Team Admin

Student Reports and Statistics

(SRAS)

Software Design Document

Prepared by:
Mai Nguyen
Whitner Reichman
Judy Gonzalez

Release Date
March 12, 2023

Table of Contents

Executive Summary

Student Reports and Statistics (SRAS) is an anonymous reporting and tracking system designed for college students. SRAS aims to provide transparent information and report tracking for students that is lacking in the annual safety reports conducted by Campus Safety, as well as the anonymous (and non-anonymous) support systems in place at the Title IX office. With SRAS, students can file a report anonymously, access school-wide data, and re-access every report they have submitted.

This document provides technical information regarding the software design of SRAS.

Document Versioning

Date	Owner	Comment
03/05/2023	Mai Nguyen	Feature discussion M.3, G.1, and G.2, high-level architecture diagram, major components and detail design for userInput, FileReport, AccessReport, and Report
03/05/2023	Kyle Moriarty	Feature discussion P.1 and P.2
03/05/2023	Judy Gonzalez	Feature discussion P.3 and L.1, major components and detail design for Stats, Filter, and MoreInfo
03/05/2023	Whitner Reichman	Initial text for rough draft, feature discussion L.2, M.1, and M.2, major components and detail design for Home, ManageReports, and Admin
03/12/2023	Mai Nguyen	Updated High-level Architecture Diagram, detail design for UserInput, FileReport, Home, ReportsManager, and AccessReport, crossed out Admin component
03/12/2023	Judy Gonzalez	Updated Stats component UML diagram and description, updated MoreInfo component UML diagram, and crossed out Filtered Search in the feature discussion and the Filter component

Feature Matrix

The feature matrix enumerates the technical features required to support each of the business requirements. The discussion section provides details regarding the constraints and functionality of the feature. The ids are used for traceability. Features that can be removed should strike-through the feature id and have a comment added to identify why this feature can be removed without impacting the BRD requested functionality.

Priority Codes:

H - High, a must have feature for the product to be viable and must be present for launch
M - Medium, a strongly desirable feature but product could launch without
L - Low, a feature that could be dropped if needed

ID	Pri	Feature Name	Comment	BRD ID
P.1	H	CLI		C.1, D.2, E.1
P.2	M	Action (submit, cancel, return home, etc.)		C.1, D.2, D.3, D.4, D.10, D.11
P.3	H	Language	Implementation in Java	C.1, C.2, C.3, T.1, D.1, D.2, D.3, D.4, D.5, D.6, D.7, D.8, D.9, D.10, D.11, D.12, D.13, E.1, E.2, E.3, E.4
L.1	L	Error Handling		E.1
L.2	M	Save to Database		D.1, D.2, D.3, D.5, D.6, D.7, D.8, D.9, D.10, D.12, E.2
M.1	M	View Database		D.3
M.2	M	Remove From Database		D.4
M.3	H	Generate ID		E.3
G.1	H	Generate Report		D.2, E.3, D.1, D.5, D.6, D.7, D.8 D.9, D.13, D.14
G.2	L	~~Filtered Search~~	Feature not implemented	E.4, D.5, D.6

Feature Discussion

P.1 - CLI

SRAS' anonymous reporting system has simple user interactions which can be easily operated in a Command Line Interface. SRAS provides a CLI for filing reports through a command line terminal.

P.2 - Action

The program must support different actions taken within the Command Line Interface, such as going to the next page, canceling a report, returning home, and submitting a report. These actions are available to the user by Command Line prompts so that they may use the application smoothly. Some actions, such as Admin access, will be password protected.When an Admin types in the password, more action prompts will be presented to them.

P.3 - Language

SRAS' must be usable with a number of different devices. Therefore, our group has decided to use JAVA instead of Python, JavaScript, C++, etc. because it has cross-platform support, meaning that it is able to move easily from one computer system to another once the code is written. This supports CLI interaction and consistent GUI behavior across different platforms (i.e. Windows, Mac, and Linux).

