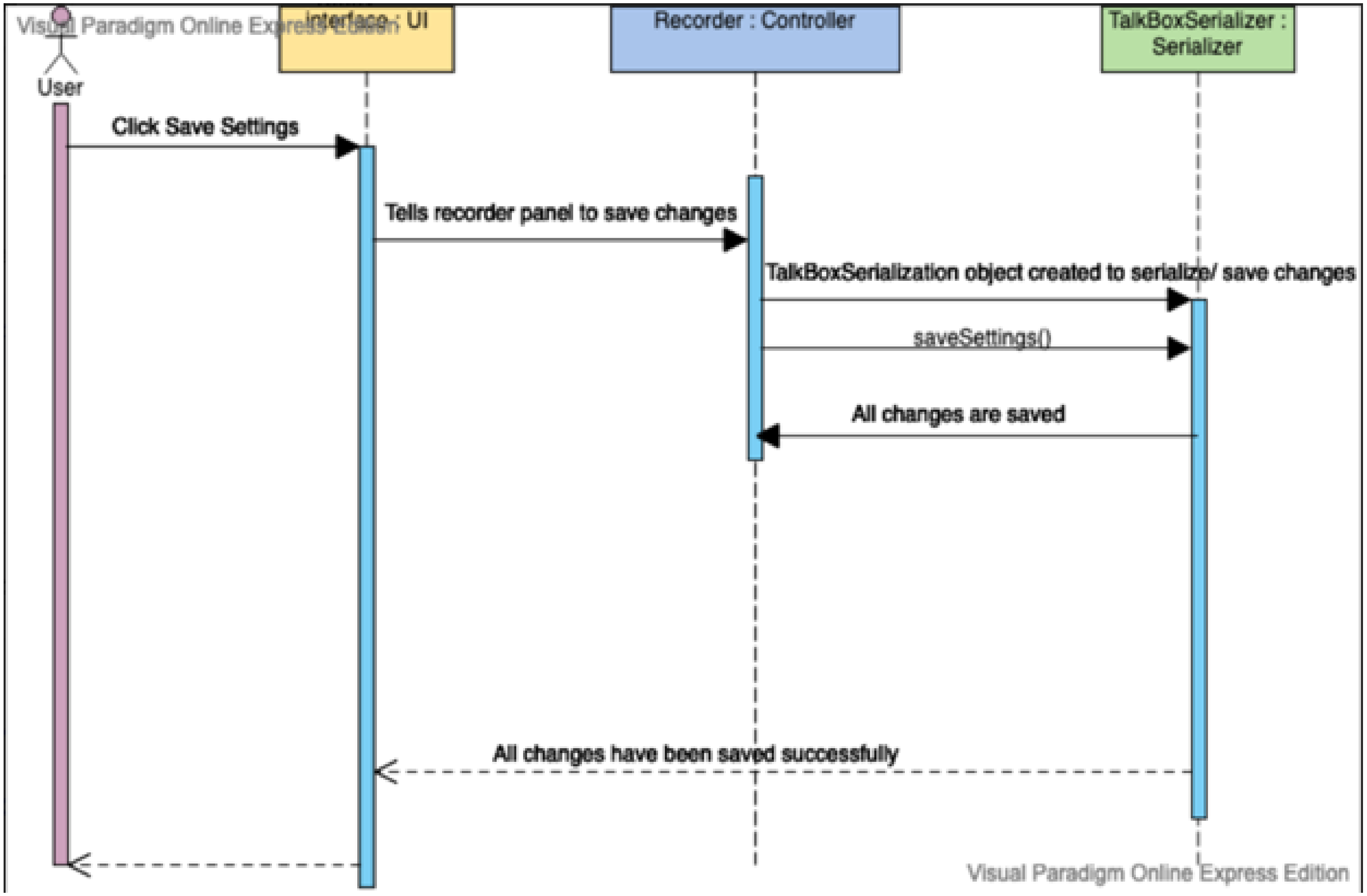
This is a sequence diagram showing the upload image to a button procedure within the program. User switches to edit mode, this tells the recorder class that the mode is being switched and hence a call to the PlayEditToggle class is made and the mode is changed and shown to user. The user then clicks the update image button, and a call is made to the recorder class to allow the sequence to progress. User is shown a file chooser and asked to select an image file. After the user selects a file a call is made within the recorder class to update the buttons associated image file, and then a call is made to SimPreview. This call to SimPreview allows the buttons icon to be updated on the GUI, and shown to the user. The sequence is then complete and an image is uploaded to a button successfully.

