University of Central Arkansas

Software Engineering

# User's Manual: UCA Scheduler

11th December 2011

# Contents

# List of Figures

# 1 System Introduction

## 1.1 Purpose of Document

This document is intended to give the user of the UCA Scheduler software (hereinafter user) information and instructions on the intended use and purpose of the UCA Scheduler program. This document will be a reference point for the user, allowing the user to determine the proper installation and use of the Scheduler. Upon use of this document, the user shall be able to determine the proper procedures for obtaining the desired output of the Scheduler.

## 1.2 General Overview of Software System Development

The purpose of this project is to provide a working "Scheduler" that parses HTML documents, detailing upcoming events and their dates to take place in the Student Center and Ida Waldran buildings on the University of Central Arkansas campus, generated by the software system iCal. The result of this parsing shall produce plaintext output both directly readable and usable by the user, such that each event is output in a natural language (English) and separated by date, for all events at least one day and at most thirty days in advance. For further readability, a calendar view of the events parsed from the HTML files is to also be generated. This "Scheduler" is to be used to replace the method currently used by the client to extract relevant information from iCal's output, which constitutes manually analyzing the HTML documents for relevant information and retyping it in plaintext, in order to reduce time consumed by the task for the client.

### 1.2.1 Overview of Development Process

The Scheduler was developed using a methodology knowing as the incremental development lifecycle. The development is split up into different phases, each phase focuses on one task, while completing smaller components of other tasks. This allows for components of various tasks to be developed in parallel with each other.

The phases were split up into milestones, where each milestone had a main focus, but also developed portions of the previous and next milestone.

### 1.2.2 Milestone 0

The first milestone consisted of developing the Requirements Document for the Scheduler product that the customer would be interested in. It involved creating a detailed report of exactly what features the Scheduler would be required to have.

A presentation is also involved, presenting the Requirements Document that was developed and gathering any feedback upon modifications that must be made to said document.

### 1.2.3 Milestone 1

The next milestone involved developing a Rapid Prototype to demonstrate the look and feel of the product detailed in the Requirements Document. The functionality of the Rapid Prototype does not need to exist at this point, but the basic display of such a program must exist, to give the customer a demonstration of exactly what the Scheduler will be. This also gives the customer an opportunity to make corrections and modifications to the Scheduler before a large amount of development and planning has taken place.

This milestone also continued development of the Requirements Document, such that the Requirements Document was corrected with any modifications or new additions (such as items specified by the customer during the Rapid Prototype demonstration).

A presentation must be made for this milestone, detailing any changed made to the Requirements Document, along with a demonstration of the Rapid Prototype.

### 1.2.4 Milestone 2

The development of the Rapid Prototype continued in this development milestone, adding more functionality to it to give the customer an even better demonstration of exactly what the Scheduler will do and how it will operate. This marks the end of the development of the Rapid Prototype.

The Requirements Document was updated once more to accommodate any changes presented by the customer, and any mandatory changes to logic. This point also marks the end of the changes to the Requirements Document.

This milestone signals the beginning of the development of the Specifications Document. The Specifications Document is a document that details everything included in the Requirements Document, except with more technical detail. It details how components interact with each other, what components exist, and is of much benefit to developers.

This milestone also begins development on the Software Project Management Plan (SPMP). The SPMP is a document designed to detail all aspects of the development of the Scheduler. It details how the product will be developed along with a timeframe for development. The SPMP helps to maintain order as the Scheduler is being developed, and keeps the product on track throughout the development.

Finally, the Acceptance Test Plan began development in this milestone. The Acceptance Test Plan is a detailed plan of how the requirements specified in the Requirements Document are to be tested to guarantee that they are in working order. The Acceptance Test Plan details exactly how each component is to be tested, along with the specified input and expected output for each component.

A presentation must be made about any changed made to the Requirements Document, a demonstration of the updated Rapid Prototype, details of the

Specifications Document, Software Project Management Plan, and the Acceptance Test Plan.

### 1.2.5 Milestone 3

Milestone 3 includes an updated Specifications Document, including more details about specifications (such as Scenarios (expected normal and abnormal use of the Scheduler), Class Modelling (a portrayal of each class in the product and its expected functionality), and State Chart Diagram (a detailed diagram of every state the Scheduler could be in)).

The SPMP was also updated to include more details about the Scheduler. The update includes an update to the Work Breakdown Structure that reflects all work performed on the Scheduler, as well as an update to the schedule of when components are expected.

This milestone also begins development of the High-Level Design, which is a more detailed design of the Scheduler and all of the components involved in it. The actual implementation of the Scheduler should reflect the design detailed in the High-Level Design. This development includes Interaction Diagrams (diagrams detailing how each component will interact with each other component) and a UML Detailed Class Diagram (a diagram that details every class in the Scheduler, along with how every other class connects to it).

Lastly, this milestone also required the development of a Traceability Matrix, used to trace everything detailed in the High-Level Design to the Specifications Document, and eventually back to the Requirements Document.

A presentation is also required for this milestone. The presentation should detail all changed made to the Specifications Document, the Software Project Management Plan, and the High-Level Design.

### 1.2.6 Milestone 4

For the next milestone, the Specifications Document was updated to accommodate all changes made. This milestone marks the end of all updates to the Specifications Document without the use of a Change Request Form.

The SPMP was also updated to reflect the changes in the project, work that has been done, work that needed to be done, and to keep track of everything currently in the project.

The High-Level Design was also updated to reflect any modifications to the diagrams created. This ensures that all of the designs are correct, which reflect the previous documents and will be used to develop the rest of the Scheduler.

The Detailed Design begins with this milestone, the Detailed Design is implementing the Scheduler in Program Description Language (PDL), which is a non-compilable, human-readable document, detailing exactly what and how each component of the Scheduler will work.

The Traceability Matrix was updated to trace the components in the PDL back to the Specifications Document, and back even further to the Requirements Document.

A presentation must also be made for this milestone, detailing any changes made to the Specifications Document, the Software Project Management Plan, and the High-Level Design. The presentation must also cover the Program Description Language designed in this milestone.

### 1.2.7 Milestone 5

The SPMP is to be updated for milestone 5 to reflect the most recent breakdown of tasks for the project, as well as updating and making sure the schedule is on track for development of the Scheduler.

The next document that requires updating is the Design, the Interaction Diagram, UML Class Diagrams, and Program Description Language are to have development continued and updated throughout this milestone. This milestone is the last milestone for development on these documents, the design of the Scheduler shall be completed by the end of this milestone.

The next item to begin development on is the actual implementation of the Scheduler. Development upon the source code, written in Java, begins in this milestone. Roughly half of the project shall be completed and in working source code for this milestone. Development of the source code comes almost directly from the Program Description Language that was developed in the previous milestone.

The Traceability Matrix is also to be updated, the Traceability Matrix is updated to reflect the design and implementation, and trace them back to the Specifications and Requirements Document.

