CS2043 Project - Design Documentation

Alec Sobeck
Kieran Lea

Thursday, June 5, 2014

1. Introduction

1.1 – Purpose

The purpose of this design document is to expand upon the requirements document and provide a full and proper design of the billboard system.

1.2 – Scope

This document contains all the required details to complete the implementation phase of the CS2043 project. A full UML class diagram and sequence flow diagrams are provided for every use case, alongside descriptions and relevant UI screenshots.

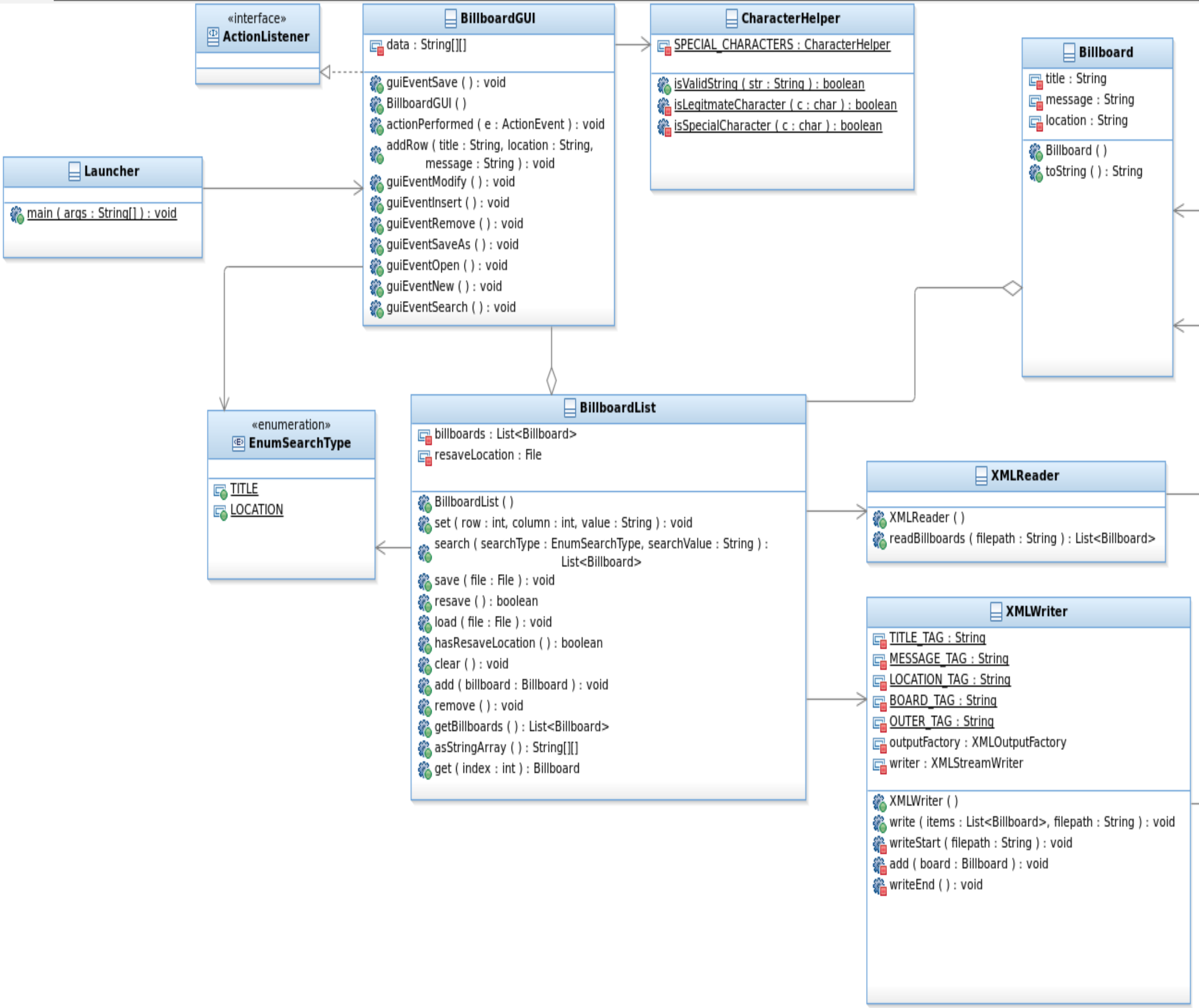
1.3 - Context

This design document is based entirely on the *CS2043 Requirements Document – Team 2.* No additional requirements or functionality is introduced; only existing details in the requirements document are expanded upon.

2. Body

2.1 - Design Module Viewpoint

2.1.1 – System Class View


«interface»
ActionListener
BillboardGUI
data : String[][]
guiEventSave ( ) : void
BillboardGUI ( )
actionPerformed (e : ActionEvent) : void
addRow (title : String, location : String, message : String) : void
guiEventModify ( ) : void
guiEventInsert ( ) : void
guiEventRemove ( ) : void
guiEventSaveAs ( ) : void
guiEventOpen ( ) : void
guiEventNew ( ) : void
guiEventSearch ( ) : void
CharacterHelper
SPECIAL_CHARACTERS : CharacterHelper
isValidString (str : String) : boolean
isLegitmateCharacter (c : char) : boolean
isSpecialCharacter (c : char) : boolean
Billboard
title : String
message : String
location : String
Billboard ( )
toString ( ) : String
Launcher
main (args : String[]) : void
«enumeration»
EnumSearchType
TITLE
LOCATION
BillboardList
billboards : List<Billboard>
resaveLocation : File
BillboardList ( )
set (row : int, column : int, value : String) : void
search (searchType : EnumSearchType, searchValue : String) : List<Billboard>
save (file : File) : void
resave ( ) : boolean
load (file : File) : void
hasResaveLocation ( ) : boolean
clear ( ) : void
add (billboard : Billboard) : void
remove ( ) : void
getBillboards ( ) : List<Billboard>
asStringArray ( ) : String[][]
get (index : int) : Billboard
XMLReader
XMLReader ( )
readBillboards (filepath : String) : List<Billboard>
XMLWriter
TITLE_TAG : String
MESSAGE_TAG : String
LOCATION_TAG : String
BOARD_TAG : String
OUTER_TAG : String
outputFactory : XMLOutputFactory
writer : XMLStreamWriter
XMLWriter ( )
write (items : List<Billboard>, filepath : String) : void
writeStart (filepath : String) : void
add (board : Billboard) : void
writeEnd ( ) : void

[1] See footnote one for BillboardGUI attributes.

2.1.2 – System description

The system design is quite simple. The backend is composed of the Billboard, BillboardList, XMLReader, and XMLWriter classes. The frontend is handled in a single class which extends JFrame in order to display a window. In addition, the frontend makes use of a class called CharacterHelper to verify that input doesn't contain illegal characters.

The front and back ends interact through the BillboardList class. Further, all operations specified in the requirements document can be performed on the BillboardList class. The GUI creates an instance of this BillboardList class and maintains it, calling the appropriate methods in response to user generated events.

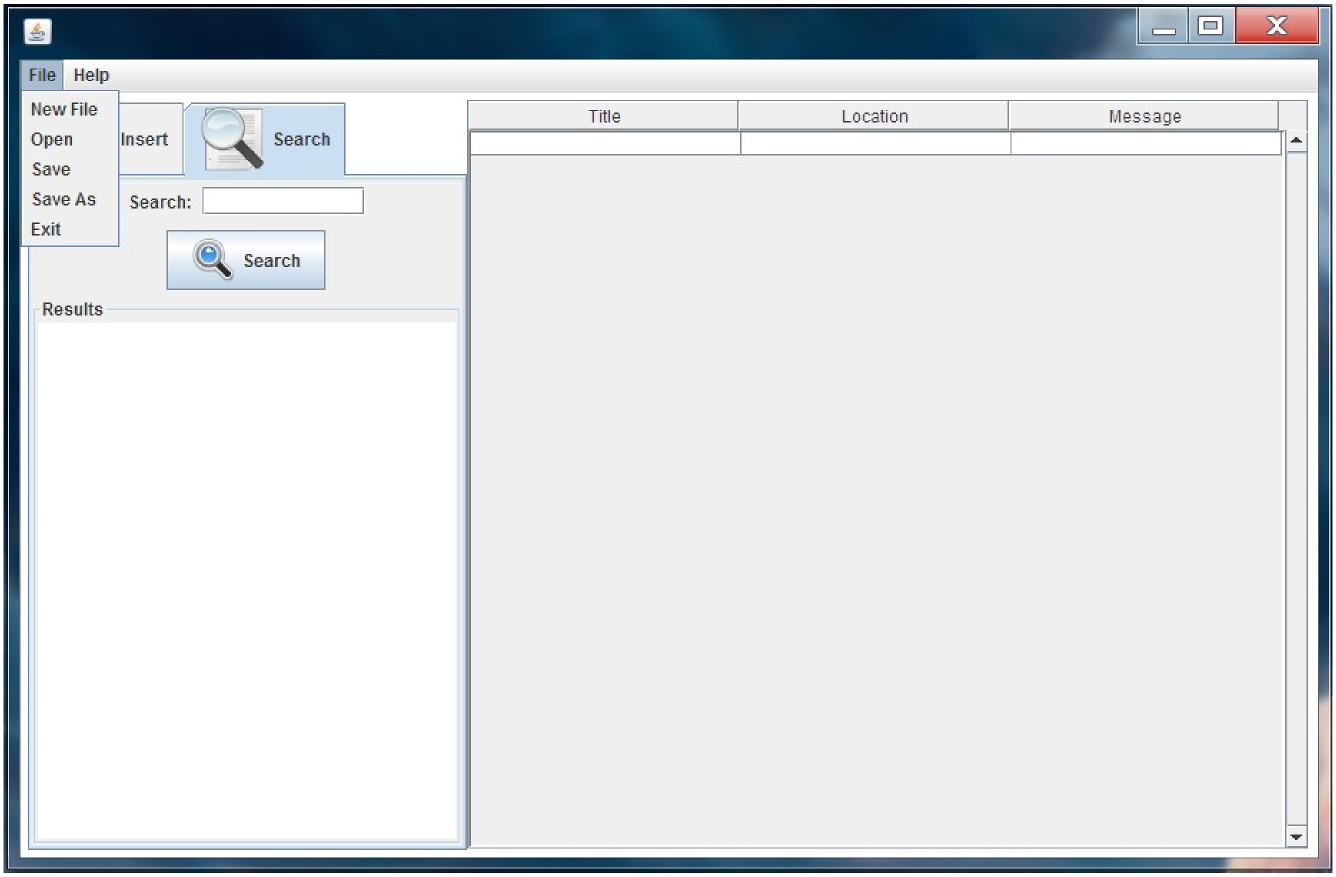
Extra clarifying information:

- Data attributes of the BillboardGUI class are listed in footnote 1 so the UML diagram may be displayed on one page.
- Most accessor and mutator methods are not specified in the class diagrams to reduce bloat
- EnumSearchType is used in place of integer constants to determine which attribute to base a search on
- The creation of the GUI is simple enough that it can be handled entirely in the constructor of the JFrame class.
- Buttons and menubar items are assigned independent action listeners which activate them they are selected by the user.
- The user interface screenshots for the different use cases are largely the same. Most are invoked by the File or Help menus and have no additional GUI beyond that.