Swap Profiles:


User
interface : UI
SimPreview : GUI Preview
TalkBoxConfig : Controller
ProfilesList : LinkedList Controller
Click a Swap Button
Swap profile to respective button press
setProfile(profile number)
loadProfile(profile number)
loadProfile(profile number)
Load profile from list of profiles
Respective profile is loaded onto simPreview
Profile is successfully swapped
Visual Paradigm Online Express Edition

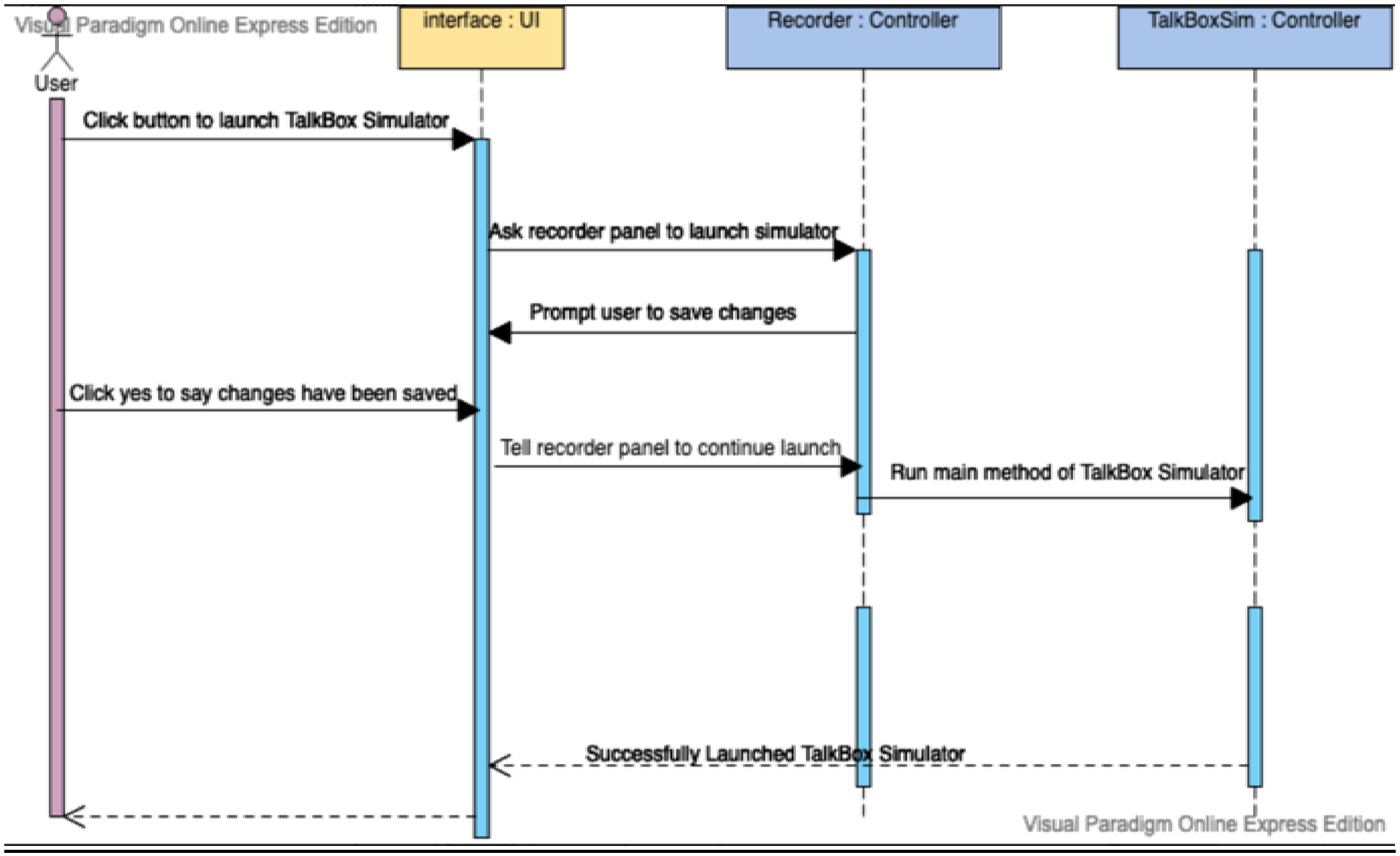
This is the sequence diagram for swapping profiles on the simulator preview inside the configuration application. The user first clicks a swap button and a call is made to SimPreview which respectively updates the simulator preview to have the correctly swapped profile. This is done by calling the TalkBoxConfig class which holds a profile list field. This class finally calls the ProfileList class (which is a list of created profiles). From this class (list), the respective profile is returned back to the simPreview class and the simulator preview in the configuration is updated with the newly swapped profile.