A presentation must be made for this milestone, detailing any changes made to the Software Project Management Plan, the Program Description Language, and any other changed documentation. The presentation must also cover the portions of the Scheduler that have been implemented, and give a demonstration of the currently implemented components.

### 1.2.8 Milestone 6

The very last milestone of the project, this is where all of the work from the previous milestones converges. This milestone will have the complete program, along with all documentation for the program turned in on a CD or DVD.

The Requirements Document must be turned in, and any requirements listed but not implemented must be clearly labelled.

The Acceptance Test Plan must also be turned in, along with any requirements not met clearly labelled.

The specifications must also be turned in, along with Object-Oriented Analysis for the project, which includes: A complete set of scenarios that covers all re-

quirements listed in the Requirements Document that have been implemented, Class modelling from all stages previously made, and Dynamic Modelling previously made.

The Software Project Management Plan must be updated to reflect all work performed on the project, along with when every task was completed, dependencies, and the complete schedule for project completion.

The Object-Oriented Design, including all High-Level Design diagrams such as UML interaction diagrams, and UML detailed class diagrams, and the Program Description Language are all to be turned in.

All source code developed during the project for the Scheduler is to be completed and included with the product delivery.

Implementations Testing is to be done, including a black-box test case for one subordinate use case listed in the Requirements Document, and a glass-box test case for the most complex module developed and included in the source code turned in as part of this milestone. Details about the test cases shall be provided, including the tested unit, how test data was found, actual input, expected output, state information, who the tester was, and any discrepancies between expected output and actual output during earlier test runs.

A final traceability matrix shall be produced to show exactly how everything in the implementation traces back to other documents and, ultimately, to the Requirements Document.

Screen Shots of every GUI window that is launched by the Scheduler shall be provided, along with example, professional, appropriate data entered to demonstrate what the GUI will look like in action. Each screenshot shall have a description along with it that details how the Scheduler will go about launching the GUI, along with the purpose of the GUI.

A detailed User's Manual will also be turned in, the User's Manual shall include, at a minimum: Cover Page, Table of Contents, System Introduction, System Set Up, High-Level View of System, Tutorial, Detailed View of System, Index, and Appendices.

This milestone shall also be accompanies by a presentation. This presentation shall include the final work breakdown of all tasks, the implementation testing, to give a demonstration of the code, black-box testing, and glass-box testing. The presentation shall also include demo runs of the Scheduler, demoing the run of the Scheduler over the scenarios given in the Acceptance Test Plan.

# 2 System Set Up

This section shall detail the explicit steps required in order to first install, then run the Scheduler.

## 2.1 Installation

### 2.1.1 Pre-installation Check

The latest version of the Java Runtime Environment (JRE) must be install upon the system in which you wish to execute the Scheduler. As of this writing, the latest version of Java is version 6 update 29. In order to obtain the latest version of Java, please visit `http://www.java.com/getjava/` and follow the on-screen instructions to download the latest version of JRE.

### 2.1.2 Obtaining The Executable

You must obtain the ZIP archive from whatever the agreed upon means of delivery were. For the purposes of this document, we will assume that the agreed method of delivery was an e-mail message. Upon receiving the ZIP archive, you must download the archive (Figure 1).

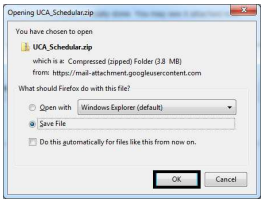



Figure 1: Downloading the file from the agreed means of transmission

### 2.1.3 Installing The Scheduler

Once the download has completed, you must extract the file to a folder (Figure 2). Once you click "Extract All" (shown in the figure), follow the on-screen wizard, and it should extract the files into a new folder labelled UCA_Scheduler, located in the same directory as the ZIP archive itself. Once the files have completed extracting, the installation is done.

## 2.2 First Use

This section will detail the steps required to begin the execution of the Scheduler.

### 2.2.1 Opening The Executable

In order to run the program, you must navigate to the location in which you extracted the files to, then you must double-click the file labelled "UCA_Scheduler.jar" to launch the program (Figure 3). The program was written in, and needs the Java Runtime Environment (JRE) in order to execute, so make sure you have followed the steps detailed in the Installation section before attempting to launch the program. Once the program is launched, it should load in the JRE. The execution of the program will look similar to that shown in Figure 4.

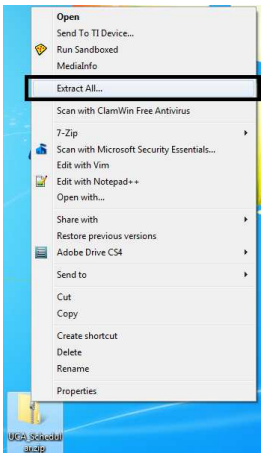



Figure 2: Extract the ZIP archive to a new folder.