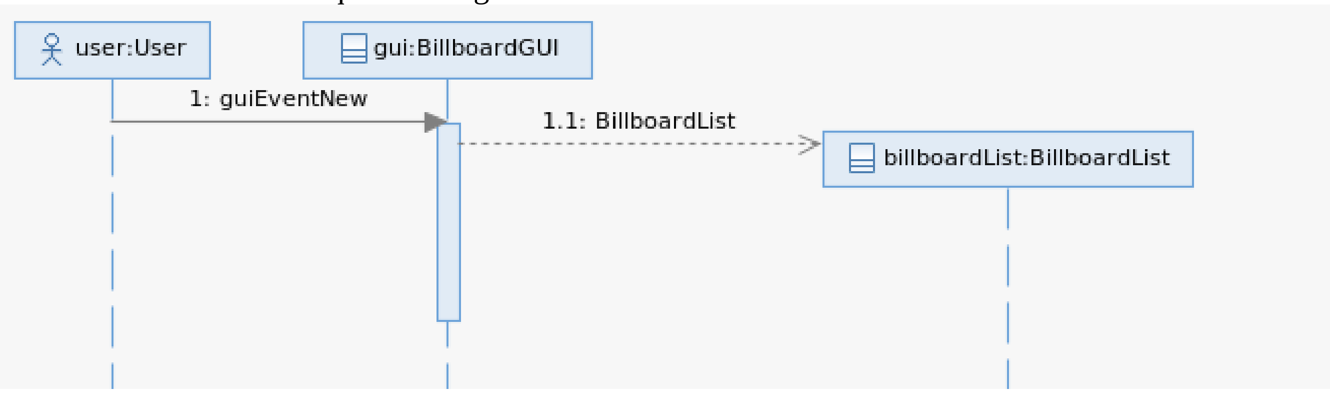
2.2 – Design Behavioural Viewpoint

2.2.1 Use-case-1 Realization View – New BillboardList

2.2.1.1 User Interface



2.2.1.2 Realization Sequence Diagram

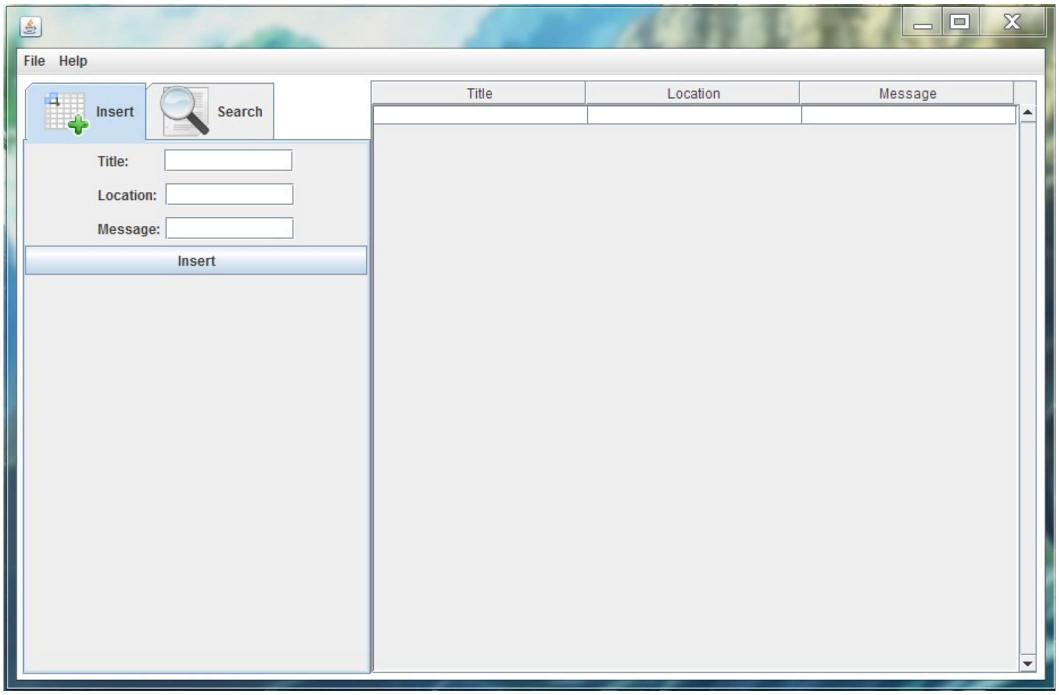


2.2.1.3 Description

This use case is invoked when the user selects the "File-New" menubar item. All this process involves is allocating a new BillboardList object in memory and clearing the GUI, so most of the work is done in the BillboardGui class. There isn't anything special to mention here. No file I/O or BillboardList methods are invoked (aside from the constructor), and there's only a short method call in the BillboardGUI class. There is no known way for this to fail, and thus, no alternative cases.

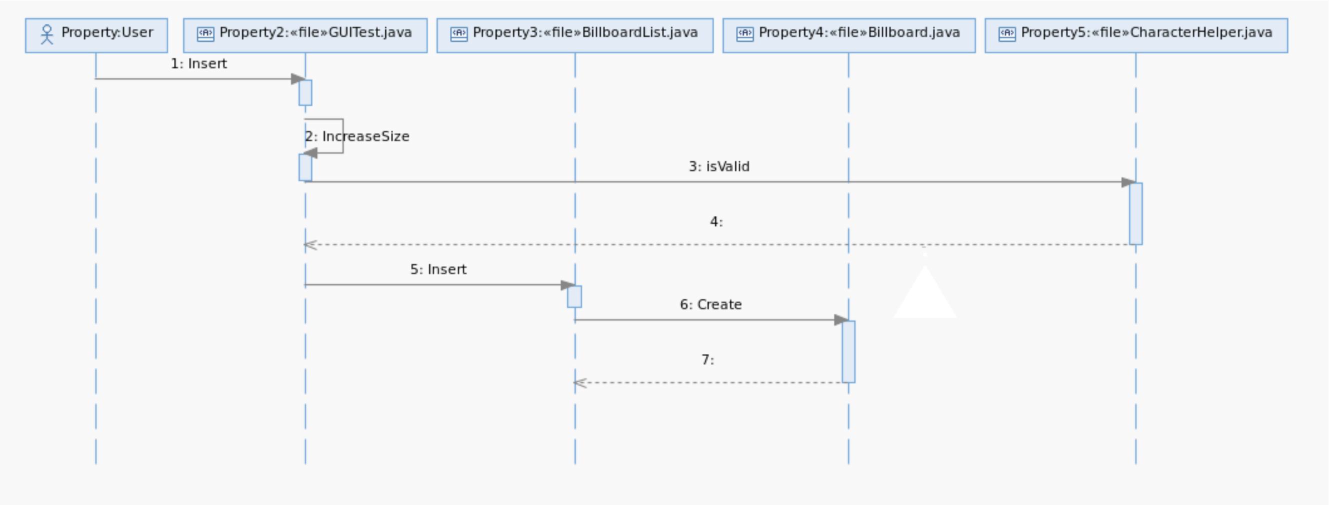
2.2.2 Use-case-2 Realization View – Insert

2.2.2.1 User Interface



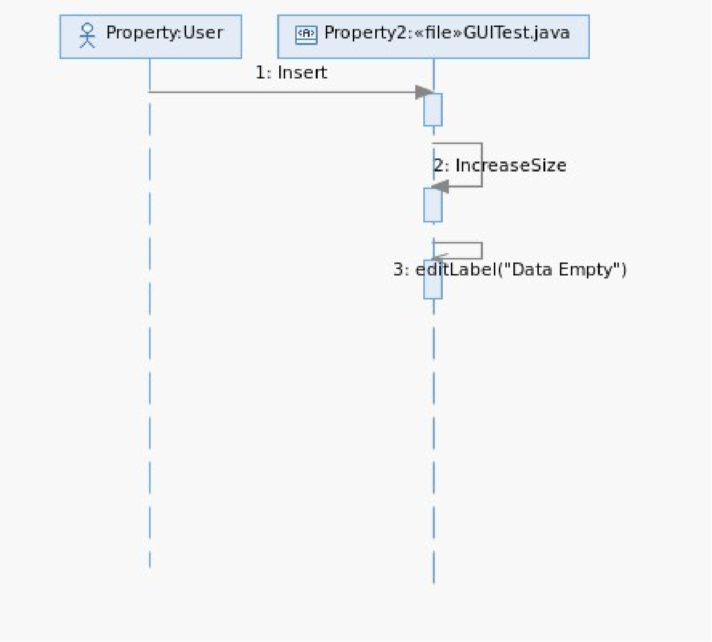
2.2.2.2 Realization Sequence Diagram

Main Case:

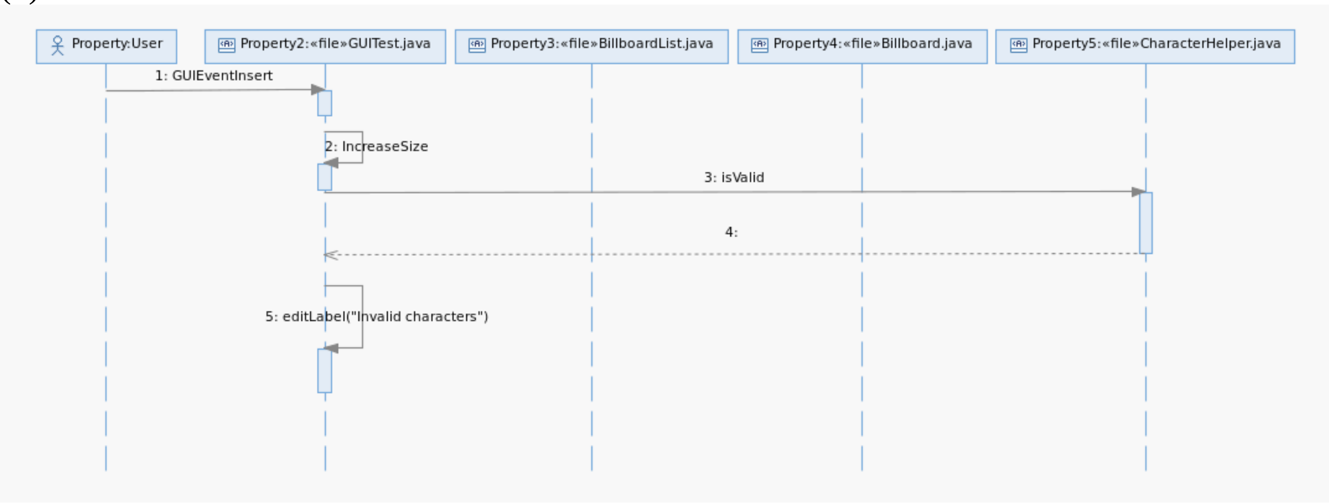

Property:User
Property2:«file»GUITest.java
Property3:«file»BillboardList.java
Property4:«file»Billboard.java
Property5:«file»CharacterHelper.java
1: Insert
2: IncreaseSize
3: isValid
4:
5: Insert
6: Create
7:

Alternate Cases:

(i)



(ii)

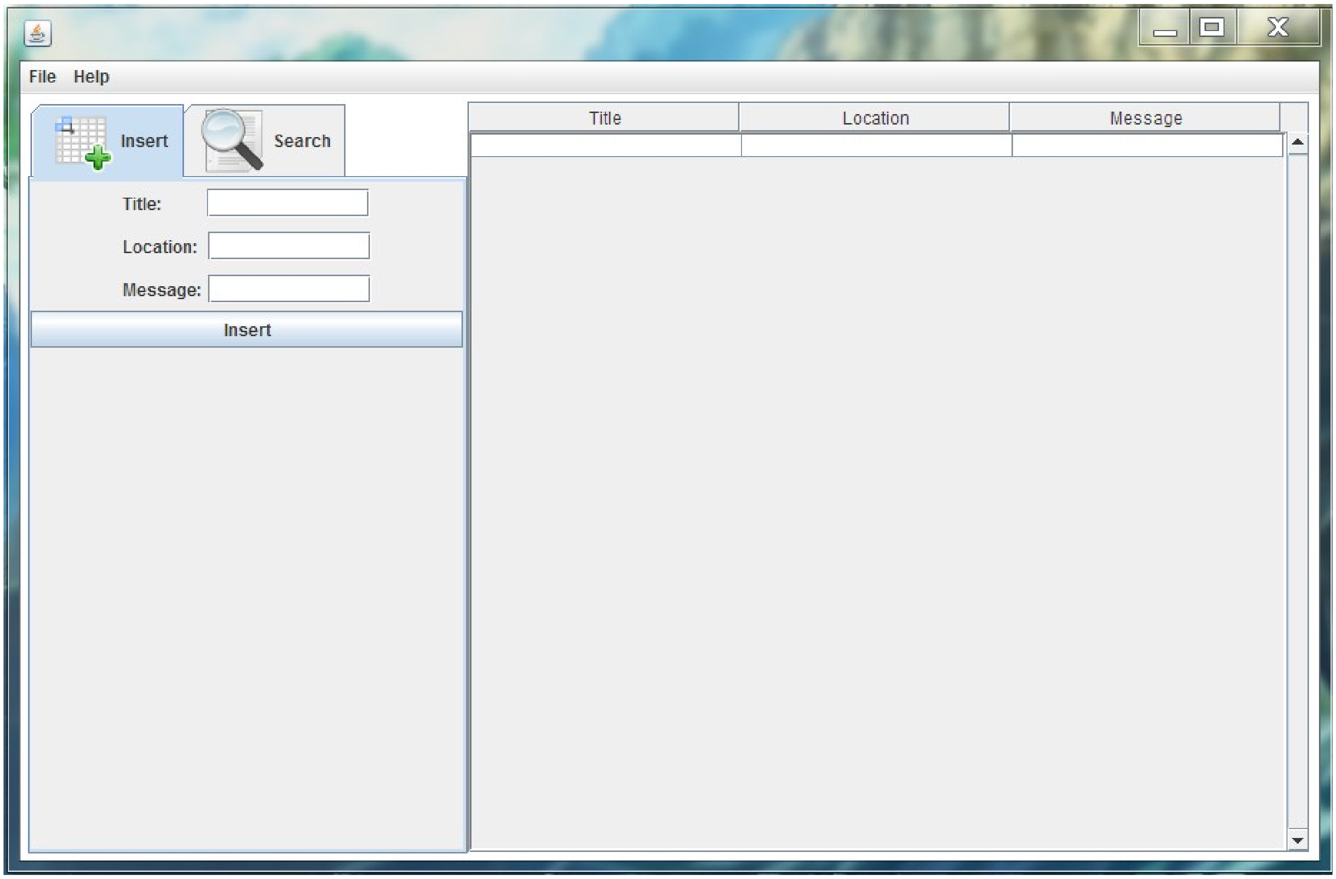

Property:User
Property2:«file»GUITest.java
Property3:«file»BillboardList.java
Property4:«file»Billboard.java
Property5:«file»CharacterHelper.java
1: GUIEventInsert
2: IncreaseSize
3: isValid
4:
5: editLabel("Invalid characters")

2.2.2.3 Description

Inserting will add one Billboard object with a title, message, and location into the BillllboardList class and update the table accordingly by adding one row with the data. It checks with the CharacterHelper class to see if the data is valid, then sends off the data to the BillboardList class. Alternate cases handle the case where no data was entered, in which case the system will not add an empty row; and the case where there are invalid characters entered.

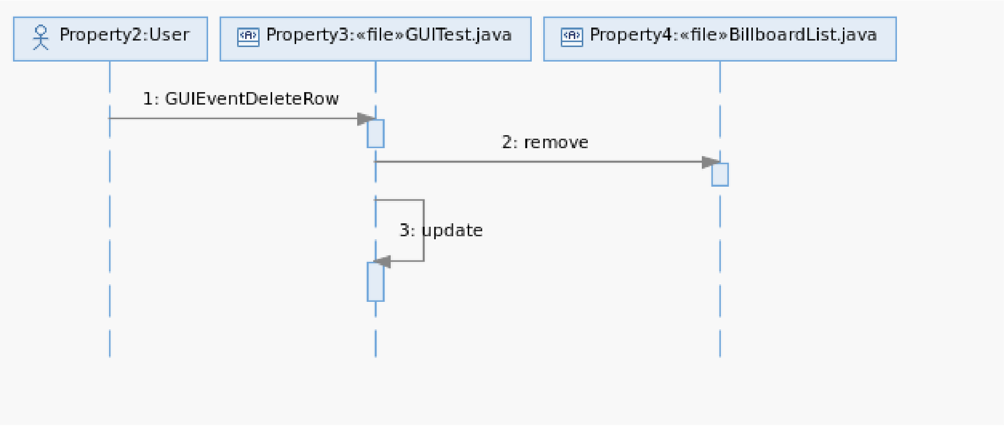
2.2.3 Use-case-3 Realization View – Remove Billboard

2.2.3.1 User Interface

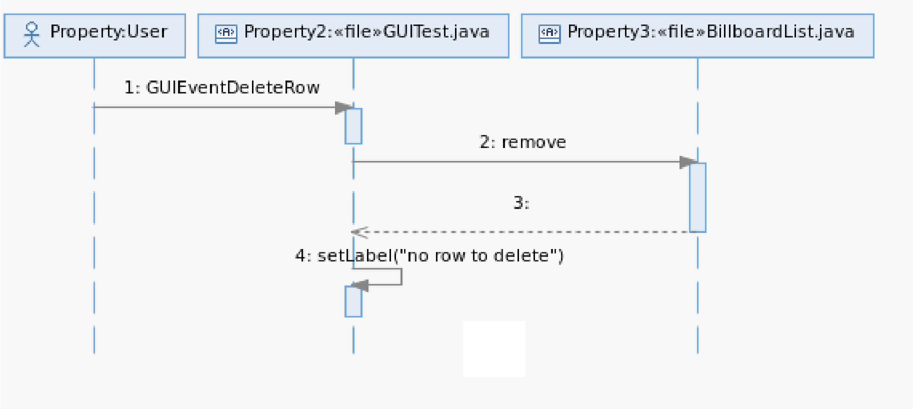


2.2.3.2 Realization Sequence Diagram

Main Case:


Property2:User
Property3:«file»GUITest.java
Property4:«file»BillboardList.java
1: GUIEventDeleteRow
2: remove
3: update

Alternate Cases:
(i)

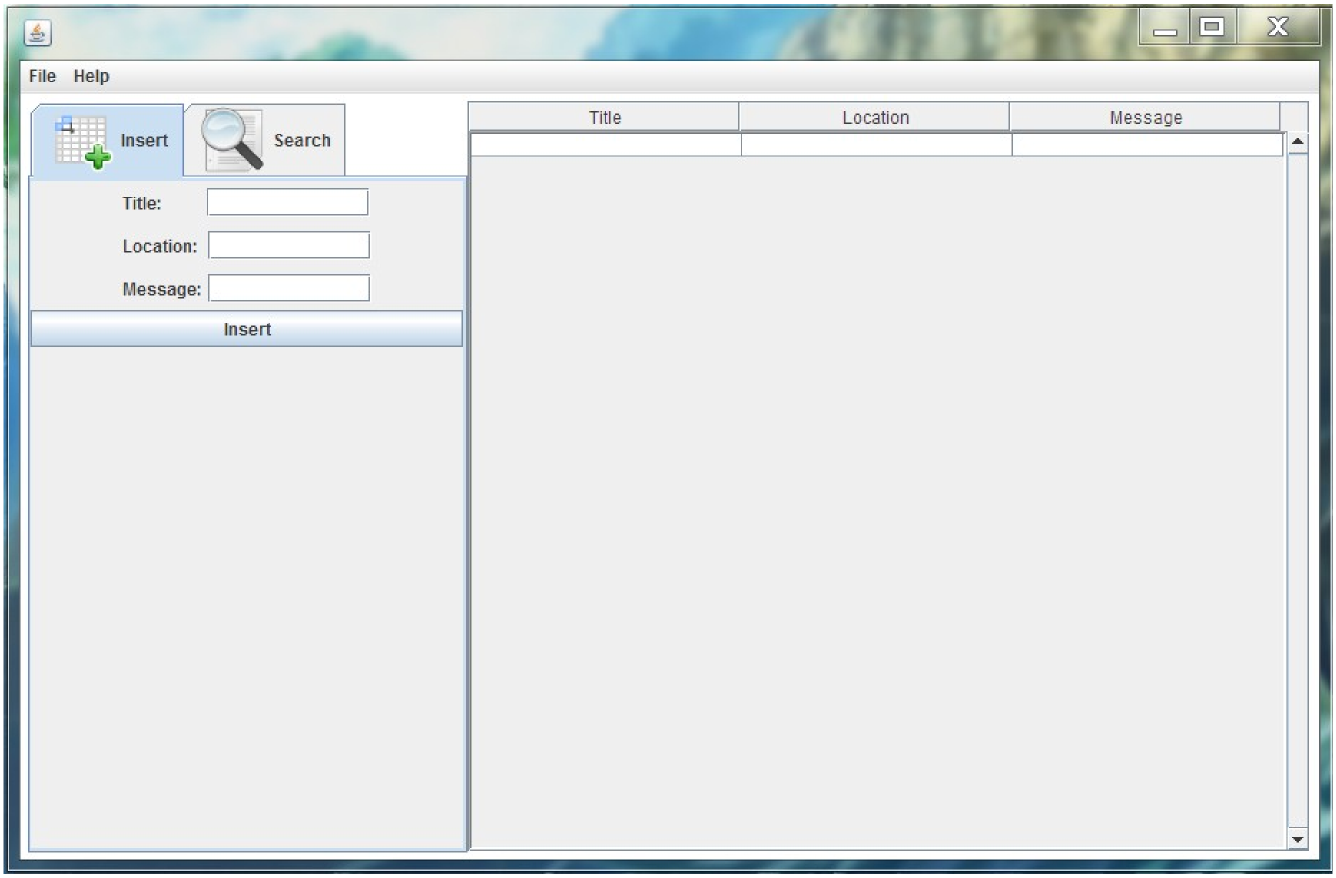

Property:User
Property2:«file»GUITest.java
Property3:«file»BillboardList.java
1: GUIEventDeleteRow
2: remove
3:
4: setLabel("no row to delete")

2.2.3.3 Description

The remove use case will remove a billboard of the user's choosing from the BillboardList and the GUI. The billboard is removed from the list using the remove() method and specifying the index of the billboard to remove - which will correspond to the row of the table being removed. After the billboard is removed from the list, the GUI is updated to reflect the new data. Alternative case (i) handles the possibility that there is no billboard to delete; in which case an error message will be issued.