Save Settings / Serialize Changes:


Visual Paradigm Online Express Edition
User
interface : UI
Recorder : Controller
TalkBoxSerializer : Serializer
Click Save Settings
Tells recorder panel to save changes
TalkBoxSerialization object created to serialize/ save changes
saveSettings()
All changes are saved
All changes have been saved successfully
Visual Paradigm Online Express Edition

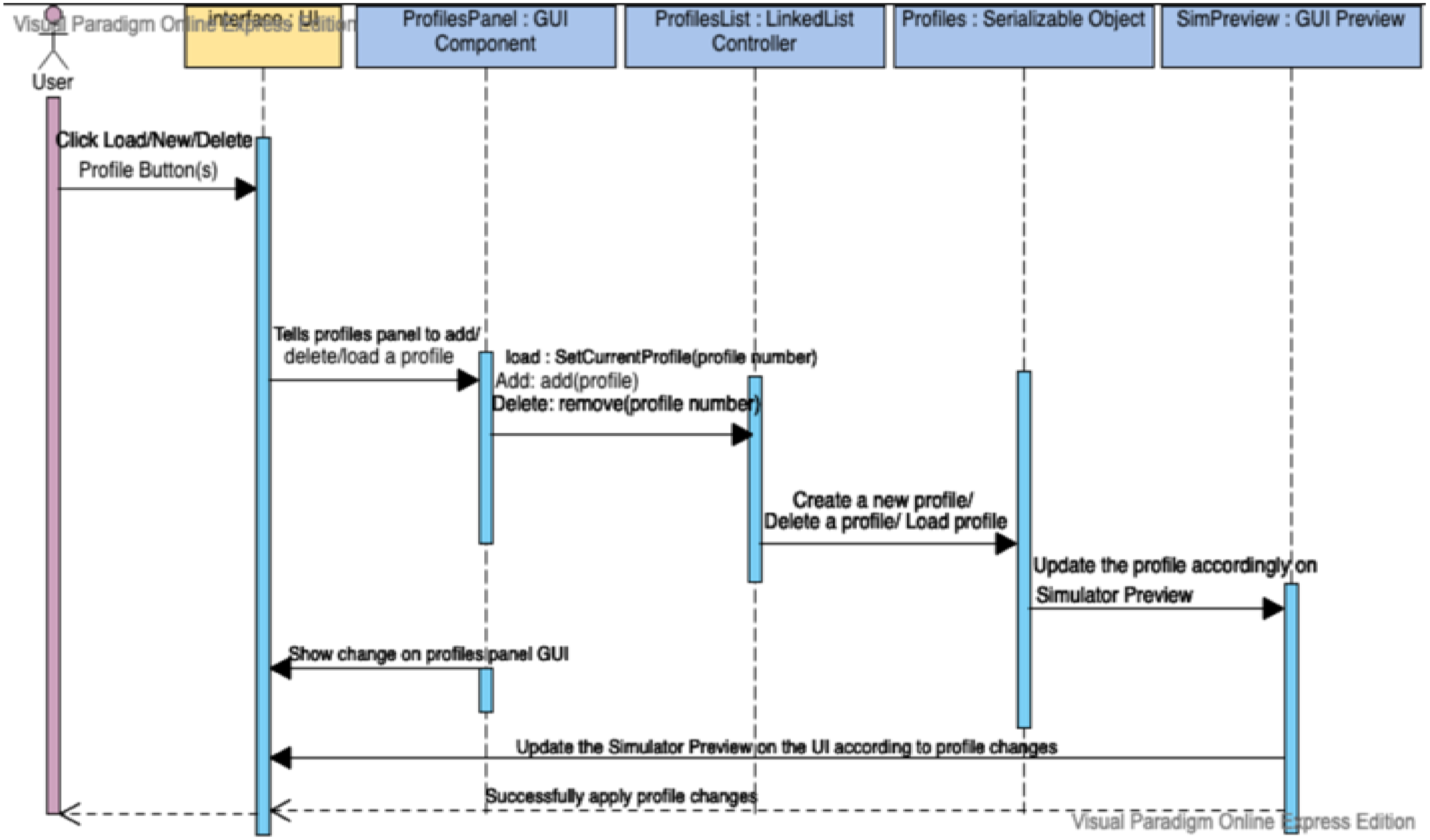
This is a sequence diagram for saving the settings / serializing the changes from the configuration application for other applications such as the simulator application and TalkBox Configuration Log application. The user first clicks save settings button, this button is contained within the recorder panel and thus the Recorder class. The Recorder class then creates a TalkBoxSerialization object to serialize the changes. The saveSettings() method is called within the TalkBoxSerialization class, this method puts all the respective changes into a .tbc file. All changes are then saved successfully and the sequence is complete.

