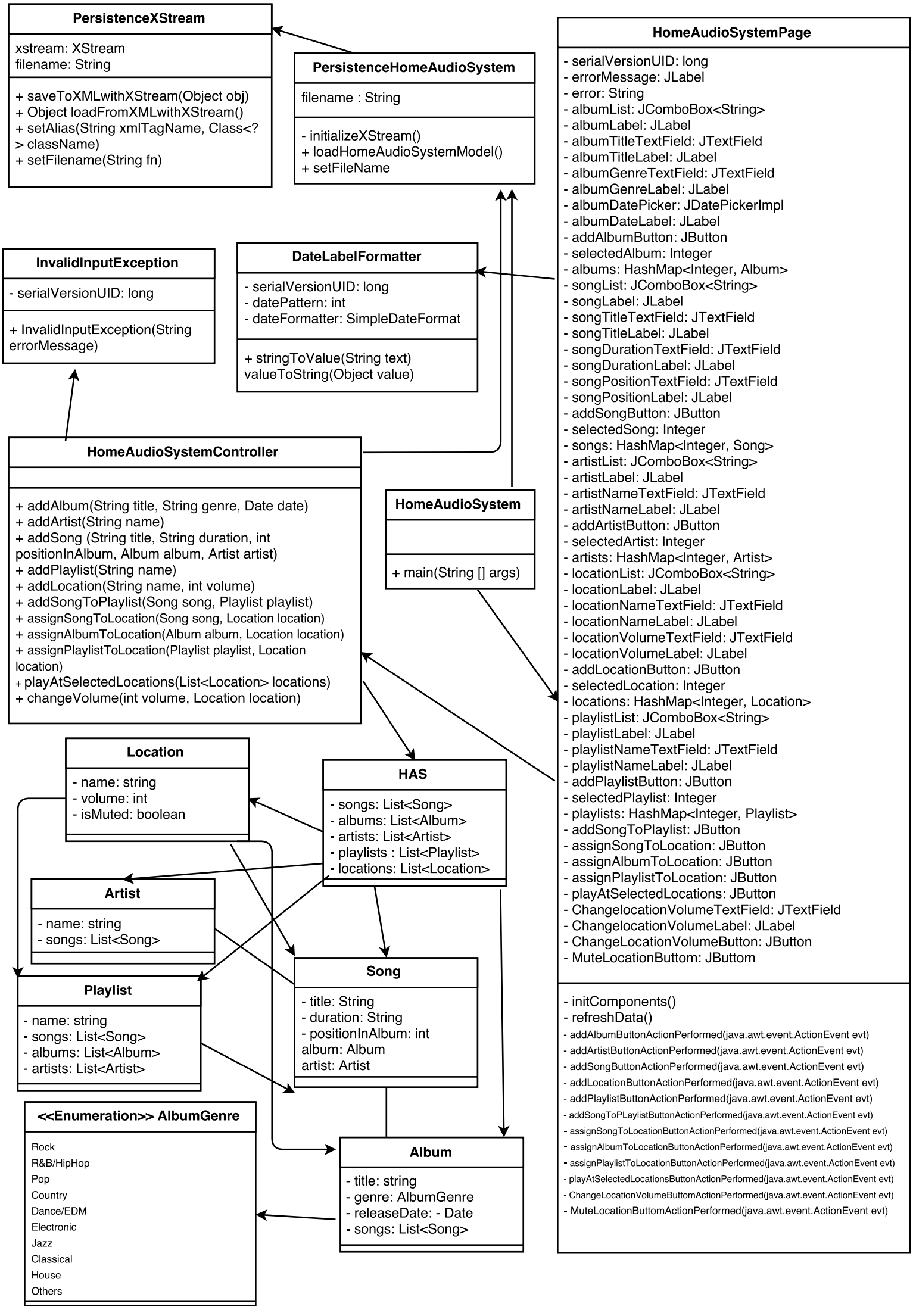
PersistenceXStream
xstream: XStream
filename: String
+ saveToXMLwithXStream(Object obj)
+ Object loadFromXMLwithXStream()
+ setAlias(String xmlTagName, Class<?
> className)
+ setFilename(String fn)
PersistenceHomeAudioSystem
filename : String
- initializeXStream()
+ loadHomeAudioSystemModel()
+ setFileName
HomeAudioSystemPage
- serialVersionUID: long
- errorMessage: JLabel
- error: String
- albumList: JComboBox<String>
- albumLabel: JLabel
- albumTitleTextField: JTextField
- albumTitleLabel: JLabel
- albumGenreTextField: JTextField
- albumGenreLabel: JLabel
- albumDatePicker: JDatePickerImpl
- albumDateLabel: JLabel
- addAlbumButton: JButton
- selectedAlbum: Integer
- albums: HashMap<Integer, Album>
- songList: JComboBox<String>
- songLabel: JLabel
- songTitleTextField: JTextField
- songTitleLabel: JLabel
- songDurationTextField: JTextField
- songDurationLabel: JLabel
- songPositionTextField: JTextField
- songPositionLabel: JLabel
- addSongButton: JButton
- selectedSong: Integer
- songs: HashMap<Integer, Song>
- artistList: JComboBox<String>
- artistLabel: JLabel
- artistNameTextField: JTextField
- artistNameLabel: JLabel
- addArtistButton: JButton
- selectedArtist: Integer
- artists: HashMap<Integer, Artist>
- locationList: JComboBox<String>
- locationLabel: JLabel
- locationNameTextField: JTextField
- locationNameLabel: JLabel
- locationVolumeTextField: JTextField
- locationVolumeLabel: JLabel
- addLocationButton: JButton
- selectedLocation: Integer
- locations: HashMap<Integer, Location>
- playlistList: JComboBox<String>
- playlistLabel: JLabel
- playlistNameTextField: JTextField
- playlistNameLabel: JLabel
- addPlaylistButton: JButton
- selectedPlaylist: Integer
- playlists: HashMap<Integer, Playlist>
- addSongToPlaylist: JButton
- assignSongToLocation: JButton
- assignAlbumToLocation: JButton
- assignPlaylistToLocation: JButton
- playAtSelectedLocations: JButton
- ChangelocationVolumeTextField: JTextField
- ChangelocationVolumeLabel: JLabel
- ChangeLocationVolumeButton: JButton
- MuteLocationButtom: JButtom
- initComponents()
- refreshData()
- addAlbumButtonActionPerformed(java.awt.event.ActionEvent evt)