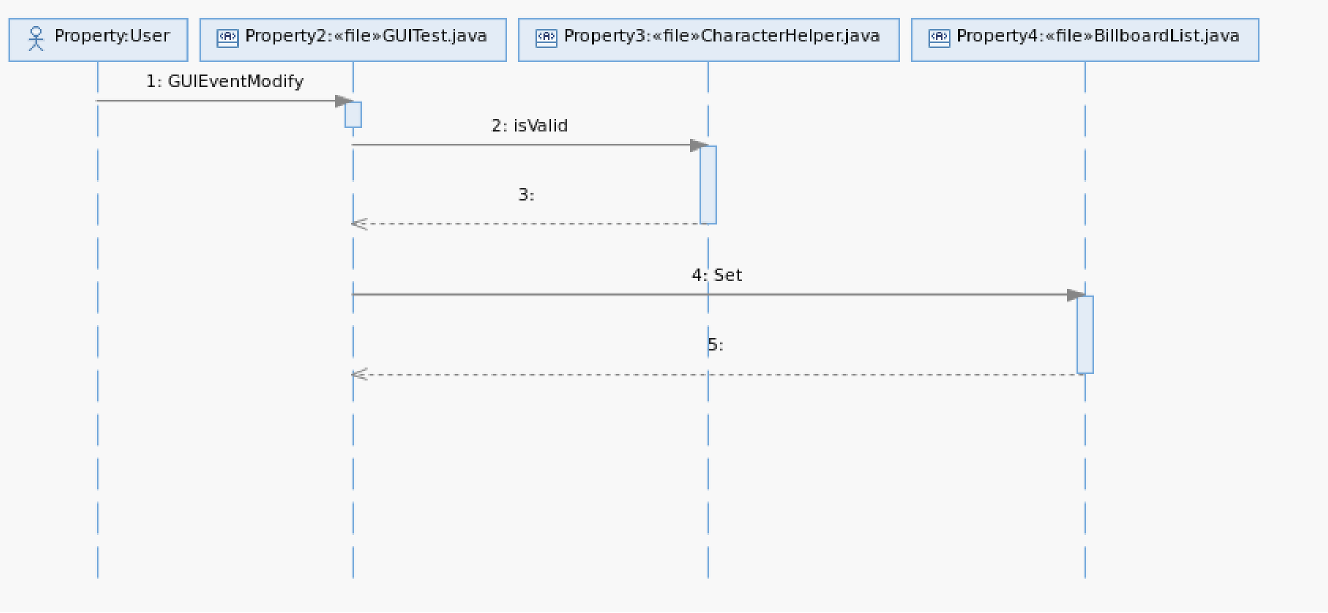
2.2.4 Use-case-4 Realization View – Modify Billboard

2.2.4.1 User Interface

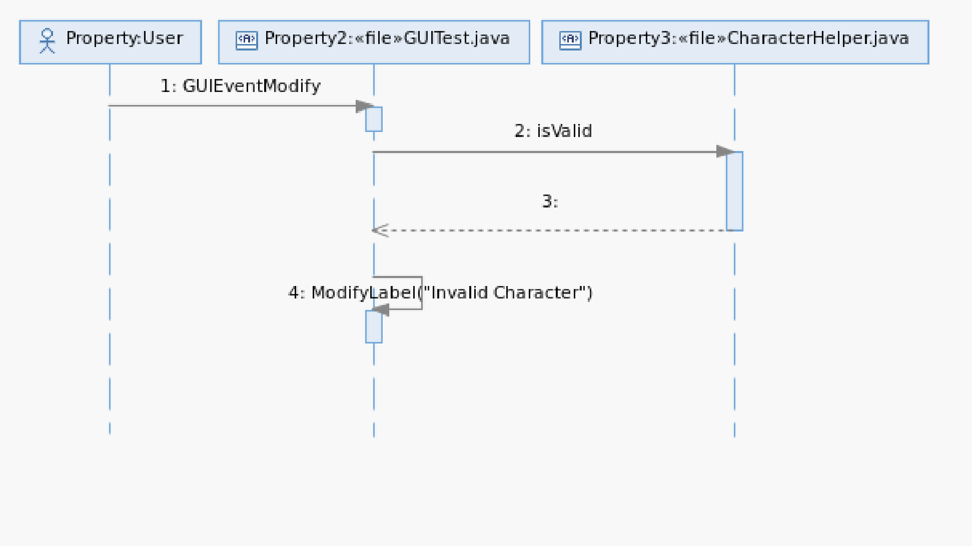


2.2.4.2 Realization Sequence Diagram

Main Case:


Property:User
Property2:«file»GUITest.java
Property3:«file»CharacterHelper.java
Property4:«file»BillboardList.java
1: GUIEventModify
2: isValid
3:
4: Set
5:

Alternate Cases:
(i)

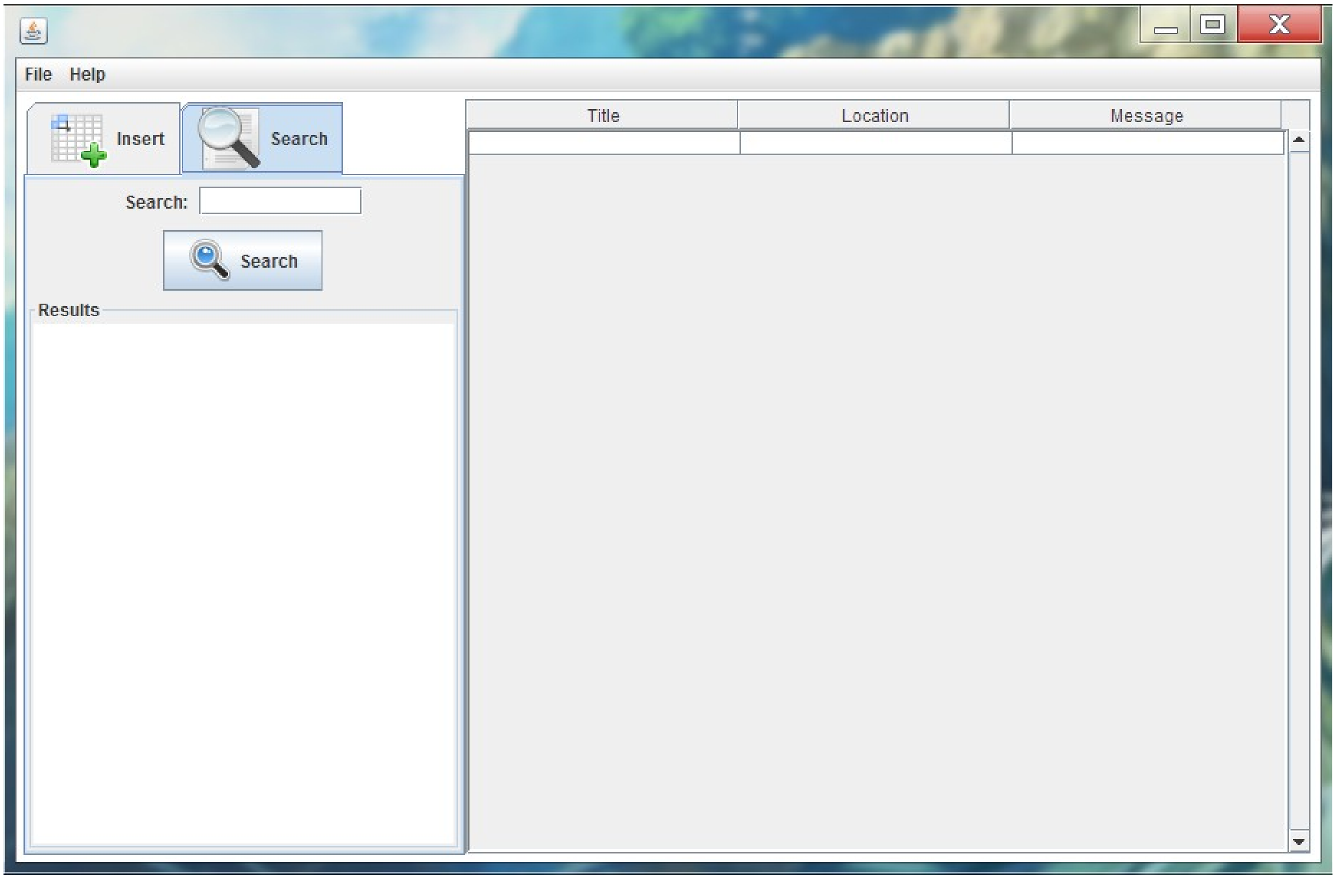

Property:User
Property2:«file»GUITest.java
Property3:«file»CharacterHelper.java
1: GUIEventModify
2: isValid
3:
4: ModifyLabel("Invalid Character")

2.2.4.3 Description

This case is invoked by the user selecting a cell on the table and directly modifying it by entering the new data from the keyboard. The data is then checked to make sure it is valid and the BillboardList is updated. Not much can go wrong here, except the user entering invalid characters.