Launch TalkBox Simulator Application:


Visual Paradigm Online Express Edition
User
interface : UI
Recorder : Controller
TalkBoxSim : Controller
Click button to launch TalkBox Simulator
Ask recorder panel to launch simulator
Prompt user to save changes
Click yes to say changes have been saved
Tell recorder panel to continue launch
Run main method of TalkBox Simulator
Successfully Launched TalkBox Simulator
Visual Paradigm Online Express Edition

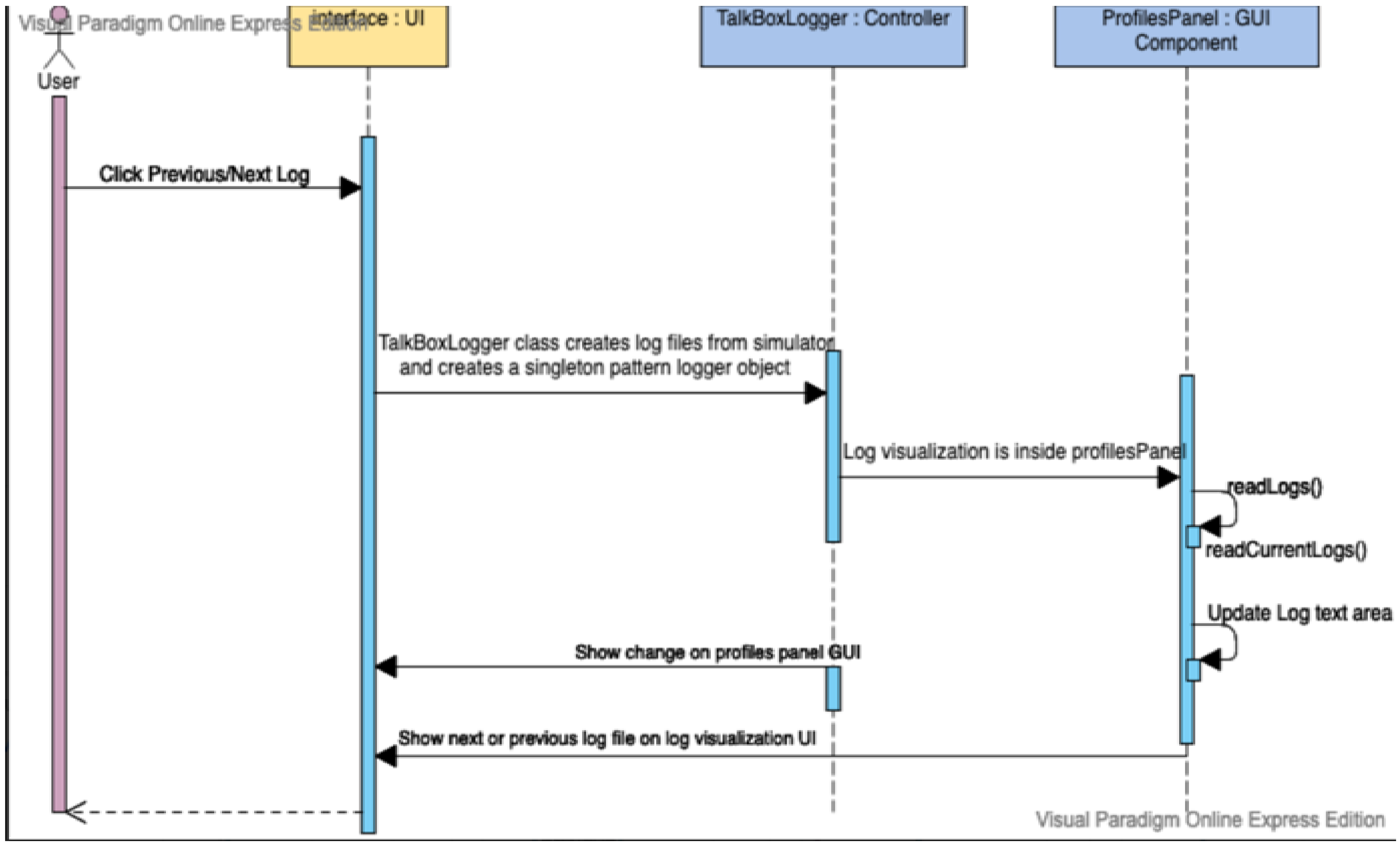
This is the sequence diagram for launching the simulator application from the configuration application. The user first clicks the Launch Simulator button, so a call is made to the Recorder class to tell the class the user has clicked the button. The Recorder class then opens up a window that prompts a user to save changes before launching the simulator application. The user hits yes, they have saved all changes. The Recorder class then makes a call directly to the TalkBoxSim class main method, which is a standalone application. The application is then successfully launched and displayed to the user. The sequence for launching the TalkBox Simulator is hence complete.