L.1 - Error Handling

If a user inputs an unknown response when filling out a report or filtering through statistics, an error message will be displayed to inform the user that an error has occurred. The user is then given a message to try again with one of the available input options.

L.2 - Save to Database

An action initiated by the 'submit' option, that reads the user's answers and saves them in a CSV file with width constant '*n*', leaving blanks where fields are empty. Once we learn databases in class, this will replace the CSV storage system.

M.1 - View Database

A feature only accessible via the 'Admin' feature and command. When called it should print the Database (or CSV in the meantime) of reports up to the most recent entry, with all fields of text and multiple choice included.

M.2 - Remove from Database

Using a call that references the 'report ID', the 'Remove from Database' feature allows for the user to delete a report if they decide to. It will be deleted from the CSV file.

M.3 - Generate ID

When a user submits a report, the program will generate a unique ID for that report. The ID may consist of a combination of numbers. The ID will be associated with the report in the database system, along with the other relevant information such as the date, mental health impact, graduating class, and other optional information. To re-access a report, the user can select the "re-access report" option from the main menu, and then enter the ID when prompted. The program will search for the report with the corresponding ID in the database or file system, and then display the report information to the user.

G.1 - Generate Report

When the user selects the "file a report" option from the main menu, the program prompts the user to enter information about the incident. The information that the user is prompted to enter includes the required and optional information. The submitted report will then be saved in the database.

~~G.2 - Filtered Search~~

~~The filtered search feature works by accessing the data in the program's database or file system, and then applying the selected filters to the data. The filters may include options such as selecting a specific date range, location, or graduating class. Once the filters are selected, the program will display the relevant data based on the criteria.~~