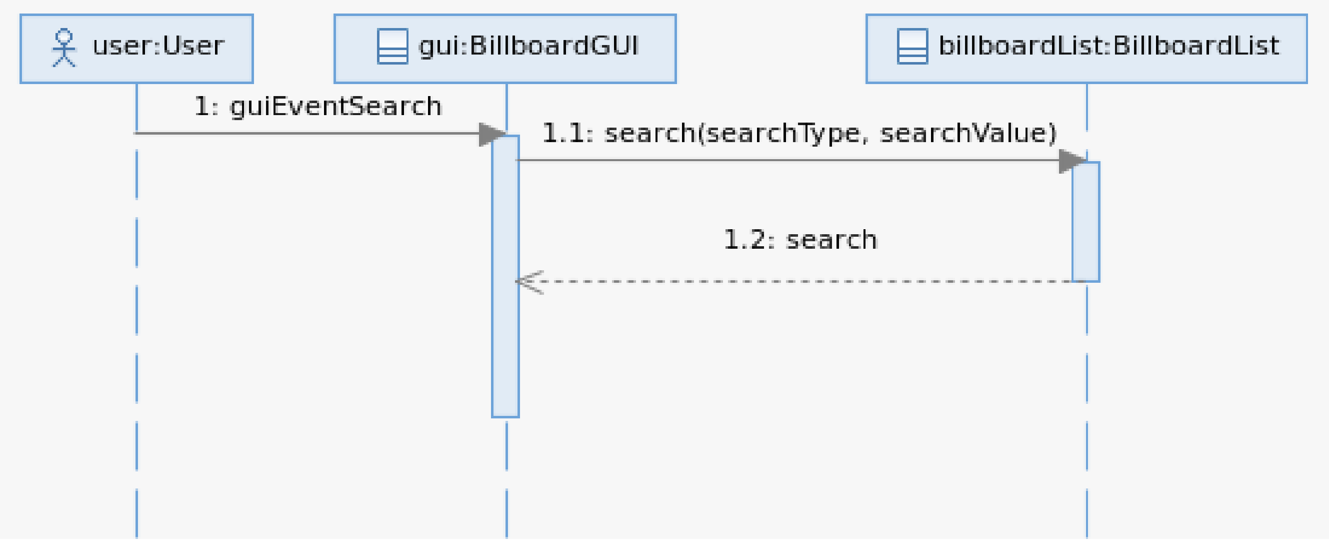
2.2.5 Use-case-5 Realization View – Search

2.2.5.1 User Interface

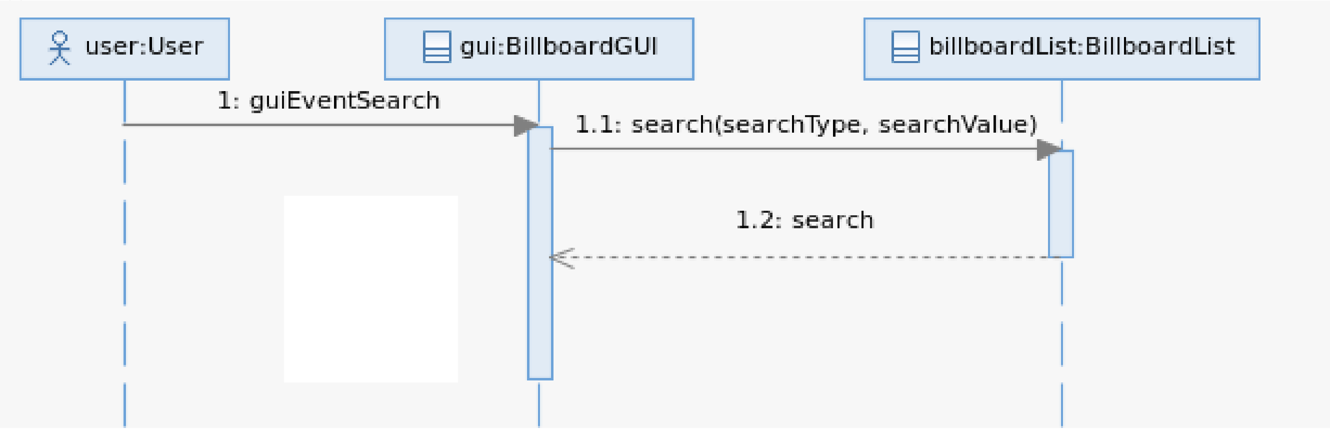


2.2.5.2 Realization Sequence Diagram

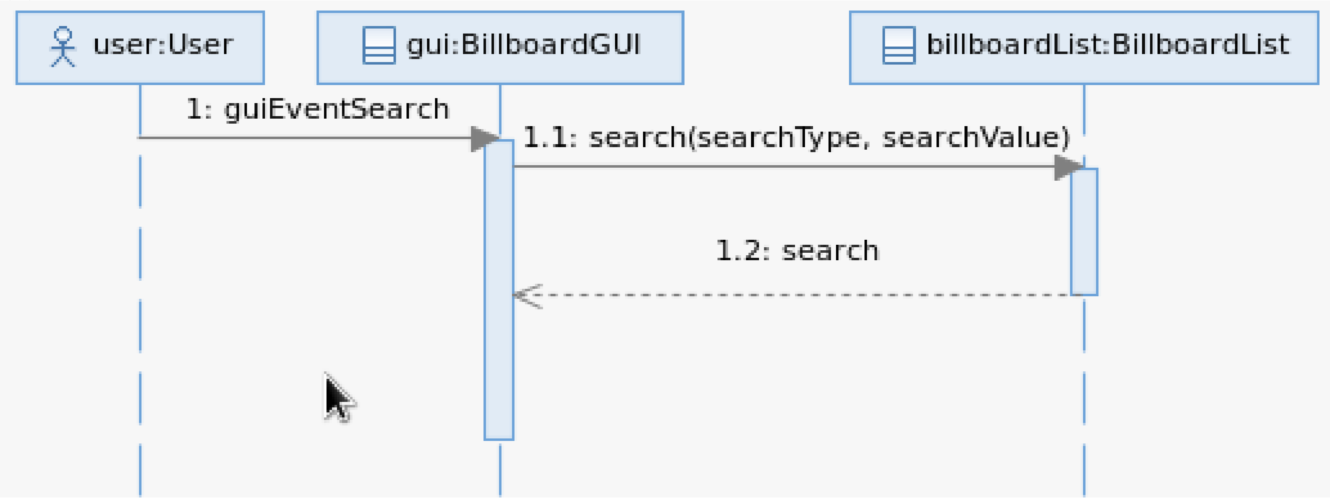
Main Case:


user:User
gui:BillboardGUI
billboardList:BillboardList
1: guiEventSearch
1.1: search(searchType, searchValue)
1.2: search

Alternative cases:
(i)


user:User
gui:BillboardGUI
billboardList:BillboardList
1: guiEventSearch
1.1: search(searchType, searchValue)
1.2: search

(ii)

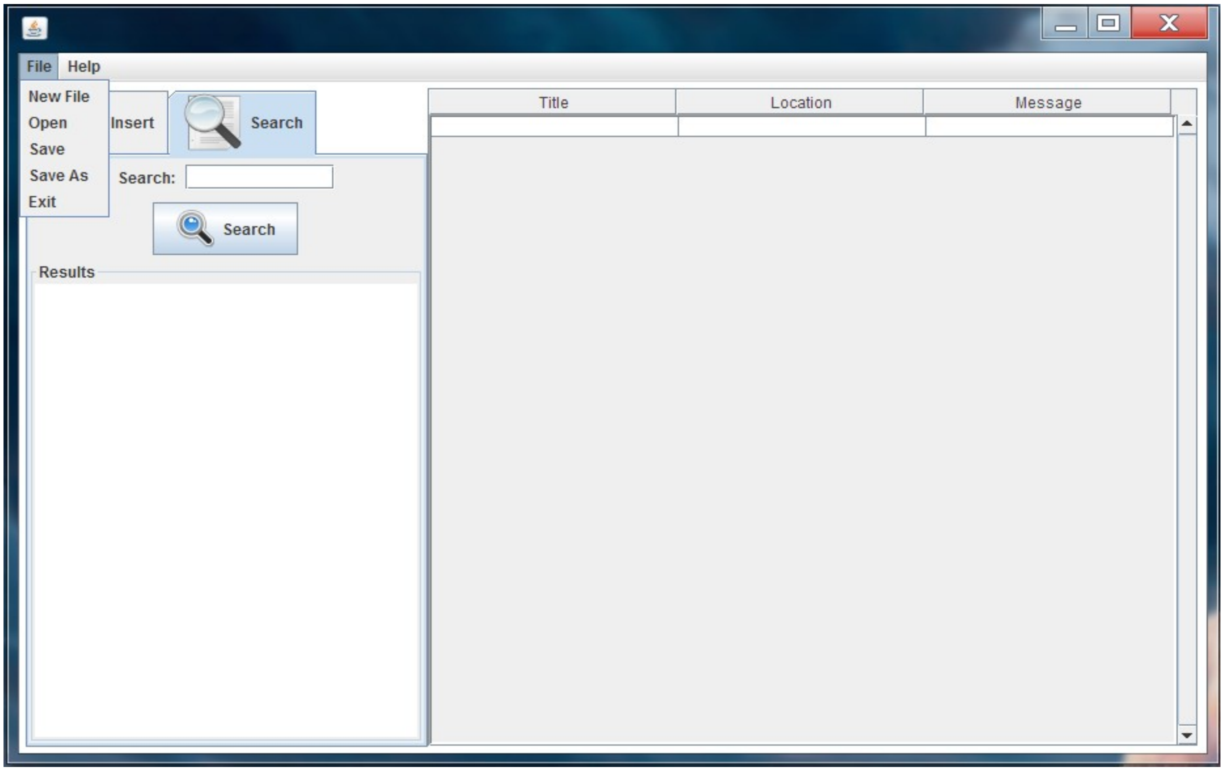

user:User
gui:BillboardGUI
billboardList:BillboardList
1: guiEventSearch
1.1: search(searchType, searchValue)
1.2: search

2.2.5.3 Description

This case is invoked by the user selecting the “Search” button in the search tab. It takes the relevant search parameters from the fields in the GUI, which are: what data attribute to base the search on and the value to search for. Most of this is handled in the BillboardList search method, which will return any applicable values in the form of a List<Billboard>. All the cases perform a similar search but the alternative cases will display the data slightly differently.