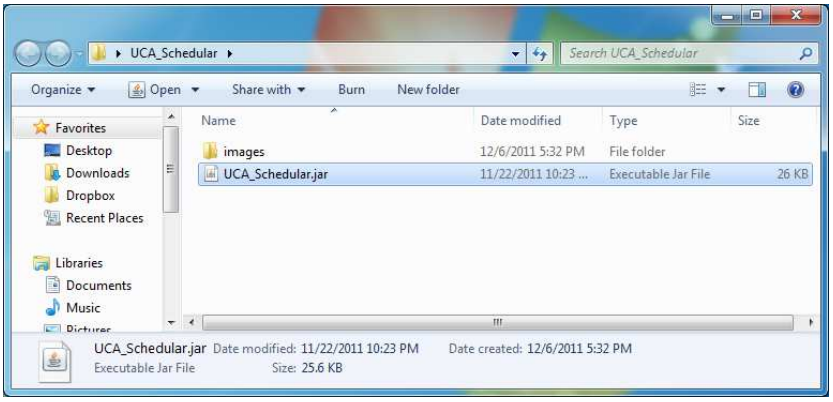



Figure 3: Double click the executable JAR file in order to run the program

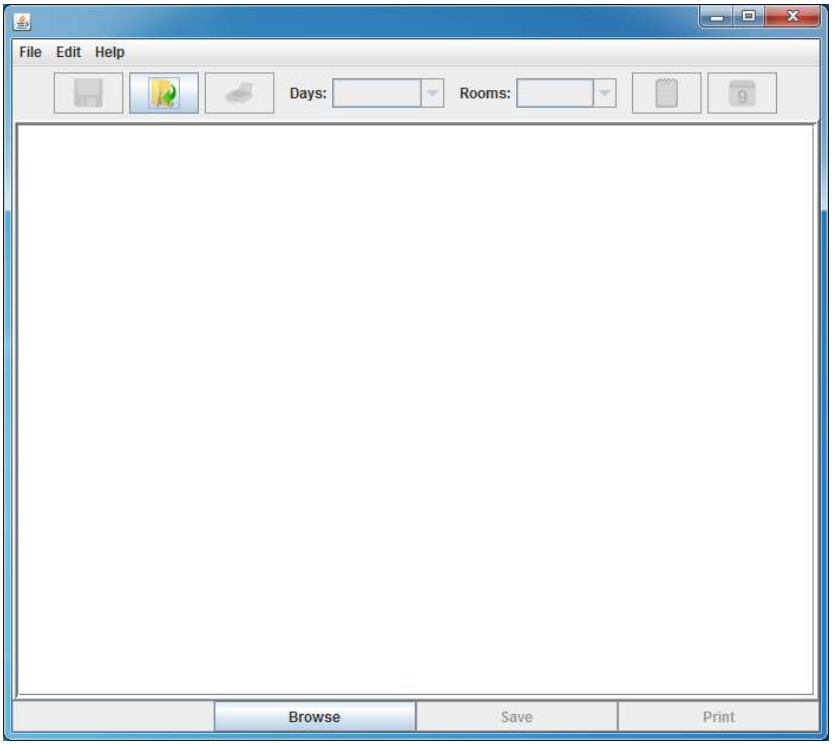



Figure 4: Program just after launch

# 3 High-Level View of System

## 3.1 Overview

The Scheduler can be simplified to to this relatively
simplistic view of all of the classes and how they interact (Figure 5). A much more detailed view will be presented later in this document.

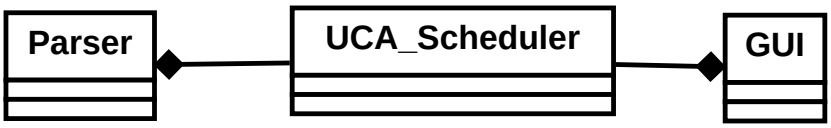



Figure 5: Simplified UML

## 3.2 How It Works: The Simple Explanation

Basically, when the Scheduler is executed, the UCA_Scheduler class will start and create instances of all of the other classes that it interacts with directly. It will then create an instance of the GUI that the user can interact with. Once the GUI receives a command from the user (such as, to parse a file), the GUI will pass the message along to the UCA_Scheduler class, which will in turn pass the message along to the class it needs to go to.

### 3.2.1 UCA_Scheduler

The UCA_Scheduler class is used as a main controller for the program. It will control all of the actions that go on, from opening the program, parsing a file, and closing the program. Upon opening the program, this class creates instances of the GUI class, and the Parser class.

### 3.2.2 GUI

The GUI class is the main, and only, interaction between the user and the program. Upon opening the program, the GUI will wait for the user to enter a command, such as choosing a file and parsing it. Once the user gives the program a command, the command will be send to the UCA_Scheduler class to determine what needs to be done with it.

### 3.2.3 Parser

The Parser class is where the majority of the work will be done. Once the UCA_Scheduler class tells the Parser to parse a specific file, it will go and process the entire file, and return an output back to UCA_Scheduler, the output will either be an error message, or a parsed file. Whatever the output is, it will be sent back to the GUI to inform the user.

# 4 Tutorial

## 4.1 Example 1: HTML File Chooser/Parsing File

This is an example to demonstrate how to choose an HTML file to load into the program.

### 4.1.1 Step 1: Open The Program

The first thing you want to do is open the program, this can be done by double-clicking the executable JAR file that was created upon following the instructions outlined in the Installation section (See Figure 6). The running program should look like that shown in Figure 7.

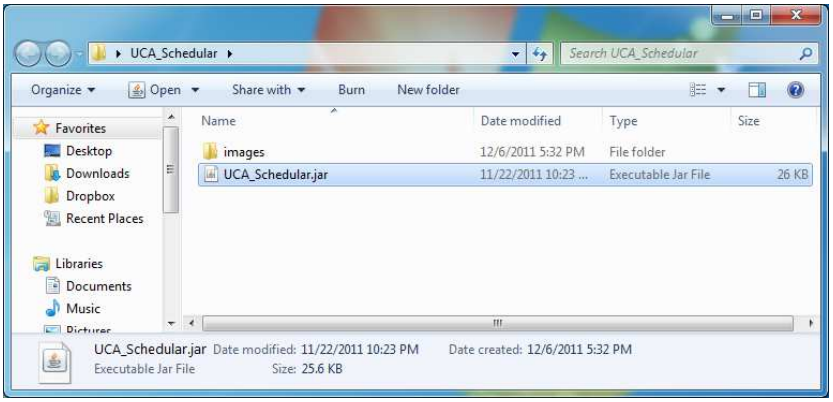



Figure 6: Executing the program

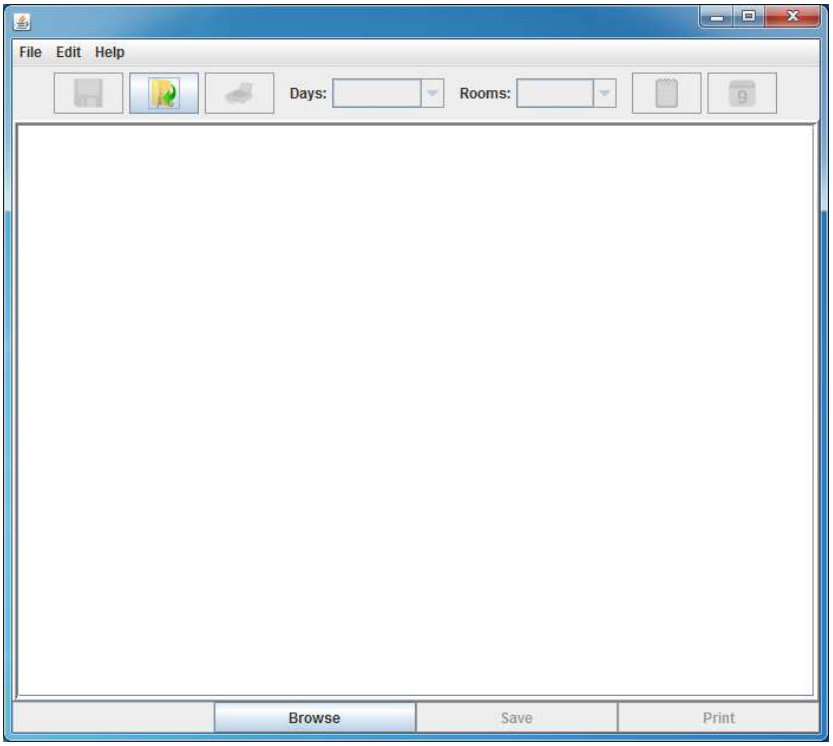



Figure 7: Initial launch of the program

#### 4.1.2 Step 2: Browsing for a file

Next, you want to select the button that looks like a folder opening, this is the "Open" button, and is one of three ways in which to browse for a file (See Figure 8). Once the open button is pressed, browse to the folder in which your iCal HTML file is saved and select that (See Figure 9). Once the iCal file has been selected, click the Open button (See Figure 9).