- addArtistButtonActionPerformed(java.awt.event.ActionEvent evt)
- addSongButtonActionPerformed(java.awt.event.ActionEvent evt)
- addLocationButtonActionPerformed(java.awt.event.ActionEvent evt)
- addPlaylistButtonActionPerformed(java.awt.event.ActionEvent evt)
- addSongToPLaylistButtonActionPerformed(java.awt.event.ActionEvent evt)
- assignSongToLocationButtonActionPerformed(java.awt.event.ActionEvent evt)
- assignAlbumToLocationButtonActionPerformed(java.awt.event.ActionEvent evt)
- assignPlaylistToLocationButtonActionPerformed(java.awt.event.ActionEvent evt)
- playAtSelectedLocationsButtonActionPerformed(java.awt.event.ActionEvent evt)
- ChangeLocationVolumeButtomActionPerformed(java.awt.event.ActionEvent evt)
- MuteLocationButtomActionPerformed(java.awt.event.ActionEvent evt)
InvalidInputException
- serialVersionUID: long
+ InvalidInputException(String
errorMessage)
DateLabelFormatter
- serialVersionUID: long
- datePattern: int
- dateFormatter: SimpleDateFormat
+ stringToValue(String text)
valueToString(Object value)
HomeAudioSystemController
+ addAlbum(String title, String genre, Date date)
+ addArtist(String name)
+ addSong (String title, String duration, int
positionInAlbum, Album album, Artist artist)
+ addPlaylist(String name)
+ addLocation(String name, int volume)
+ addSongToPlaylist(Song song, Playlist playlist)
+ assignSongToLocation(Song song, Location location)
+ assignAlbumToLocation(Album album, Location location)
+ assignPlaylistToLocation(Playlist playlist, Location
location)
+ playAtSelectedLocations(List<Location> locations)
+ changeVolume(int volume, Location location)
HomeAudioSystem
+ main(String [] args)
Location
- name: string
- volume: int
- isMuted: boolean
HAS
- songs: List<Song>
- albums: List<Album>
- artists: List<Artist>
- playlists : List<Playlist>
- locations: List<Location>
Artist
- name: string
- songs: List<Song>
Song
- title: String
- duration: String
- positionInAlbum: int
album: Album
artist: Artist
Playlist
- name: string
- songs: List<Song>
- albums: List<Album>
- artists: List<Artist>
<<Enumeration>> AlbumGenre
Rock
R&B/HipHop
Pop
Country
Dance/EDM
Electronic
Jazz
Classical
House
Others
Album
- title: string
- genre: AlbumGenre
- releaseDate: - Date
- songs: List<Song>