Load/Delete/New Profile:


Visual Paradigm Online Express Edition
User
interface : UI
ProfilesPanel : GUI Component
ProfilesList : LinkedList Controller
Profiles : Serializable Object
SimPreview : GUI Preview
Click Load/New/Delete Profile Button(s)
Tells profiles panel to add/ delete/load a profile
load : SetCurrentProfile(profile number)
Add: add(profile)
Delete: remove(profile number)
Create a new profile/ Delete a profile/ Load profile
Update the profile accordingly on Simulator Preview
Show change on profiles panel GUI
Update the Simulator Preview on the UI according to profile changes
Successfully apply profile changes
Visual Paradigm Online Express Edition

This is the sequence diagram for loading/deleting/creating a new profile. The user first hits the respective button (load/new/delete profile). A call is then made to the ProfilesPanel GUI class that passes the respective instruction. This calls the Linked List controller class called ProfilesList, which contains the methods setCurrentProfile(profile number) (for loading), add(profile) (adds a profile), remove(profile number) (deletes a profile). This class calls a class known as profiles, which is a serializable class that represent an individual profile. This class adds, removes, or loads a profile respectively. After it has done this, it will pass the respective parameters and update values to the SimPreview class which will update the profile on the simulator preview. After this is done the UI is updated with the respective profile changes, whether it be loading, creating, or deleting a profile. The sequence for loading, creating, or deleting a profile is hence complete and successful.

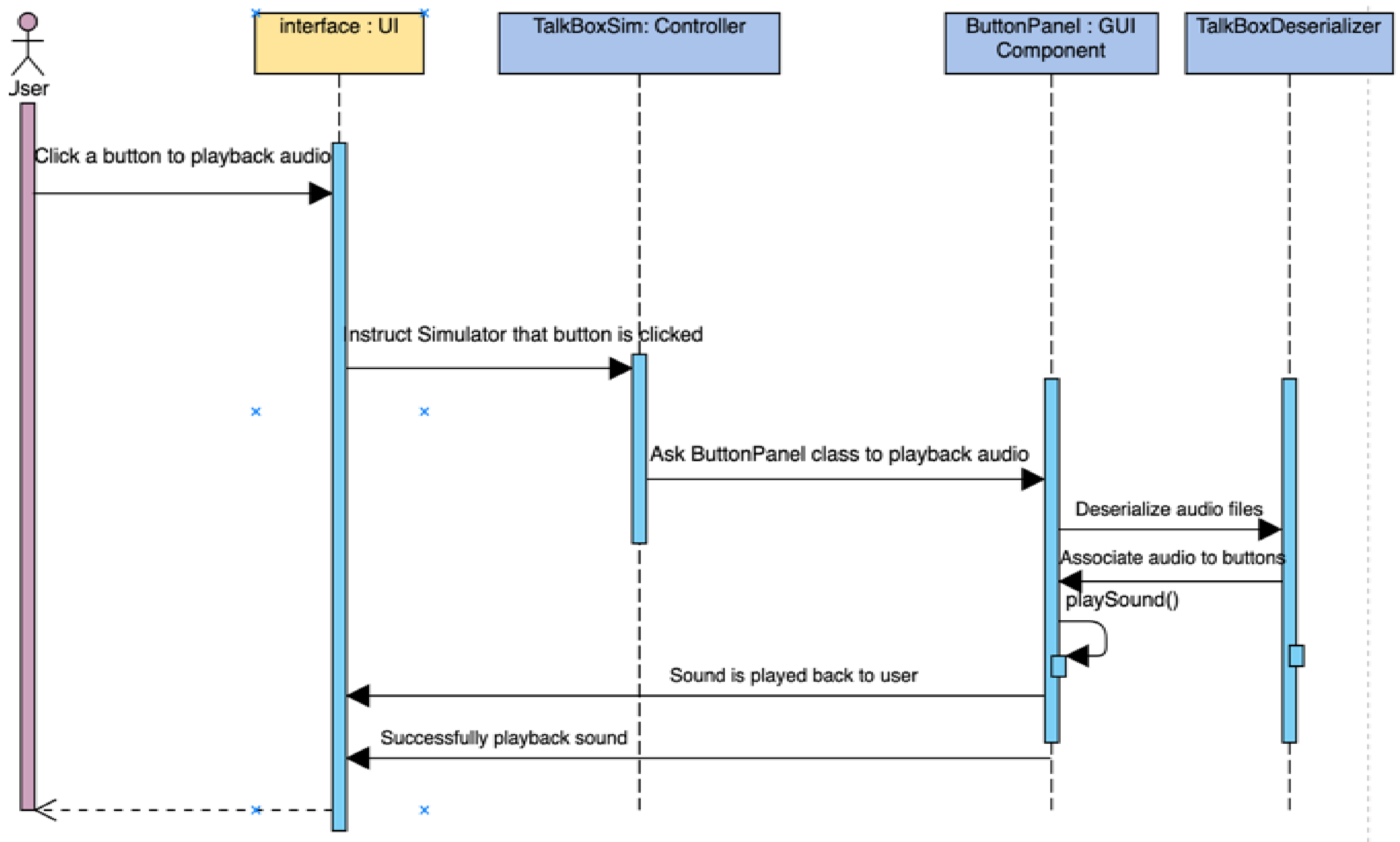
Next/Previous Log:


Visual Paradigm Online Express Edition
User
interface : UI
TalkBoxLogger : Controller
ProfilesPanel : GUI Component
Click Previous/Next Log
TalkBoxLogger class creates log files from simulator and creates a singleton pattern logger object
Log visualization is inside profilesPanel
readLogs()
readCurrentLogs()
Update Log text area
Show change on profiles panel GUI
Show next or previous log file on log visualization UI
Visual Paradigm Online Express Edition

This is the sequence diagram for loading the next or previous simulator log onto the log preview inside the configuration application. The user first clicks Previous/Next Log after which a call is made to the TalkBoxLogger controller class. This class creates the log files from the simulator using a singleton design pattern global logger object. The log visualizations are then read onto the ProfilesPanel class GUI component, which contains the logging visualization. This visualization is updated from the files being read inside the TalkBoxLogger controller class. Within the ProfilesPanel class, readLogs(), readCurrentLogs(), and updateLogs() methods update the log to match the users action of next or previous log. After this the GUI is fully updated and the sequence for next or previous log is completed successfully.

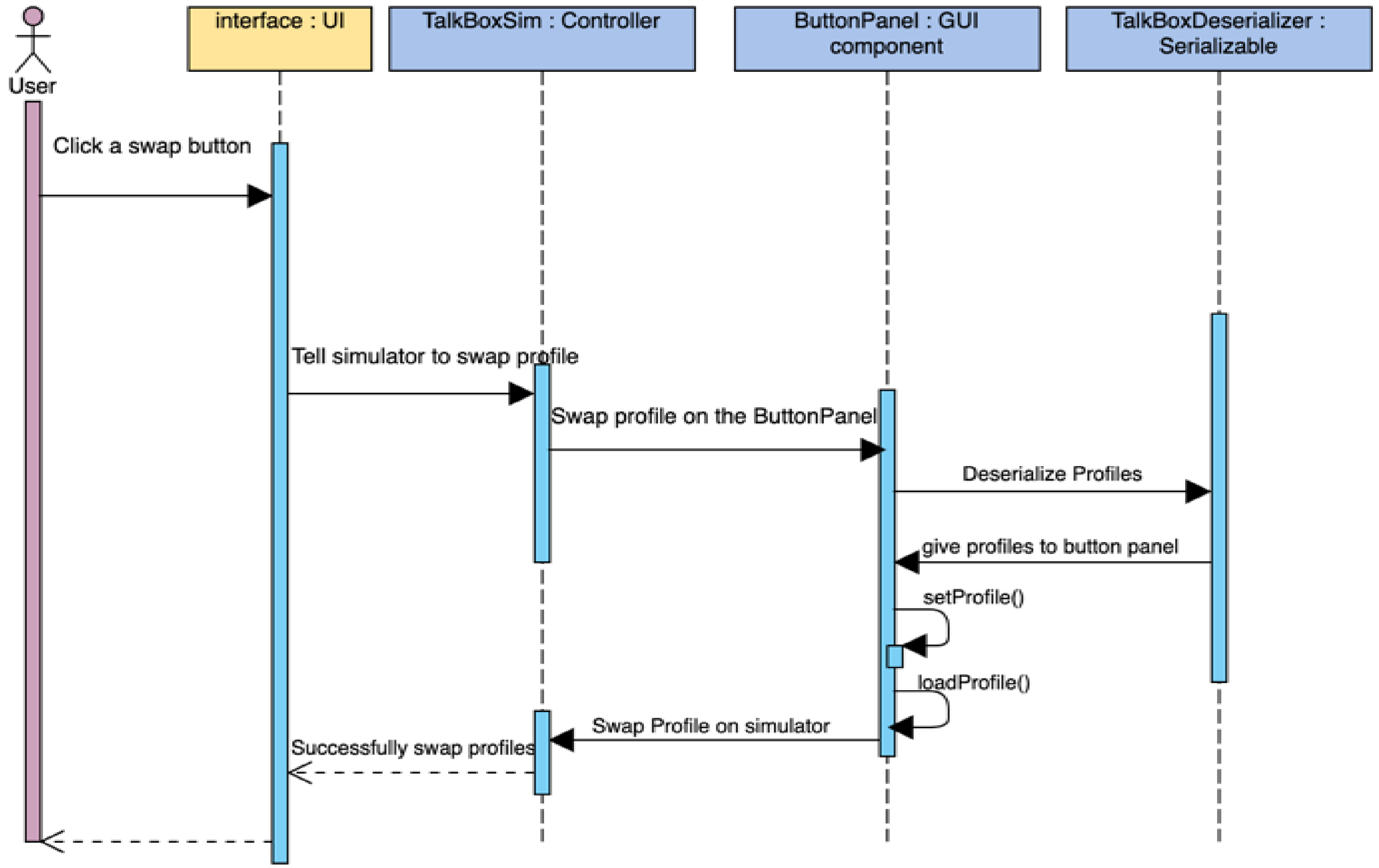
TalkBox Simulator

Playback Audio from a Button:


Jser
interface : UI
TalkBoxSim: Controller
ButtonPanel : GUI Component
TalkBoxDeserializer
Click a button to playback audio
Instruct Simulator that button is clicked
Ask ButtonPanel class to playback audio
Deserialize audio files
Associate audio to buttons
playSound()
Sound is played back to user
Successfully playback sound

This is the sequence diagram for playing audio from a button inside the TalkBox Simulator application. The user first clicks a button to playback audio from, this calls the TalkBoxSim controller class and lets it know the button was clicked, this class then invokes the ButtonPanel class to playback audio from the button. The ButtonPanel class first calls the TalkBoxDeserializer class to deserialize all the changes from the configuration application. After this deserialization, the audio is associated with each button respectively. Then the users initial command continues its sequence, and the playSound() method is called inside the ButtonPanel on the button that has been clicked. This then successfully plays back the audio to the user, and the sequence is thus complete.