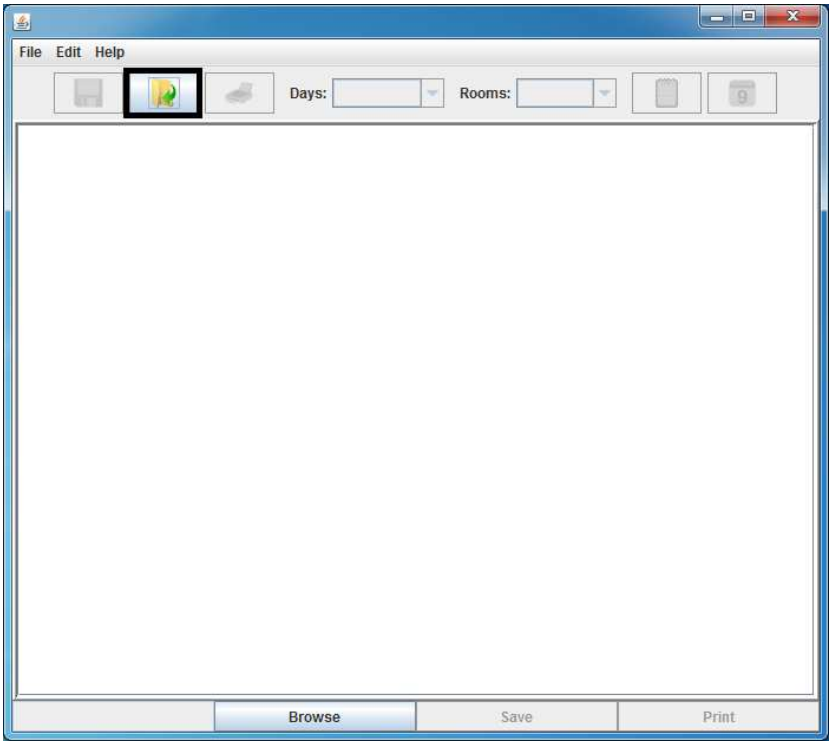



Figure 8: Press the open button to open the file selection dialog

At this point, the selected file shall be parsed, and the program shall display the parsed file (See Figure 10).

### 4.2 Example 2: Saving A File

This section describe the process needed in order to save a parsed file to a TXT file.

#### 4.2.1 Step 1: Parsing A File

Follow the steps outlined in Example 1 to open and parse a file.

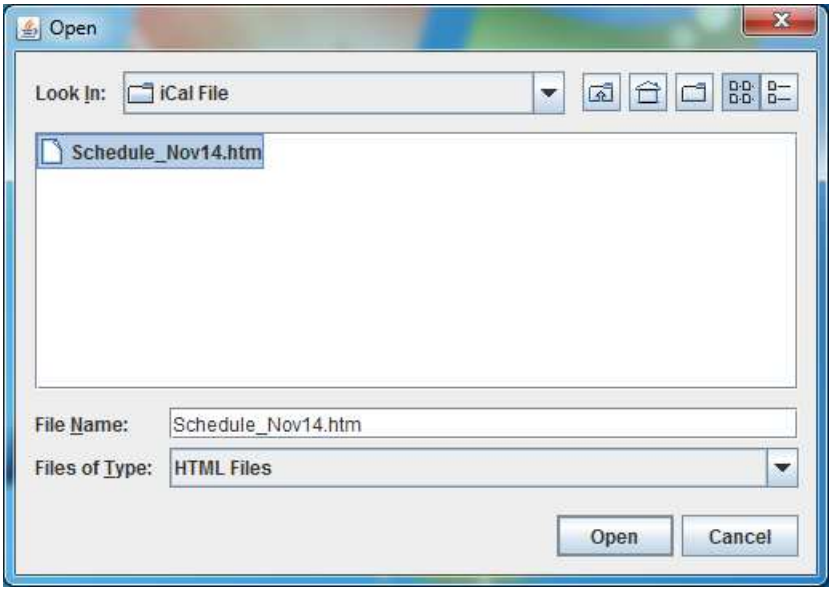



Figure 9: Find an iCal file

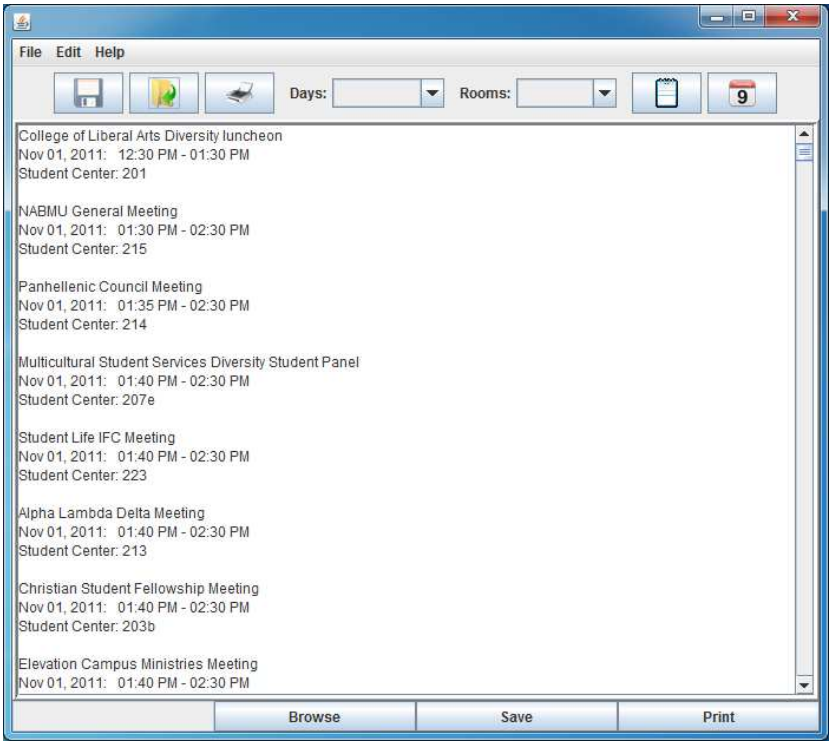



Figure 10: Parsed Output In Program

### 4.2.2 Step 2: Saving The File

First, Select the "Save" button from the menu bar (See Figure 11).