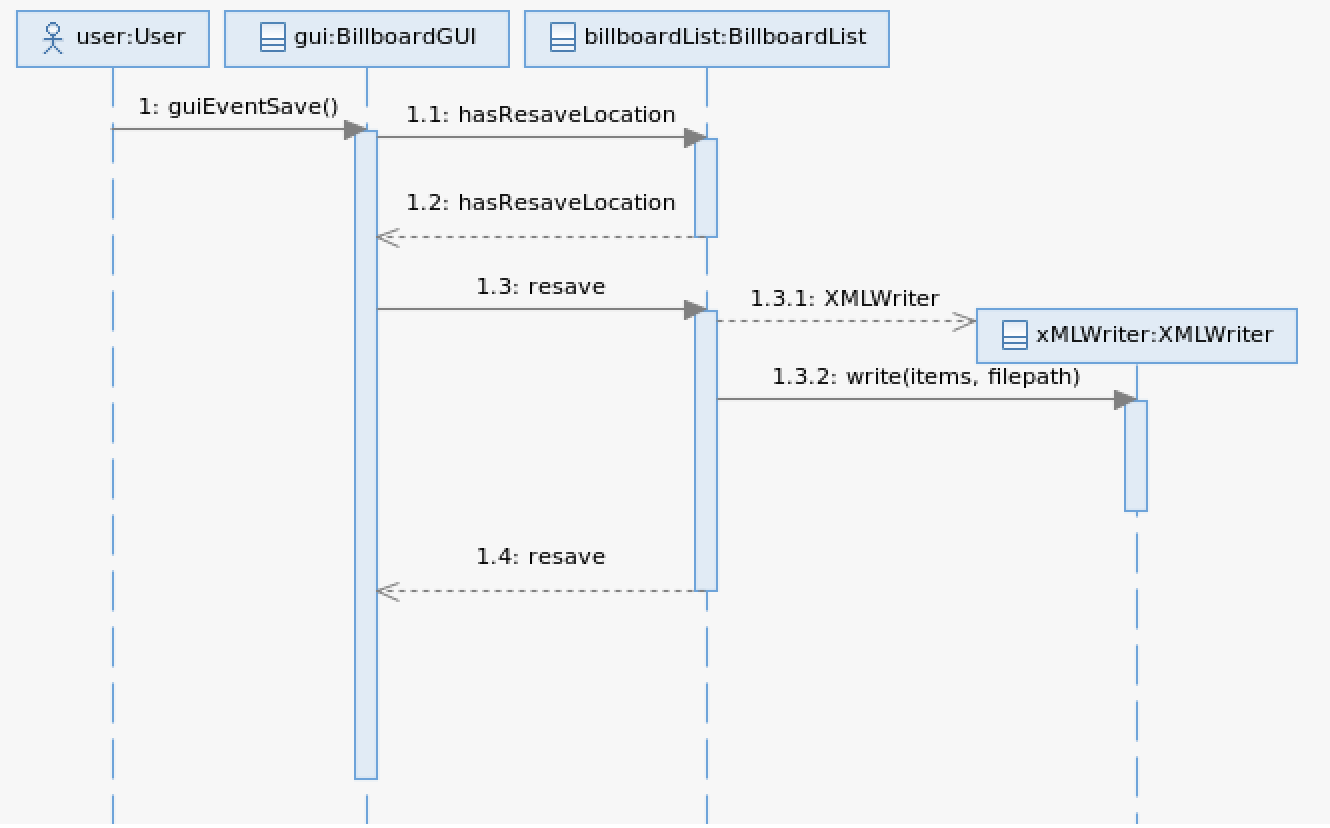
2.2.6 Use-case-6 Realization View – Save XML File

2.2.6.1 User Interface



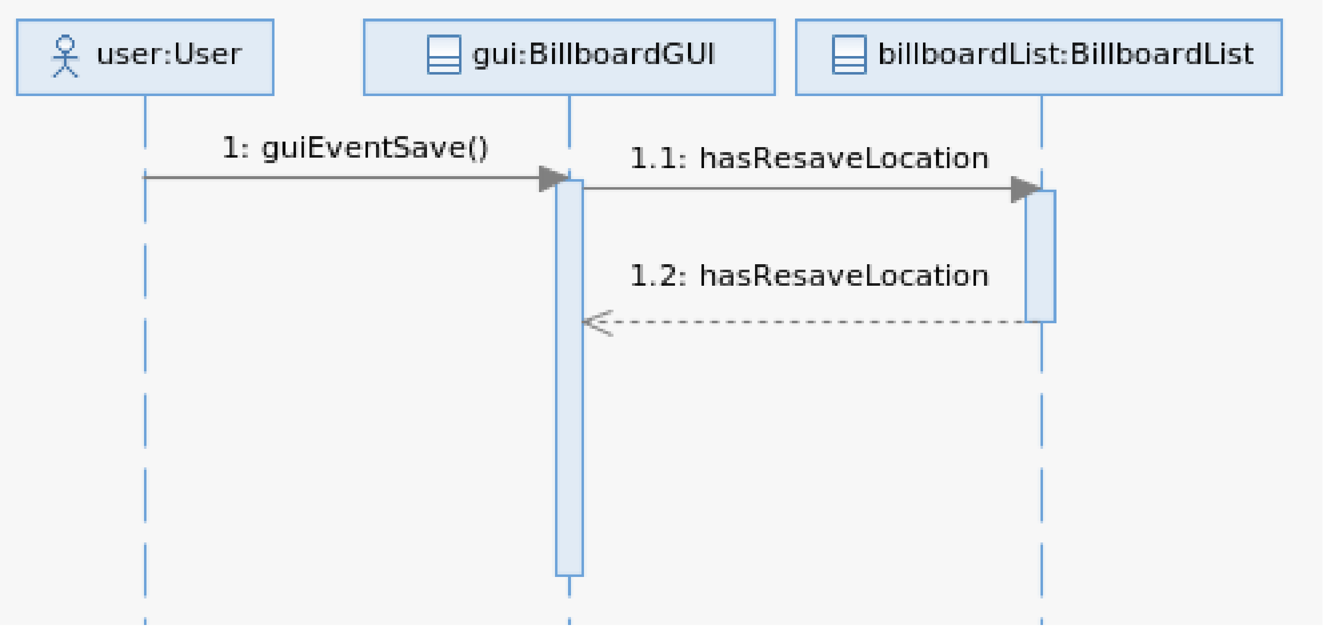
2.2.6.2 Realization Sequence Diagram

Main Case:

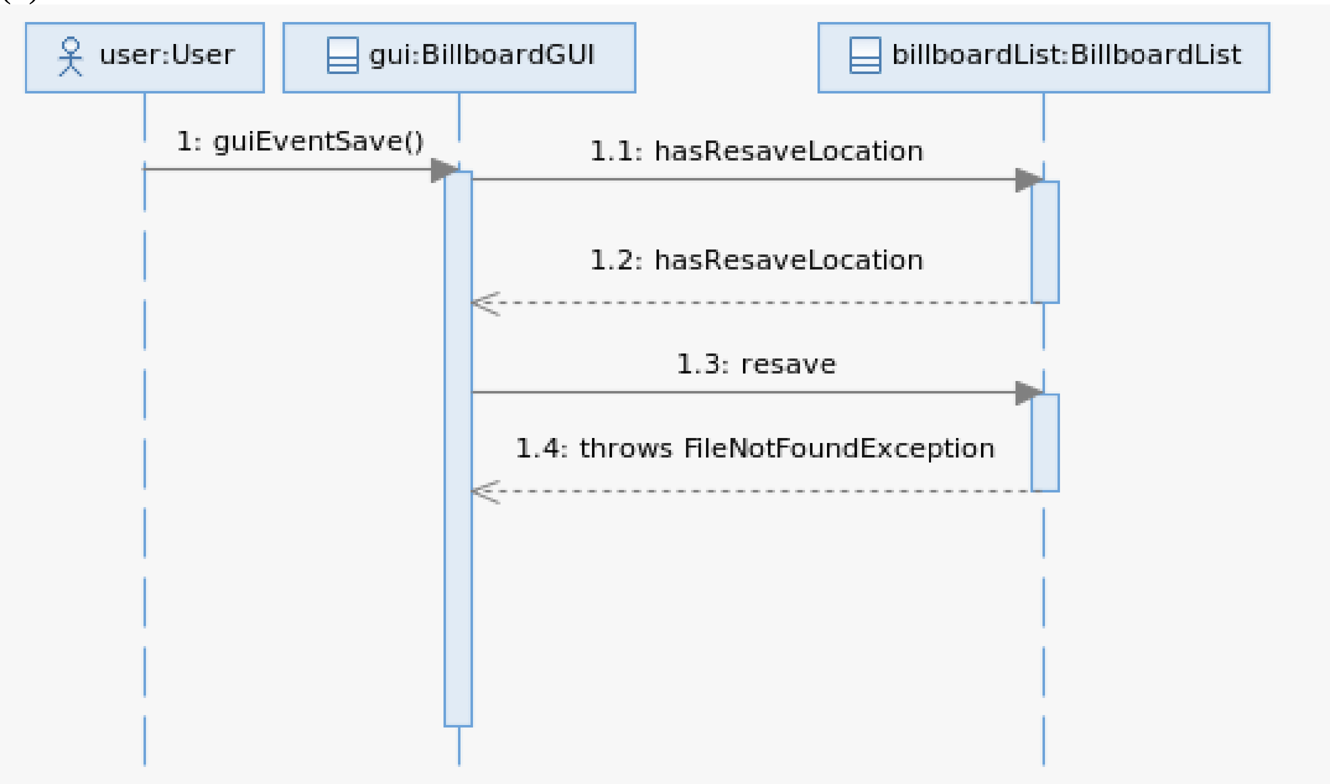


Alternative Cases:

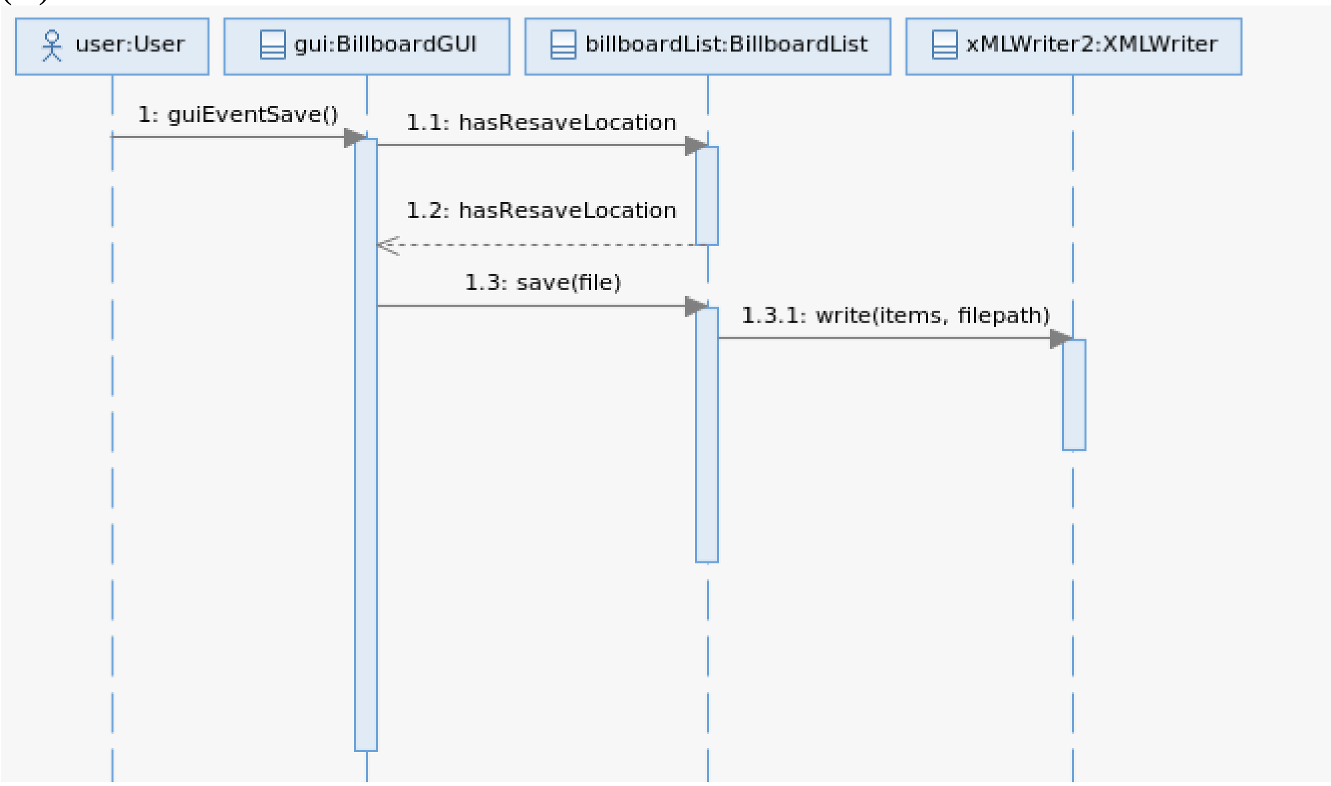
(i)