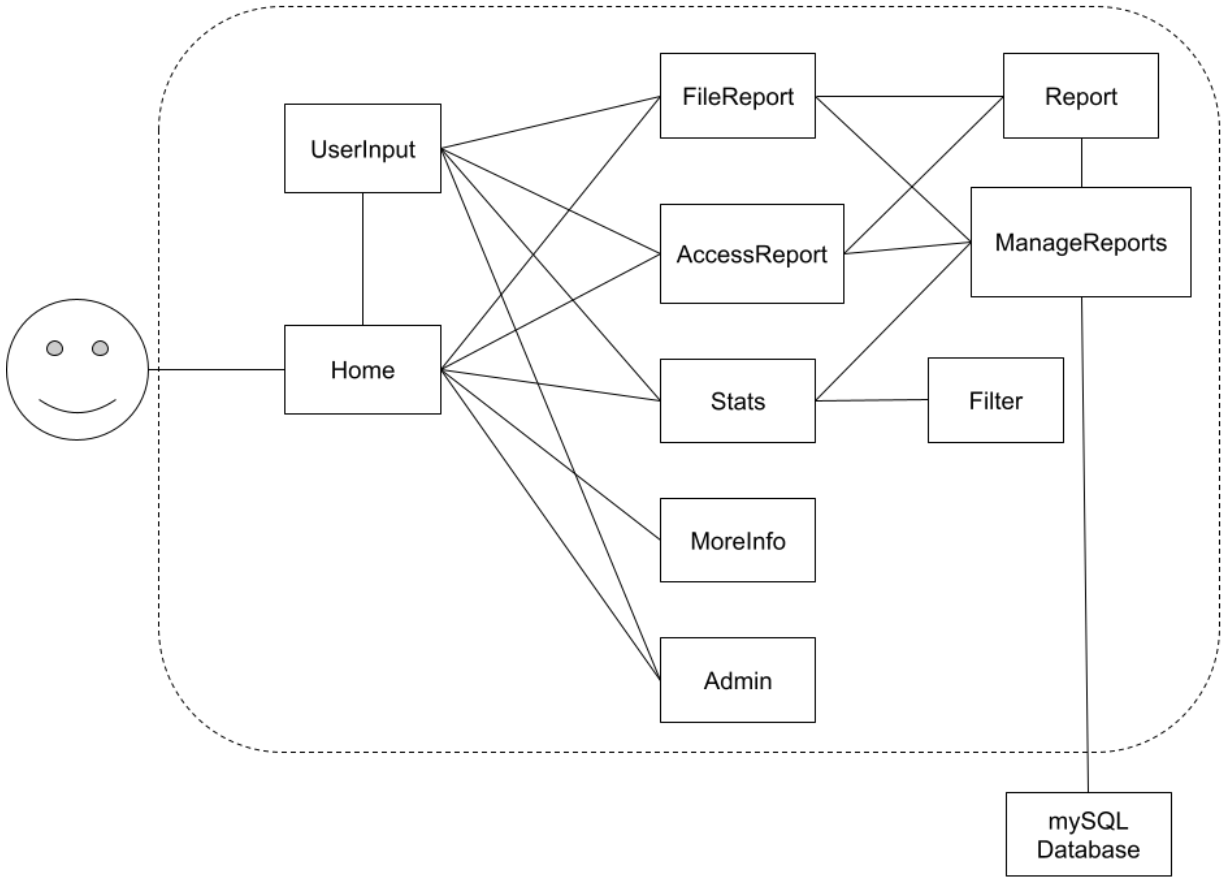
System Design

This section describes the system design in detail. An overview of the major system components and their roles will be provided, followed by more detailed discussion of components. Component discussion will reference the technical features the component helps satisfy.

Note: Design co-evolves with implementation. It is important to start with some sense of components and their interactions, but design docs should be updated during implementation to capture the actual system as-built.

Architecture Overview

High-level System Architecture



High-level Architecture Diagram

SRAS interacts with the host computer. The user and data file appear outside of the system boundary for SRAS. Inside of the SRAS system boundary are several components that interact with one another.

Major Components

Home

P.2
The Home component is essentially the first page or list of prompts that the user will see after opening the application. It is the main hub of navigation, and from it users can file a report, view database info, get more information on SRAS, or log-in as an administrator.

UserInput

P.2, L.1
The UserInput component is a Scanner-type class that detects user input. It will support different scenarios like entering numbers or text, and it will also have error handling in case of user error. There will be many methods defined within this component that can be called on for any other actions, like Re-Accessing a report or selecting a prompt from the Home page.

FileReport

P.2, L.2, M.3, G.1
The FileReport component is responsible for generating a new report, asking for report information from the user, and working with ManageReports to save a new report to the database.

This component uses the Command Pattern Design to construct the report, keep track of the information provided by the user and execute all those commands when the user is ready to submit the report. This pattern design shares the same implementation as AccessReport.

AccessReport

P.2, M.1
The AccessReport component allows the user to enter report ID and view the submitted report and edit it, in which this component works with ManageReports to update the edited information in the database.

This component uses the Command Design Pattern to keep track of the edits that the user would like to make and execute all those commands when the user is done making edits. This pattern design shares the same implementation as AccessReport.

Report

G.1

The Report component is responsible for holding information pertaining to an incident reported by the user.

ManageReports

L.2, M.1, M.2
The ManageReports component is a connecting class that helps store and retrieve information from a CSV file.

Stats

P.2, M.1
The Stats component is responsible for showing all of the current statistics and an option for the user to filter through those statistics.

~~Filter~~

~~G.2~~
~~The Filter component is responsible for filtering the statistics based off of date, location, and/or graduation class. The user specifies the criteria they want to filter by.~~

~~This component will use the Decorator Design Pattern to stack the criteria a user is filtering by.~~

MoreInfo

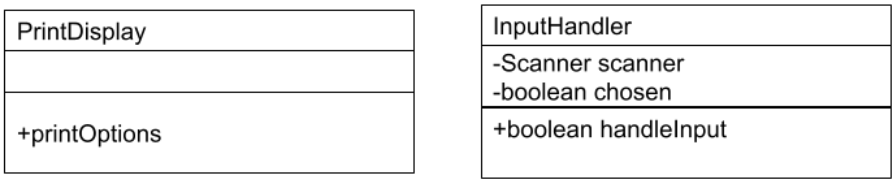
P.2
The MoreInfo component is responsible for displaying the ‘additional resources’ information available for a user (i.e. Campus Safety, the Title IV Office, and the Counseling Center) or the ‘about’ information for our program.