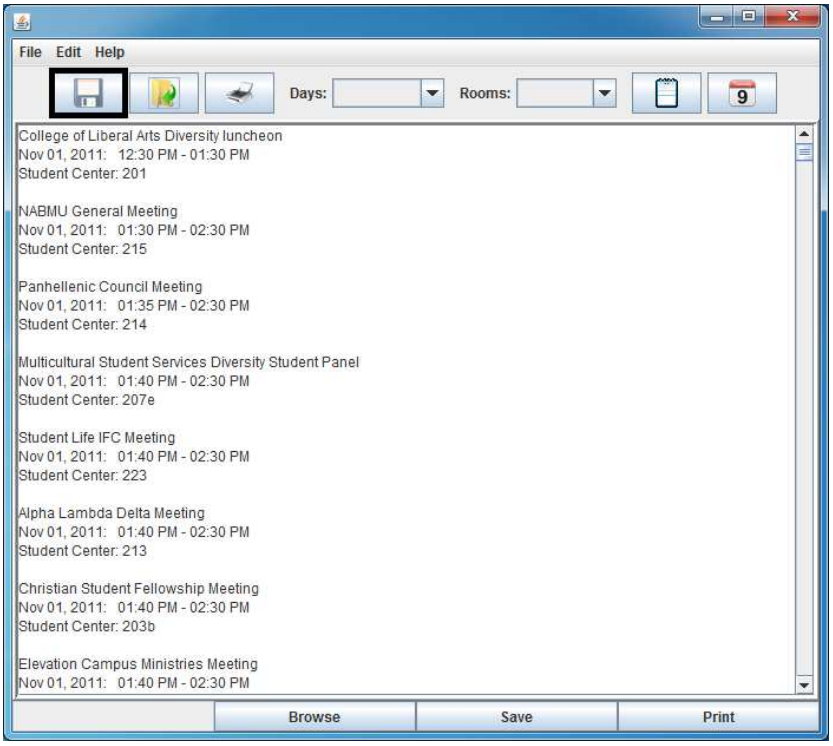



Figure 11: Press the save button to save the parsed file

Next, select the location to save the file to (See Figure 12).

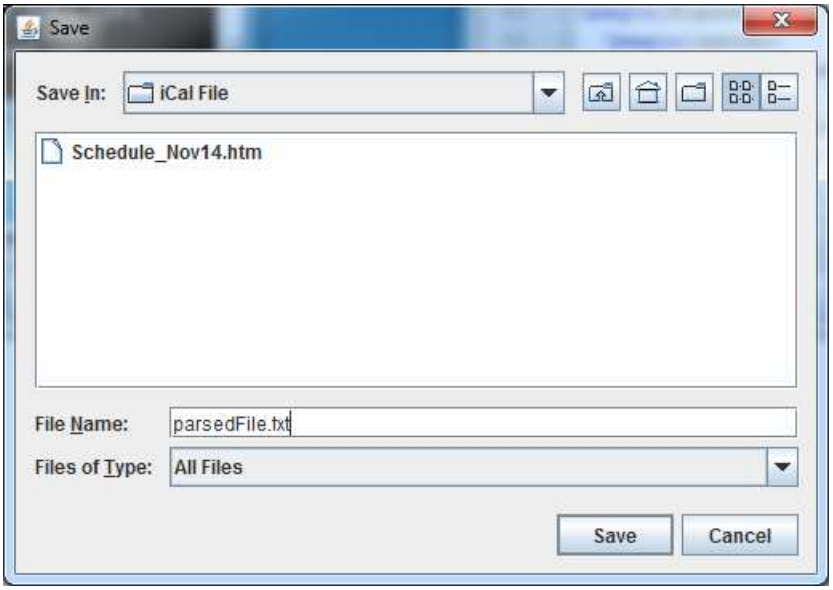



Figure 12: Navigate to a file location

Finally, press the "Save" button to save the file (See Figure 13).

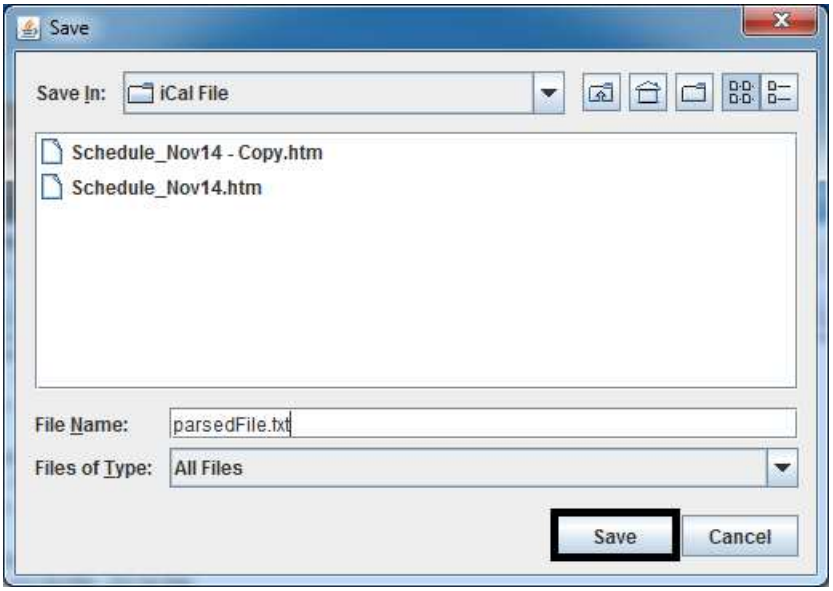



Figure 13: Save the file

The program will then write the output generated to the specified location.

## 4.3 Example 3: Printing A File

### 4.3.1 Step 1: Parsing A File

Follow the directions outlined in Example 1 for instructions on how to parse a file.

### 4.3.2 Step 2: Printing A File

Once the file has been parsed, click the "Print" button at the top, near the Open and Save buttons (See Figure 14).

Once the print button has been pressed, you will see a Print dialog window open, select the printer and choose Ok to print the file (See Figure 15).