(ii)



(iii)

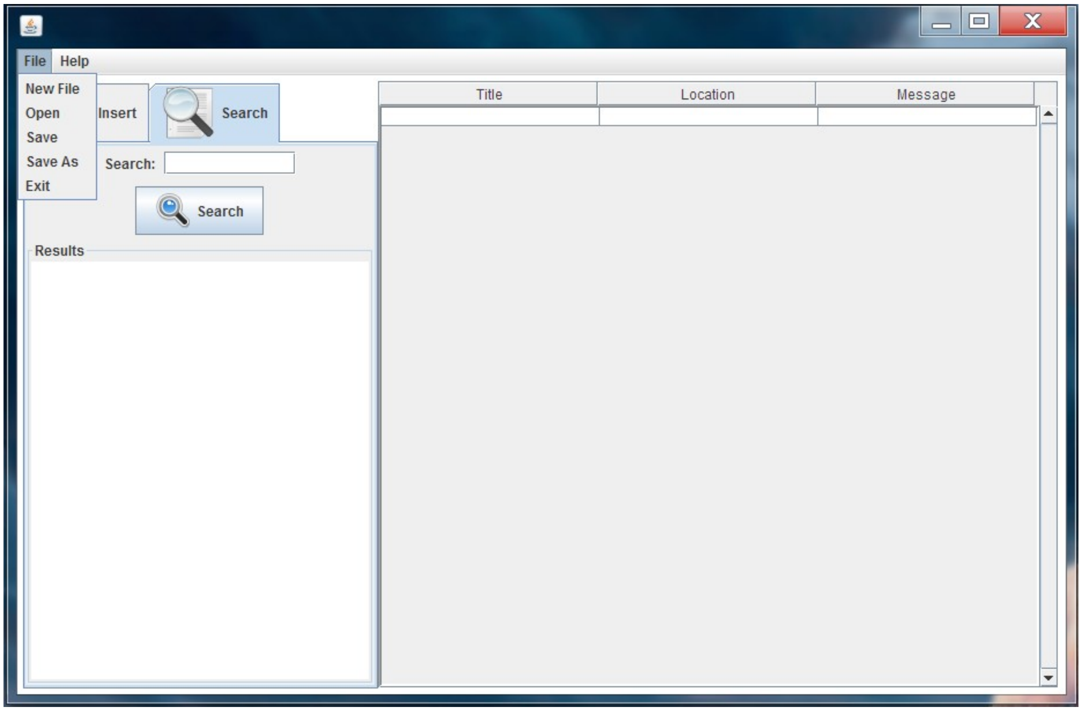


2.2.6.3 Description

This case is invoked by the user pressing the “File-Save” menubar item. As with other programs, the save functionality is different than the “Save As” functionality. It requires that a file has been loaded or saved so that the system knows where to resave the file. So, the first thing that the save function has to do is ensure there's a BillboardList in memory to save. If there isn't a BillboardList to save, then no file I/O will happen – this case is handled under alternative case (i). Secondly, the program has to make sure there's a valid resave location for the BillboardList. If there isn't, then like other programs the “Save-As” functionality gets invoked by default. This is handled by alternative case (iii). The final alternative case is that somehow the destination file doesn't exist; this possibility is handled by alternative case (ii). Should all these checks pass successfully then the file will write successfully.

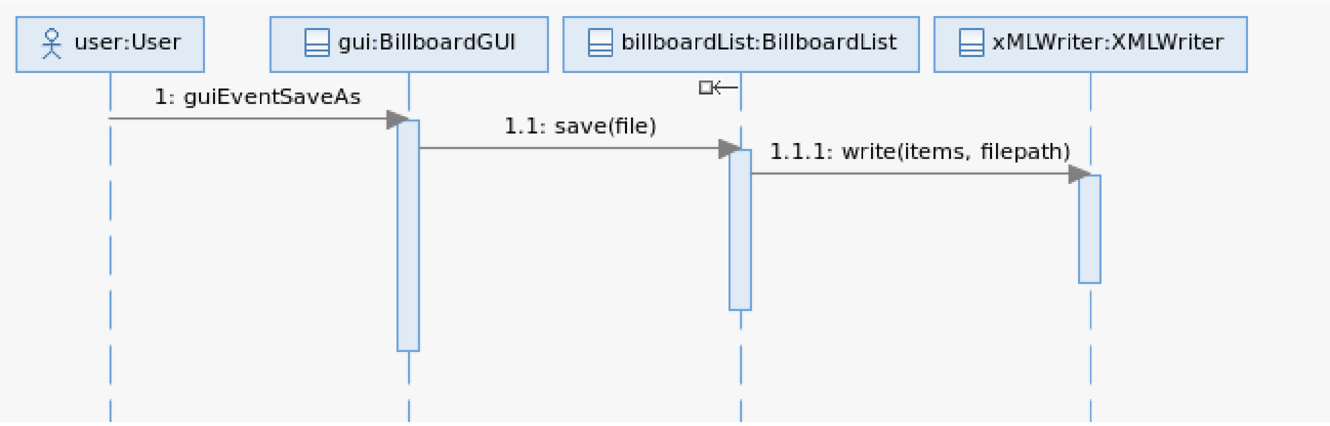
2.2.7 Use-case-7 Realization View – “Save As”

2.2.7.1 User Interface



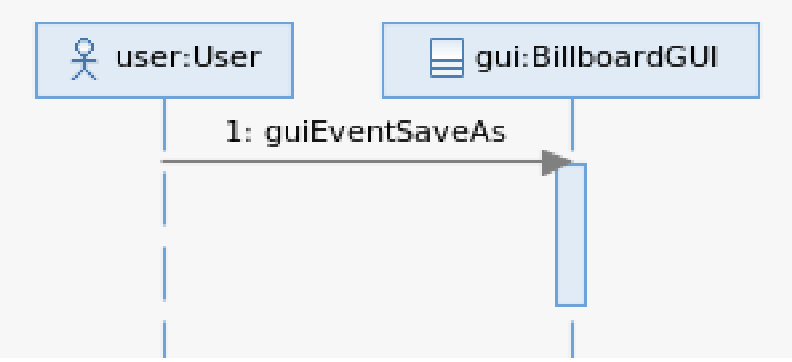
2.2.7.2 Realization Sequence Diagram

Main Case:

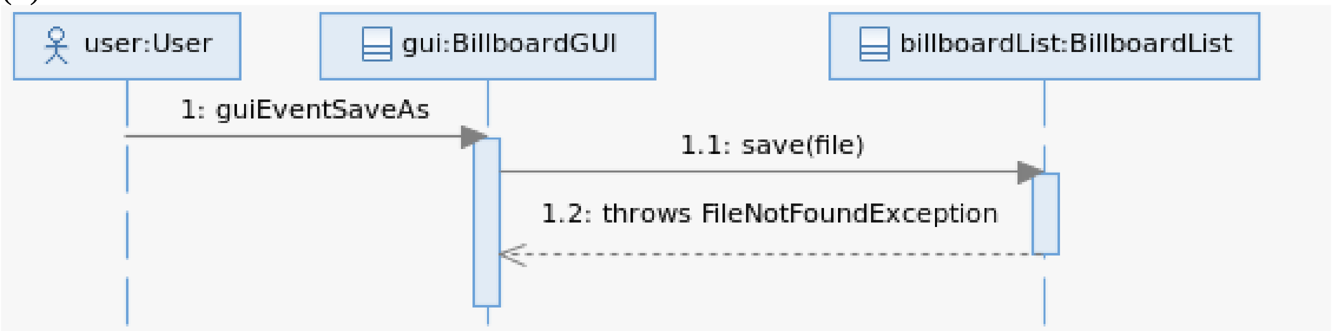


Alternative Cases:

(i)



(ii)

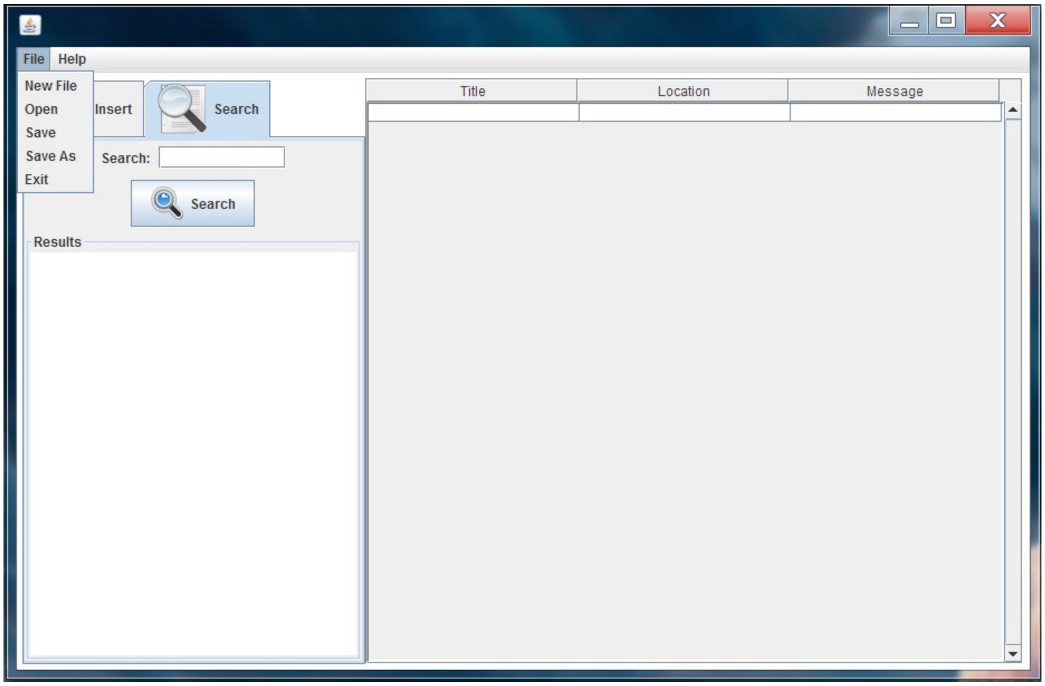

user:User
gui:BillboardGUI
billboardList:BillboardList
1: guiEventSaveAs
1.1: save(file)
1.2: throws FileNotFoundException

2.2.7.3 Description

This case is invoked by the user pressing the “File-Save As...” menubar item. This functionality allows the user to save a BillboardList to a specified location on disk. This differs from the save functionality because it allows the user to select exactly where the BillboardList will be saved. The alternative cases handle things that could possibly go wrong when saving; while the main case assumes everything goes successfully. Alternative case (i) handles the case where the user selects “File-Save As” but decides to cancel the save while the JFileChooser is open – in which case no file I/O will happen. Alternative case (ii) handles the case where a FileNotFoundException is caused, because somehow an invalid filepath was given. Saving is handled largely by the backend, while the filepath is gathered in the gui.