~~Admin~~

~~M.1~~
~~The Admin Component is responsible for viewing all of the data recorded in the system.~~

Detail Design

Home Component



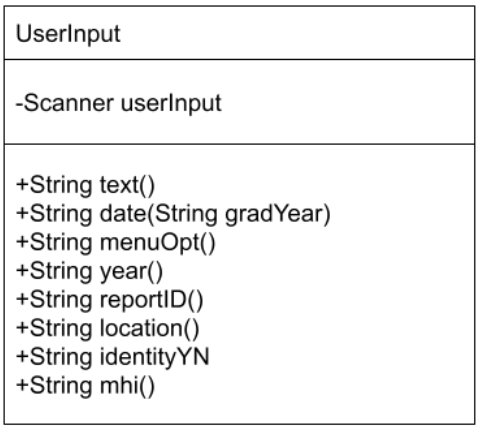
PrintDisplay

HomePage primarily contains the logic to print out the options of the home menu.

InputHandler

InputHandler encapsulates the logic to make sure the user selects only existing menu options and directs them to the correct option once chosen.

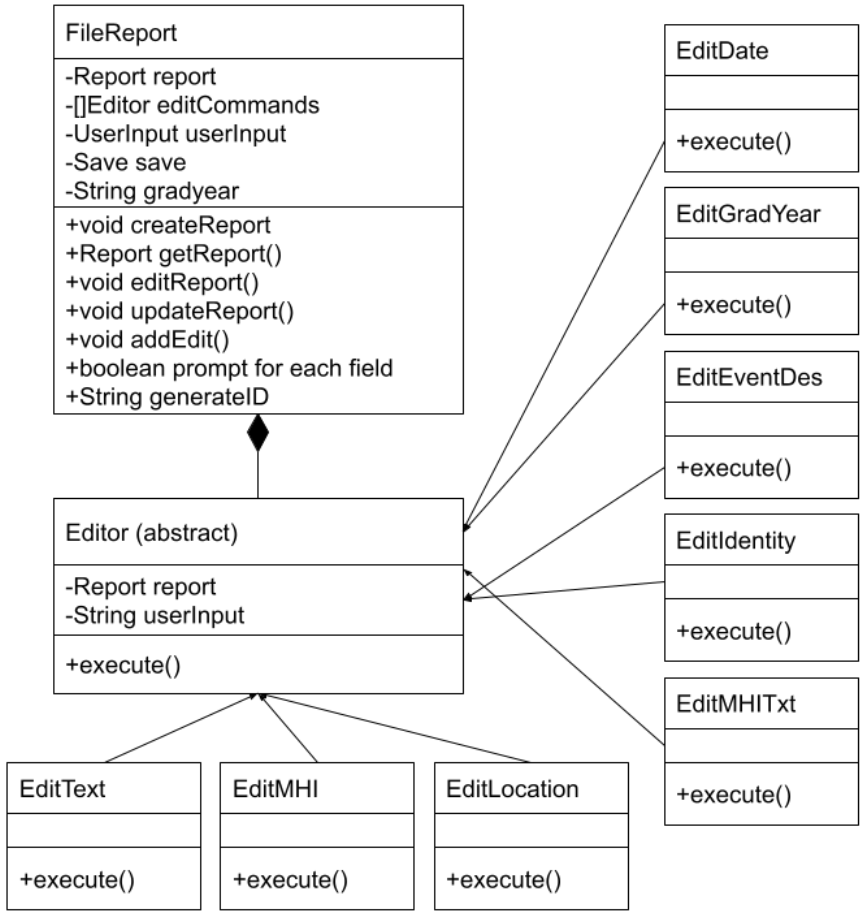
UserInput Component



UserInput

UserInput encapsulates the logic to ask for user input and make sure that the user enters prompted information in the correct format. Whenever user input is required, a method from this class is called corresponding to the prompt. If a user provides input in a format that is not expected, then an error message appears to provide the correct format and the user is put into a loop until the expected format is provided.