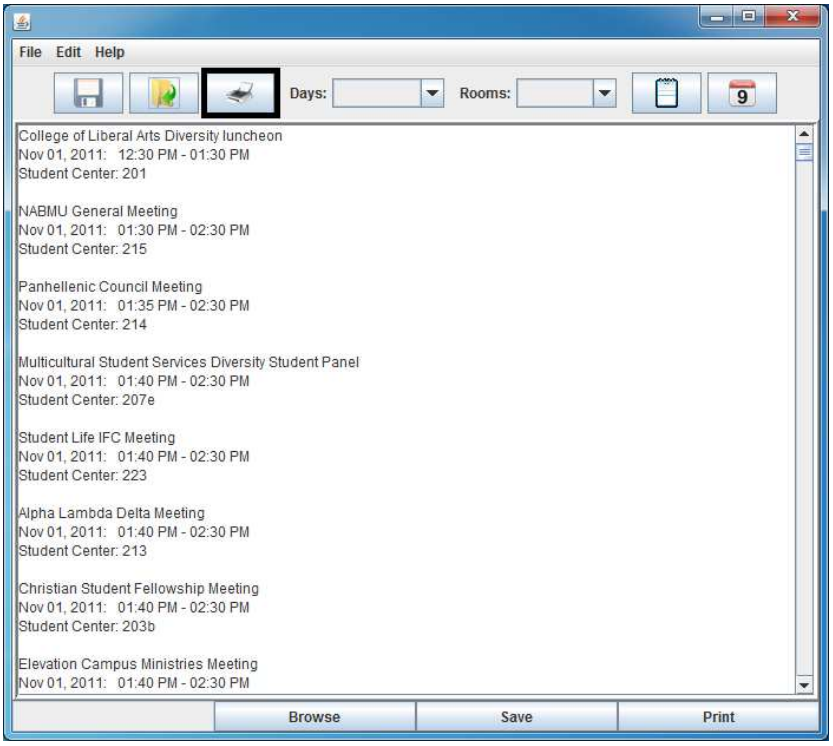



Figure 14: Print button

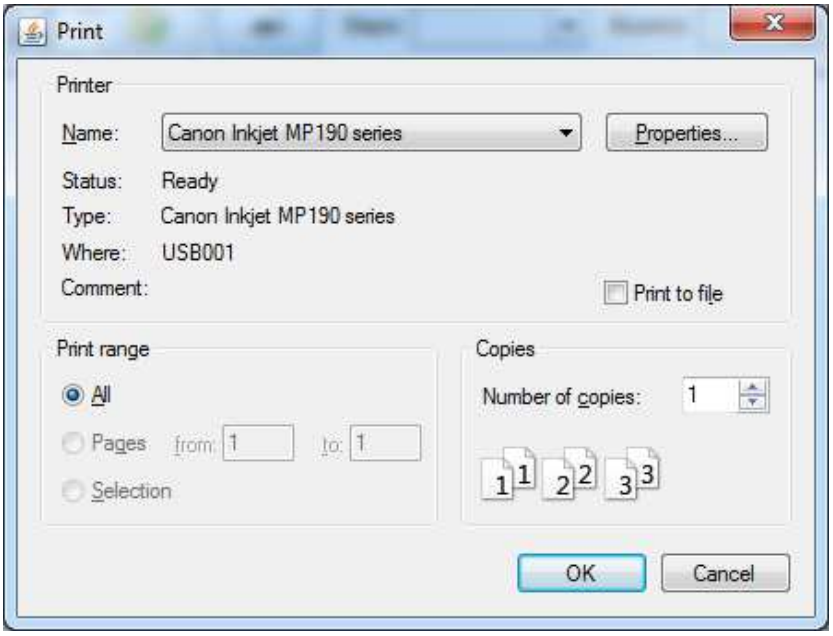
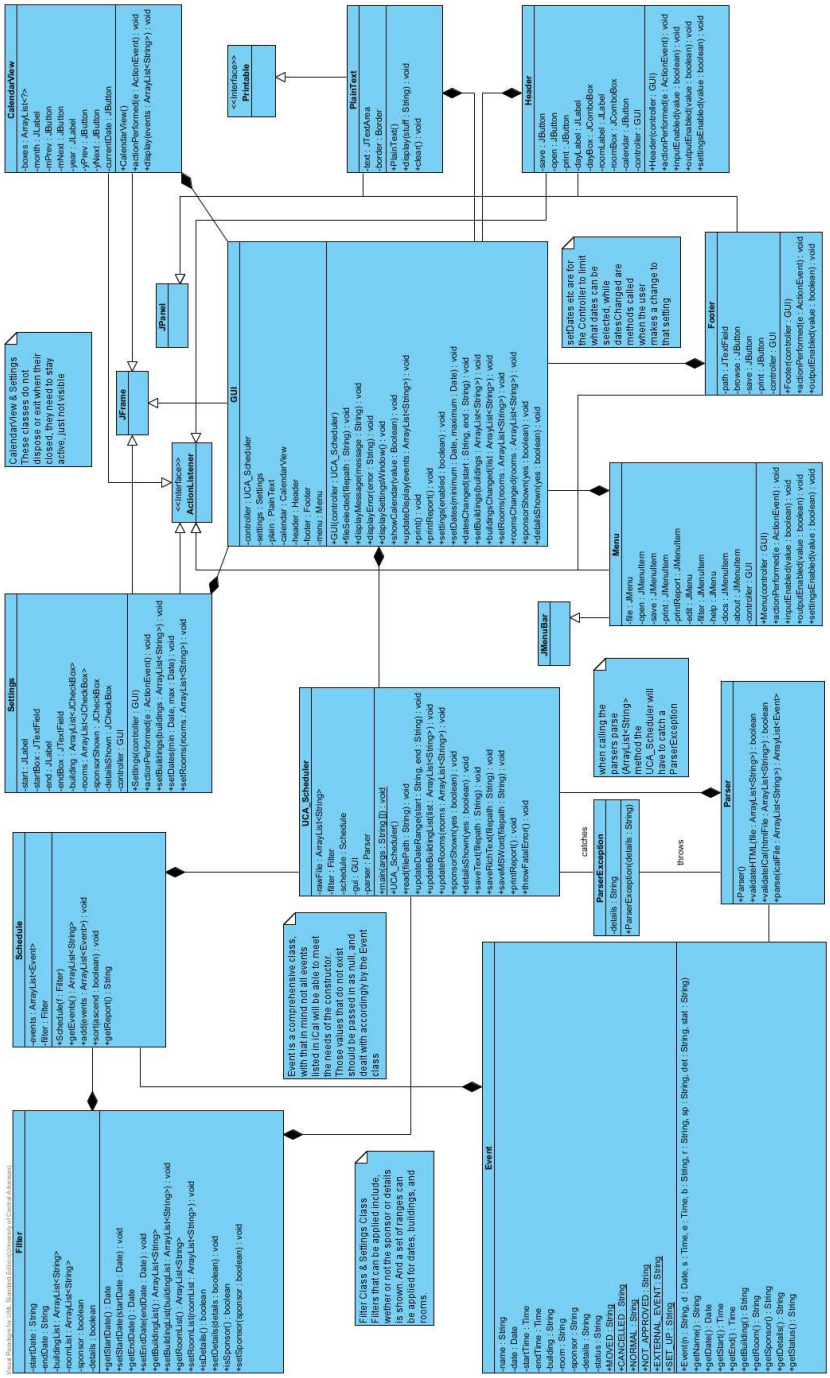



Figure 15: Print The File

# 5 Detailed View of System

## 5.1 Detailed UML



## 5.2 UCA_Scheduler

The UCA_Scheduler class is the main driving class of the entire program. This class is the central hub of the whole program, directing all messages from the GUI to the underlying classes that implement the majority of the work.

### 5.2.1 main

The main method of the UCA_Scheduler class is the main entry point of the whole program. The only task it has is to declare and initialize a UCA_Scheduler object.

### 5.2.2 UCA_Scheduler

The UCA_Scheduler function is the constructor for the UCA_Scheduler class. It initializes and assigns values for all of the private members of the class. By initializing a GUI object, it also opens a GUI window for the user to interact with.

### 5.2.3 read

Given a file path, the read method will read the file and store the contents in a variable. It will then call methods to verify that it is, in fact, a valid HTML iCal file. If it is not, it will tell the GUI to display an error message.

### 5.2.4 updateDateRange

Given a start and end date, it will update the filter's settings for the start and end dates specified.

### 5.2.5 updateBuildingList

Given an ArrayList of strings, it will update the list of buildings stored in the filter.

### 5.2.6 updateRooms

Given an ArrayList of rooms, this method will update the list of rooms that are stored in the filter.

### 5.2.7 sponsorShown

Sets the filter to display the sponser.

#### 5.2.8 detailsShown

Sets the filter to display the details.