Class Diagram Description

Home Audio System (HAS) is composed of several classes as shown in the class diagram.

HomeAudioSystem

The class contains the main method which loads the model and starts the user interface of the Home Audio System.

Method Index

- Main(String [] args)

HomeAudioSystemPage

HomAudioSystemPage class is responsible for creating the user interface of Home Audio System. The UI created by this class will be used to interact with the user and constantly update when user uses controller to manipulate the model.

Method Index

- addAlbumButtonActionPerformed(java.awt.event.ActionEvent evt)

  Add an album when the button in the interface is pressed
- addSongButtonActionPerformed(java.awt.event.ActionEvent evt)

  Add a song when the button in the interface is pressed
- addArtistButtonActionPerformed(java.awt.event.ActionEvent evt)

  Add an artist when the button in the interface is pressed
- addLocationButtonActionPerformed(java.awt.event.ActionEvent evt)

  Add a location when the button in the interface is pressed
- addPlaylistButtonActionPerformed(java.awt.event.ActionEvent evt)

  Add playlist when the button in the interface is pressed
- addSongToPlaylistButtonActionPerformed(java.awt.event.ActionEvent evt)

  Add a song to the playlist when the button in the interface is pressed
- assignSongToLocationButtonActionPerformed(java.awt.event.ActionEvent evt)

  Assign a song to the location when the button in the interface is pressed.
- assignAlbumToLocationButtonActionPerformed(java.awt.event.ActionEvent evt)

  Assign an album to the location when the button in the interface is pressed.
- assignPlaylistToLocationButtonActionPerformed(java.awt.event.ActionEvent evt)

Assign a playlist to the location when the button in the interface is pressed.

- playAtSelectedLocationsButtonActionPerformed(java.awt.event.ActionEvent evt)

  Play the songs at the multiple locations when the button in the interface is pressed. The user can also choose to play at one location.

- ChangeLocationVolumeButtonActionPerformed(java.awt.event.ActionEvent evt)

  Adjust the volume at certain location when the button in the interface is pressed.

- MuteLocationVolumeButtonActionPerformed(java.awt.event.ActionEvent evt)

  Change the volume to zero at certain location when the button in the interface is pressed.

HomeAudioSystemController

The controller of the HAS has several responsibilities. The methods in this class are responsible for adding album, artist, song, playlist, and location and updates the library of HAS. It will display error message when the input is empty. Also it allows to assign song, album and playlist to a location where user desires, and play at the selected location with adjustable volume.

Method Index

- addAlbum(String title, String genre, Date date)

  Add album entered by user to the library

- addArtist(String name)

  Add artist name entered by user to the library

- addSong(String title, String duration, int positionInAlbum, Album album, Artist artist)

  Add song entered by user to the library

- addPlaylist(String name)

  Add playlist entered by user to the library

- addLocation(String name, int volume)

  Add location entered by user to the library

- addSongToPlaylist(Song song, Playlist playlist)

  Add a song to playlist

- assignSongToLocation(Song song, Location location)

  Assign a song to a specific location

- assignAlbumToLocation(Album album, Location location)

  Assign an album to a specific location

- assignPlaylistToLocation(Playlist playlist, Location location)

Assign a playlist to a specific location

- playAtSelectedLocations(List<Location> locations)

  Play song, album, or playlist that were assigned to the selected locations

- changeVolume(int volume, Location location)

  Change volume at selected location

InvalidInputException

InvalidInputException class allows controller to display error message at certain conditions specified by the controller.

Method Index

- InvalidInputException(String errorMessage)

  Constructs a new exception with the specified error message.

PersistenceHomeAudioSystem

PersistenceHomeAudioSystem class manages the HAS library. It efficiently stores and retrieves data from the library.

Method Index

- initializeXStream()

  Set class with xml tag name for storing data

- loadHomeAudioSystemModel()

  Load the instance of HAS

- setFileName(String name)

  Set name of the file where data is stored

PersistenceXStream

PersistenceXStream class uses XStream library to write and read from an XML file.

Method Index

- saveToXMLwithXStream(Object obj)

  Save data to xml file

- loadFromXMLwithXStream()

  Load xml file where data is stored

- setAlias(String xmlTagName, Class<?> className)

  Set xml tag name for classes when storing data

- setFilename(String fn)

  Set file name with given string

DateLabelFormatter

DateLabelFormatter class is responsible for the format of the date display

Method Index

- stringToValue(String text)

  Parses text from the beginning of the given string to produce an object

- valueToString(Object value))

  Formats a Date into a date/time string

The following classes are the domain models which identify the principal concerns in the system. They are defined using UML class diagrams that include objects, attributes and association and then automatically generated by Umple.

HAS

The HAS class manages how the information is stored for Location, Artist, Song, and Playlist within the library. The HAS was designed using the singleton pattern, such that library which contains all information about the Locations, Artists, Songs, and Playlists will be restricted to instantiate just once. The singleton pattern is most appropriate for this application as it allows for only one library to be created, insuring a singular database for all information to be stored. Therefore, restricting the potential for loss of information through multiple libraries.

Song

The Song class stores information on title, duration, position in an album, album name, and artist name for each song in the HAS library.

Artist

The Artist class stores the name of the artist as well as a list of every song that the artist is associated with.

Playlist

The Playlist class stores the name of the playlist, and lists containing the songs, artists, and albums on the playlist.

Album

The Album class stores the name of the album, the album release date, album genre, and a list containing all songs on the album.