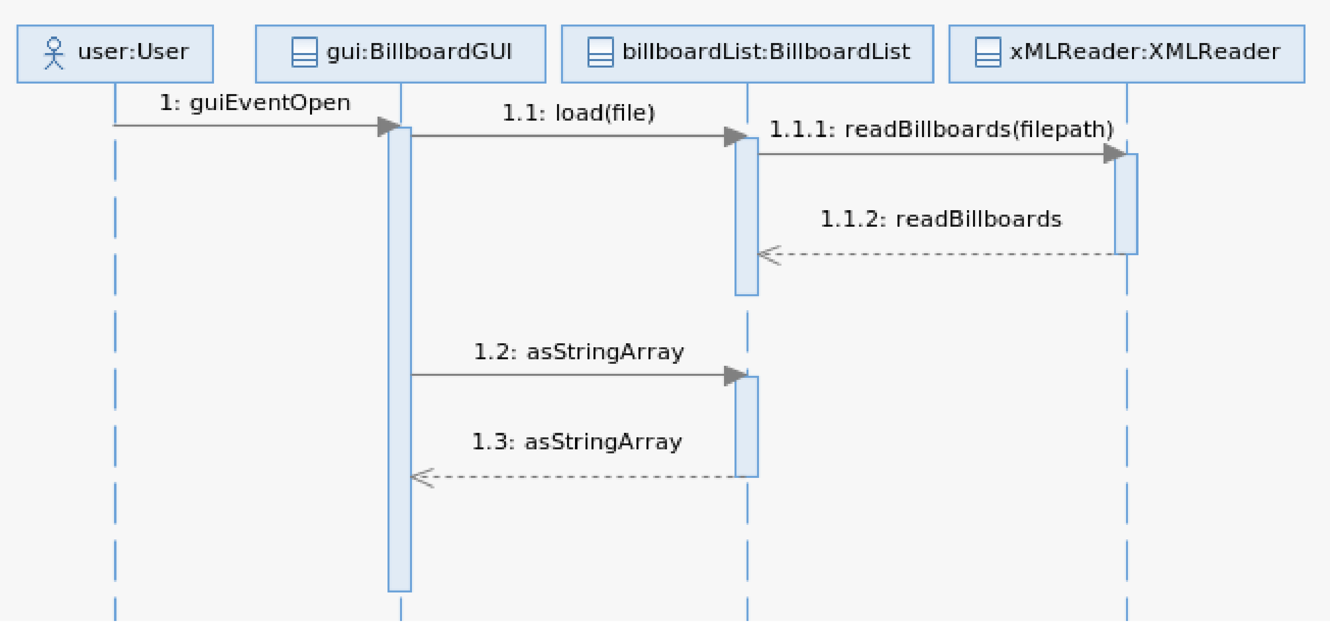
2.2.8 Use-case-8 Realization View – Open XML File

2.2.8.1 User Interface



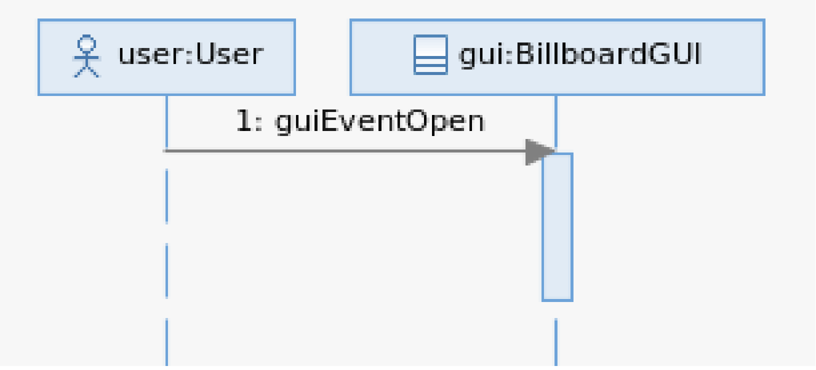
2.2.8.2 Realization Sequence Diagram

Main Case:

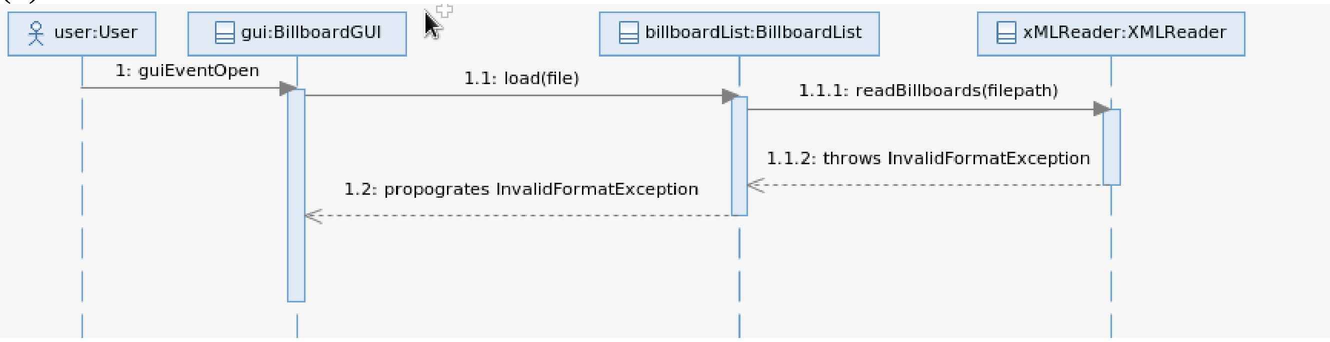


Alternative Cases:

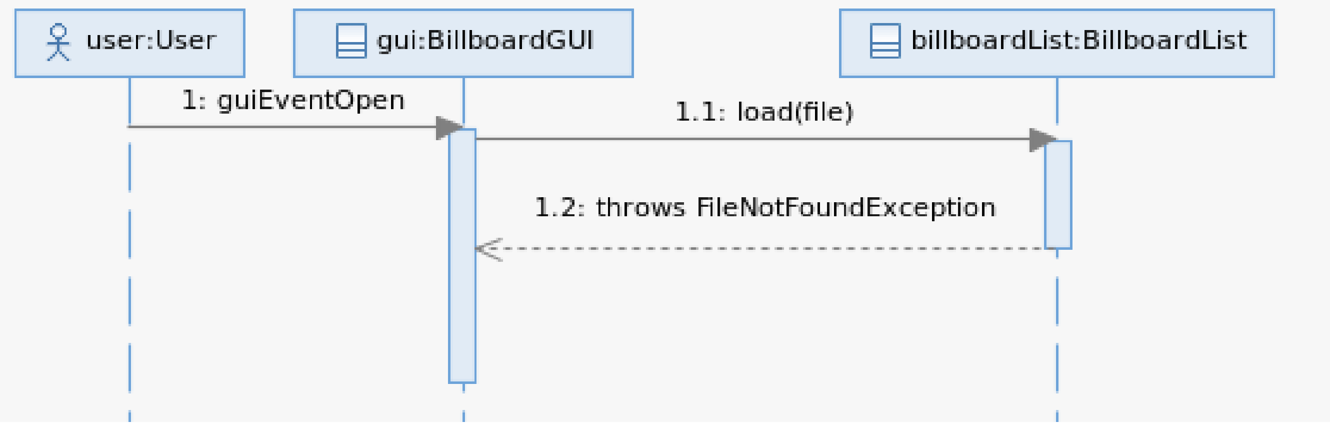
(i)



(ii)


user:User
gui:BillboardGUI
billboardList:BillboardList
xMLReader:XMLReader
1: guiEventOpen
1.1: load(file)
1.1.1: readBillboards(filepath)
1.1.2: throws InvalidFormatException
1.2: propogrates InvalidFormatException

(iii)


user:User
gui:BillboardGUI
billboardList:BillboardList
1: guiEventOpen
1.1: load(file)
1.2: throws FileNotFoundException

2.2.8.3 Description

This case is invoked when the user selects the “File-Open” menubar item. This functionality allows the user to open a BillboardList XML file that already exists on the hard drive. The majority of this process, including data validation, is handled in the backend. The GUI is responsible for attaining the filepath of the XML file to open. After opening the file and reading it into memory, the GUI is then updated to reflect the new data. The main case assumes a file is successfully opened. Alternative case (i) handles the case where the user selects the “File-Open” option but opts to press the decline button in the JFileChooser. Alternative case (ii) handles the case where an invalid file format is present. In this case, an invalid file format is something that satisfies one of these following criteria: (1) The filename does not end with “.xml” or (2) the XML file is formatted in a way such that it can't be used to reconstruct a BillboardList. Alternative case (iii) handles the case where the user somehow manages to select a file that does not exist with the JFileChooser.

Appendix A: Footnotes

[1] Instance Variables of the BillboardGUI class:
The following is a non-exhaustive list of all the instance variables of the BillboardGUI class, specified in UML format. As this is only a prototype GUI this list is subject to slight changes or additions during the implementation phase to correct any layout or functionality issues.

- billboards : BillboardList
- menuFile : JMenu
- menuItemNewFile : JMenuItem
- menuItemSaveAs : JMenuItem
- menuItemExit : JMenuItem
- menuBar : JMenuBar
- menuHelp : JMenu
- menuItemAbout : JMenuItem
- menuItemNew : JMenuItem
- menuItemOpenFile : JMenuItem
- contentPane : JPanel
- tab1Pane : JPanel
- tab2Pane : JPanel
- tab3Pane : JPanel
- table : JTable
- tableScrollPane : JScrollPane
- panelTitle : JPanel
- panelInsert : JPanel
- panelInsertContents : JPanel
- spinner : JSpinner
- labelInsertRows : JLabel
- buttonInsert : JButton
- labelTitle : JLabel
- textFieldTitle : JTextField
- panelLocation : JPanel
- labelLocation : JLabel
- textFieldLocation : JTextField
- panelInsertContents2 : JPanel
- panelMessage : JPanel
- labelMessage : JLabel
- textFieldMessage : JTextField
- panelInnerInsert : JPanel
- buttonInsertRow : JButton
- columnNames : String[]
- data : String[][]
- model : DefaultTableModel