#### 5.2.9 saveText

Given a file path to save a schedule to, this method will open the output file specified and write the parsed data to that specified file, in a plain text format.

#### 5.2.10 saveRichText

This optional method will, given an output file path, open the path and write parsed schedule to the output in Rich Text format.

#### 5.2.11 saveMSWord

This optional method will, given an output file path, open the path and write the parsed schedule to the output in Microsoft Word format.

#### 5.2.12 printReport

This method will print a report of details about specific events, such as the number of events that have been moved, cancelled, etc.

#### 5.2.13 throwFatalError

This method is to throw an error message if something goes wrong that the program itself cannot recover from.

### 5.3 ParserException

This class is used to communicate from the Parser, to the UCA_Scheduler class that an error has occured and cannot be recovered from.

#### 5.3.1 ParserException

This is the method used to pass the details of the error that occured in the Parser, back to the UCA_Scheduler class.

## 5.4 Parser

This is the main functionality class of the program, this class manages all of the processing of information from the input HTML file. When called to process an HTML file, it will either return the resulting schedule information, or it will cause an error to occur, informing the user that there is something wrong and it is unable to process the specified HTML file.

### 5.4.1 Parser

The constructor for the Parser class does nothing.

### 5.4.2 validateHTML

This method will walk through an HTML file and determine if it is a valid HTML file or not. Because iCal does some funky stuff with their HTML tags, such as not closing them in reverse order of opening, or not closing at all, the validateHTML has a tolerance for the number of errors that may occur with the tags in an HTML file.

### 5.4.3 validateICal

This will, given a file, check to see if it contains the correct header information that every iCal file should have. If it does contain the correct header data, it will return that the file is valid. If the header data cannot be found in the file, it will return that the file is not a valid iCal file.

### 5.4.4 parse

The parse method of the Parse class is where the majority of the work happens. This method will step through an iCal file that has already been validated and remove all of the details from the file, such as room, building, name, status, etc. of each event, and store all of the parsed events.

## 5.5 Schedule

The Schedule class is used to store and organize events.

### 5.5.1 Schedule

The constructor for the Schedule class simply takes the argument passed to it, a Filter object, and sets that object as the current private filter object, allowing for the filters to be used in the Schedule class.

#### 5.5.2 getEvents

The getEvents method is used to simply get all of the events stored in the Schedule class and return them, parsing out everything filtered by the Filter class.

#### 5.5.3 add

Adds an event to the ArrayList of events stored in the Schedule class.

#### 5.5.4 sort

Sorts all events in either ascending or descending order of the date of the event.

#### 5.5.5 getReport

Returns a report generated by the events stored in the Schedule class. The report contains useful information to the user, such as the number of events cancelled.

### 5.6 Event

The Event class is used to hold the details about a specific event. Each event can be, and will be, represented as an individual event in the Event class.

#### 5.6.1 Event

The event constructor acts as the setter method. It assigns values to every field in the event, the Name, Date, Start, End, Building, Room, Sponsor, Details, and Status.

#### 5.6.2 getName

Returns the specific name of the event.

#### 5.6.3 getDate

Returns the date of this event.

#### 5.6.4 getStart

Returns the start time of this event.

#### 5.6.5 getEnd

Returns the end time of this event.

#### 5.6.6 getBuilding

Returns the building this event is taking place in.

#### 5.6.7 getRoom

Returns the room this event is taking place in.

#### 5.6.8 getSponsor

Returns the person that is sponsoring this event.

#### 5.6.9 getDetails

Returns the details about this event.

#### 5.6.10 getStatus

Returns the status of this event.

### 5.7 Filter

Filter Class

#### 5.7.1 getStartDate

Returns the start date of the filter.

#### 5.7.2 setStartDate

Assigns a new start date to the filter.

#### 5.7.3 getEndDate

Returns the end date of the filter.

#### 5.7.4 setEndDate

Assigns a new end date to the filter.

#### 5.7.5 getBuildingList

Returns a list of all of the buildings in the filter.

#### 5.7.6 setBuildingList

Assigns a new list of buildings to the filter.

#### 5.7.7 getRoomList

Returns the list of rooms in the filter.

#### 5.7.8 setRoomList

Assigns a new list of rooms to the filter.

#### 5.7.9 isDetails

Returns if the details should be displayed or not.

#### 5.7.10 setDetails

Assigns weather the details should be displayed or not.

#### 5.7.11 isSponsor

Returns if the sponsor should be displayed or not.

#### 5.7.12 setSponsor

Assigns a new option of if the sponsor should be displayed or not.

### 5.8 GUI

The GUI class is the main driving class of the portion of the program that interacts with the user. This class will create instances of the other components and add them to the GUI, forming one single GUI.

#### 5.8.1 GUI

The constructor for the GUI class will initialize all of the instances of the other classes that contribute to the GUI. It will then build a GUI and display it.

#### 5.8.2 fileSelected

Checks for the extension of a file passed to it, if it is HTM or HTML, it will attempt to call the parser with that file. If the extension is TXT, RTF, or DOC, it will attempt to save the data stored in the parser to that file. Otherwise, it will display an error message.

#### 5.8.3 displayMessage

Displays a simple message to the user.

#### 5.8.4 displayError

Displays an error message to the user.

#### 5.8.5 displaySettingsWindow

Displays the settings window.

#### 5.8.6 showCalendar

Changes the input mode from Plain Text to Calendar, or from Calendar to Plain Text.

#### 5.8.7 updateDisplay

This will clear the displays and display the most recently stored events.

#### 5.8.8 print

Initializes a new PrinterJob and sends the TXT version of the parsed iCal file to the printer.

#### 5.8.9 printReport

Prints a copy of the report rather than the events.

#### 5.8.10 settings

Sets all of the input/output/settings to either enabled or disabled, dependent upon how the function was called.

#### 5.8.11 setDates

Dates have been set from the parsing of an HTML file.

#### 5.8.12 datesChanged

Dates displayed have been changed from the user.

#### 5.8.13 setBuildings

Buildings have been set from the parsing of an HTML iCal file.

#### 5.8.14 buildingsChanged

Buildings to display have been changed by the user.

#### 5.8.15 setRooms

Rooms have been set from the parsing of an HTML iCal file.

#### 5.8.16 roomsChanged

Rooms to display have been changed by the user.

#### 5.8.17 sponsorShown

Sets the filter to either show or hide the sponsor.

#### 5.8.18 detailsShown

Sets the filter to either show or hide the details.

### 5.9 Settings

The Settings class is used to create a Settings window for the user to select/modify settings in the program such as what dates should be parsed, the rooms that should be parsed, etc.

#### 5.9.1 Settings

The constructor initializes the Settings window.

#### 5.9.2 actionPerformed

Detects when a button is pressed, and when it is, sends the changed settings back to the GUI, which then delivers the settings changes to all other components that need them.

#### 5.9.3 setBuildings

Allows for the buildings selected to be set in the Settings window.

#### 5.9.4 setDates

Allows for the dates selected to be set in the Settings window.