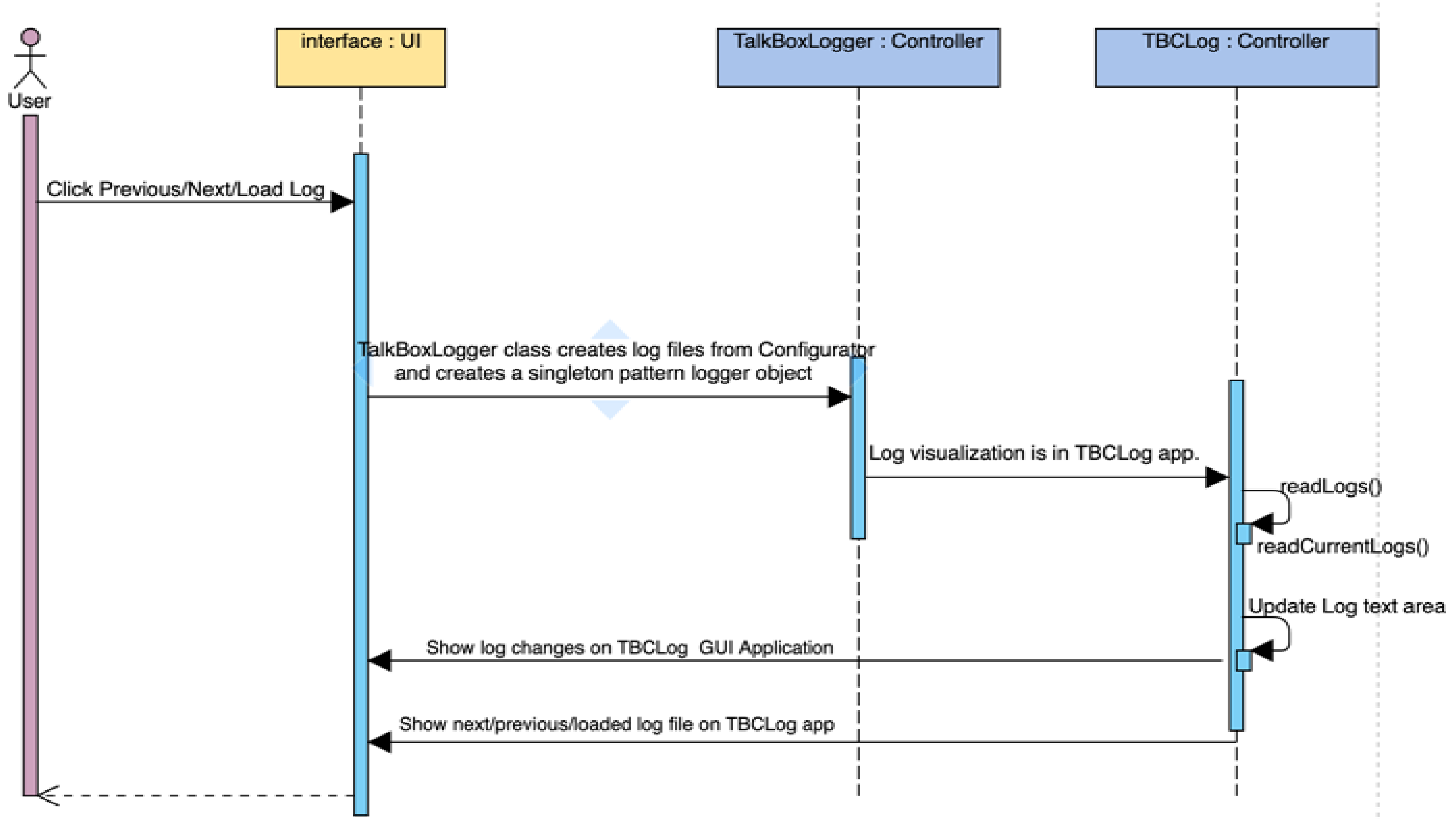
Swap Profiles:


User
interface : UI
TalkBoxSim : Controller
ButtonPanel : GUI component
TalkBoxDeserializer : Serializable
Click a swap button
Tell simulator to swap profile
Swap profile on the ButtonPanel
Deserialize Profiles
give profiles to button panel
setProfile()
loadProfile()
Swap Profile on simulator
Successfully swap profiles

This is a sequence diagram to swap profiles within the TalkBox Simulator application. The user first clicks a swap button (any of the 4 swap buttons), and a call is made to the TalkBoxSim controller class. This class then calls the ButtonPanel class, which calls the TalkBoxDeserializer class to deserialize the changes made in the configuration application. The deserialization is done, and the ButtonPanel within itself calls the methods setProfile(), and loadProfile(), which set and load a profile onto the simulator. This update is then done to the TalkBoxSim class as well as the GUI, and the user has successfully swapped a profile(s).

TalkBox Configuration Log (TBCLog)

Next/Previous/Load Log:


User
interface : UI
TalkBoxLogger : Controller
TBCLog : Controller
Click Previous/Next/Load Log
TalkBoxLogger class creates log files from Configurator and creates a singleton pattern logger object
Log visualization is in TBCLog app.
readLogs()
readCurrentLogs()
Update Log text area
Show log changes on TBCLog GUI Application
Show next/previous/loaded log file on TBCLog app

This is a sequence diagram for Next/Previous/Loading a log in the TalkBox Configuration Log application. The user first clicks Previous/Next Log after which a call is made to the TalkBoxLogger controller class. This class creates the log files from the configuration using a singleton design pattern global logger object. This class then invokes the TBCLog controller class which reads the logs that were created by the TalkBoxLogger from the TalkBox Configuration application. The TBCLog class then calls the methods: readLogs(), readCurrentLogs(), and updates the logging text area with the respective logs, whether it be the previous log, the current log, or the next log. It updates the GUI by reading from the respective created log files, and the users command of Previous/Next/Load Log is successfully completed and shown on the GUI component. This ends the logging sequence.