#### 5.9.5 setRooms

Allows for the rooms selected to be set in the Settings window.

### 5.10 Menu

The Menu class is used for creating and displaying a JMenuBar at the top of the GUI window.

#### 5.10.1 Menu

The constructor of the Menu class is used to create instances of all of the components in the menu bar, then to add them to the Menu class as components.

#### 5.10.2 actionPerformed

Upon an event taking place in the Menu class, the corresponding method is called in the GUI class by the actionPerformed method.

#### 5.10.3 inputEnabled

This allows all of the input functionality of the Menu class to be either enabled or disabled, dependant upon how the function was called.

#### 5.10.4 outputEnabled

This allows for all of the output functionality to be either enabled or disabled, dependant upon how the function was called.

#### 5.10.5 settingsEnabled

This allows for all of the settings functionality to be either enabled or disabled, dependant upon how the function was called.

### 5.11 Footer

The Footer class is used to display the options at the very bottom of the GUI, such as the option to Save and Print.

#### 5.11.1 Footer

The constructor for the Footer class is used to initialize all of the private variables, and to add the buttons to the footers own GUI, to be used by the GUI class.

#### 5.11.2 actionPerformed

The actionPerformed method is used to handle all events that take place in the Footer, and redirect the function calls to the GUI class.

#### 5.11.3 outputEnabled

The outputEnabled function is used to either enable or disable the functionality of the buttons stored in the Footer class.

### 5.12 CalendarView

Creates a display for a calendar to be shown to the user, so the user has the option of displaying events in the form of a calendar rather than simply a standard text display.

#### 5.12.1 CalendarView

The constructor of the calendar view, this constructor creates the calendar by creating a JPanel and adding all of the required components to it.

#### 5.12.2 actionPerformed

Detects when a button is pressed in the calendar, and changes the display to accommodate the new request by the user.

### 5.12.3 display

Given an ArrayList of strings, the calendar should reflect that ArrayList, in the form of a calendar display.

## 5.13 PlainText

This class is used, as part of the GUI class, to display the parsed output as plain text to the user, inside of the program before the text is saved as output.

### 5.13.1 PlainText

This is the constructor for the PlainText class, it simply initializes values and creates the GUI components of the PlainText class for use by the GUI.

### 5.13.2 display

Given a String, this method will append the string to the displayed text in the program.

### 5.13.3 clear

This method will clear all text from the display.

## 5.14 Header

The Header class is used to control and display the quick shortcut buttons displayed at the top of the GUI. Such items as Open, Save, Print, switching views, and quickly selecting a room.

### 5.14.1 Header

The constructor for the Header class simply loads all of the required images into the program, and creates/places all of the required buttons in a JPanel.

### 5.14.2 actionPerformed

The actionPerformed method handles when an event takes place with one of the buttons in the Header, and from there, will call the correct GUI function to process each command.

### 5.14.3 inputEnabled

The inputEnabled function will, when called, set all of the buttons used for gathering input from the user to enabled or disabled depending on what parameter was passed to it.

### 5.14.4 outputEnabled

The outputEnabled will, when called, enable or disable all of the output functions, dependant upon the parameter that was passed to it.

### 5.14.5 settingsEnabled

The settingsEnabled will, when called, enable or disable all of the settings functions, dependant upon the parameter that was passed to it.

# Index

# A Quick Reference Card

| | **Open A File** | **Parse A File** | **Save A File** | **Print A File** | **Modify Settings** |
|---|---|---|---|---|---|
| Step 1 | Begin the program by running the executable JAR file | Follow the directions specified in "**Open A File**" | Follow the directions specified in "**Parse A File**" | Follow the directions specified in "**Parse A File**" | Begin the program by running the executable JAR file |
| Step 2 | Press the "Open" button (icon that looks similar to a folder opening | The file is now displayed in the program as a parsed text output | Press the "Save" button in the program (icon that looks similar to a floppy disk) | Press the "Print" button in the program (icon that looks similar to a printer) | Select Edit from the menu bar at the top of the program |
| Step 3 | Navigate to the directory in which the file is stored | | Navigate to the directory in which the file is to be saved | Select the desired printer | Select Settings from the options under the Edit section |
| Step 4 | Select the file to open | | Type in the desired file name | Select the desired number of pages to print | Modify the settings as desired |
| Step 5 | Press the button that says "Open" | | Press the "Save" button | Press the "OK" button | Press the "Save" button to save the settings |

# B System Requirements

| OS | Version | Memory | Disk Space |
|---|---|---|---|
| Windows | Windows 2000 | 128 MB | 98 MB |
| Unix | Mac OS X | 64 MB | 58 MB |

# C Installation

## C.1 Step 1: Obtain the file

You should obtain the ZIP archive from the agreed upon source of delivery, for example, by email (See Figure 1 on page 6).

## C.2 Step 2: Extract the files

You should navigate to the directory in which you downloaded the ZIP archive specified in Step 1, and extract the contents of it (See Figure 2 on page 6). Follow the on screen instructors for extracting the file.

## C.3 Step 3: Executing the program

Once the files have completed extraction, navigate to the folder in which the files were extracted to and execute the JAR file (See Figure 3 on page 7). You should see a window that looks like that displayed in Figure 4 on page 7.

The program is now installed and ready for use.

# D Troubleshooting

## D.1 Error: Not A Valid HTML File

Ensure that the file you specified to the program is, in fact, an HTML file generated by the iCal program. If it is, try re-downloading the file, for it may have been corrupted.

## D.2 Error: Not A Valid iCal File

Ensure that the file you provided to the program is, in fact, an HTML file generated by the iCal program. If it is, try re-downloading the file, for it may have been corrupted.

# E Maintenance Procedures and Issues

If maintenance is required for this program, see about contacting Professor Paul Young at the University of Central Arkansas (pyoung@uca.edu), the project manager Tejaswi Yellipeddi (tejaswi.y@gmail.com), or one of the developers in the developing team.

However, this product is developed as part of a class-wide group project, and as such, after the semester ends (as of December 17th 2011), development and maintenance for this product will no longer officially exist, however, if problems do arise, see about contacting one of the aforementioned individuals.

# F Developers' Information

| **Name** | **Email** |
|---|---|
| Tejaswi Yellipeddi | tejaswi.y@gmail.com |
| Aaron Crawford | aaroncrawford84@gmail.com |
| Alex Loney | loney.alex@gmail.com |
| Bryan Schardt | brschardt@gmail.com |
| Minwoo Kim | song_of_the_rain@msn.com |
| Andrew Lorigan | andrewlorigan1@gmail.com |
| Mark Vander Lugt | mvanderlugt@live.com |
| Kellye Young | 8bit.space.romantic@gmail.com |
| Eric Barton | ericdbarton@gmail.com |
| Nathan Drake | nathan@drakedev.com |
| Edward Dobbins | gama_code999@yahoo.com |
| Daniel Macha | danielmacha@yahoo.com |
| Cody Hudson | codyhudson67@gmail.com |
| Joel Doyle | joledoyle@gmail.com |