FileReport Component



AccessReport

AccessReport encapsulates the logic to display prompts and take user input to fill out a report, allowing the user to submit it into the database or to cancel making the report. It is part of the command pattern design as an invoker and receiver that makes the commands and executes all of them at once to fill out the report and save it to the database.

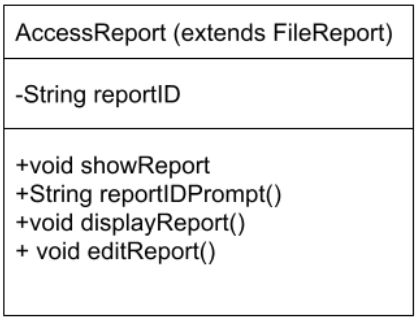
Editor

Abstract class for making edits to the report.

Edit*

Implementations of EditCommand execute method, with each edit filling in information for each attribute of the report.

AccessReport Component



AccessReport

AccessReport encapsulates the logic to display information of a report and allows users to edit the report or remove it from the database. It extends the FileReport class, so it is able to use FileReport's command pattern design to allow the user to edit reports and execute all of them at once in the edit report method.

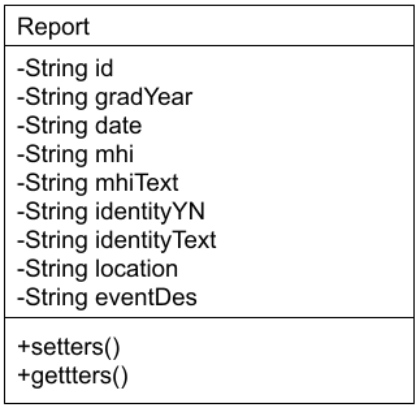
~~Editor~~

~~Abstract class for making edits to the report.~~

~~Edit*~~

~~Implementations of EditCommand execute method, with each edit make changes to an attribute of the report.~~

Report Component



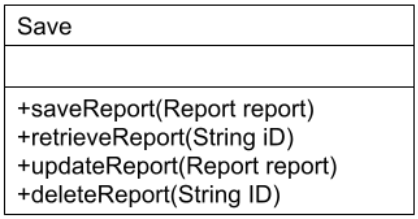
Report

A report is made up of various information, which will be all String objects, and can be set and get by other components to access and set information of the report.

~~Identity~~

~~Identity encapsulates the personal information attributes that the user may want to provide.~~

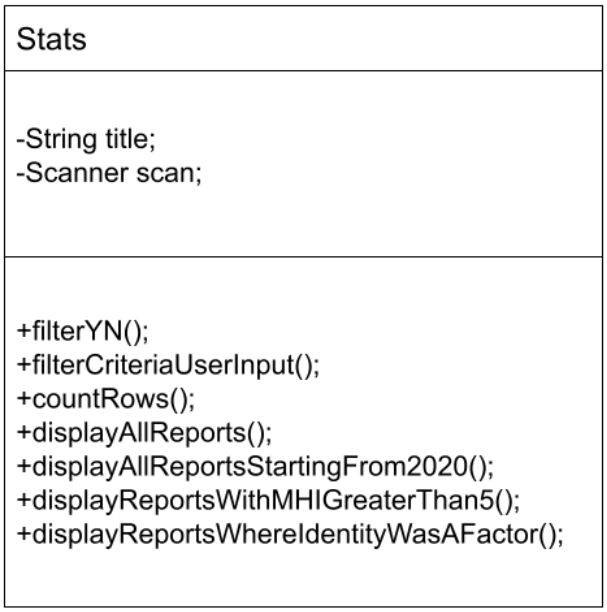
ManageReports Component



Save

This class has methods that act as data manager, interacting with the mySQL database system. It contains a method to save report to the database, retrieve report from the database using report ID as primary key, update report in database, and delete report using report ID as primary key.

Stats Component

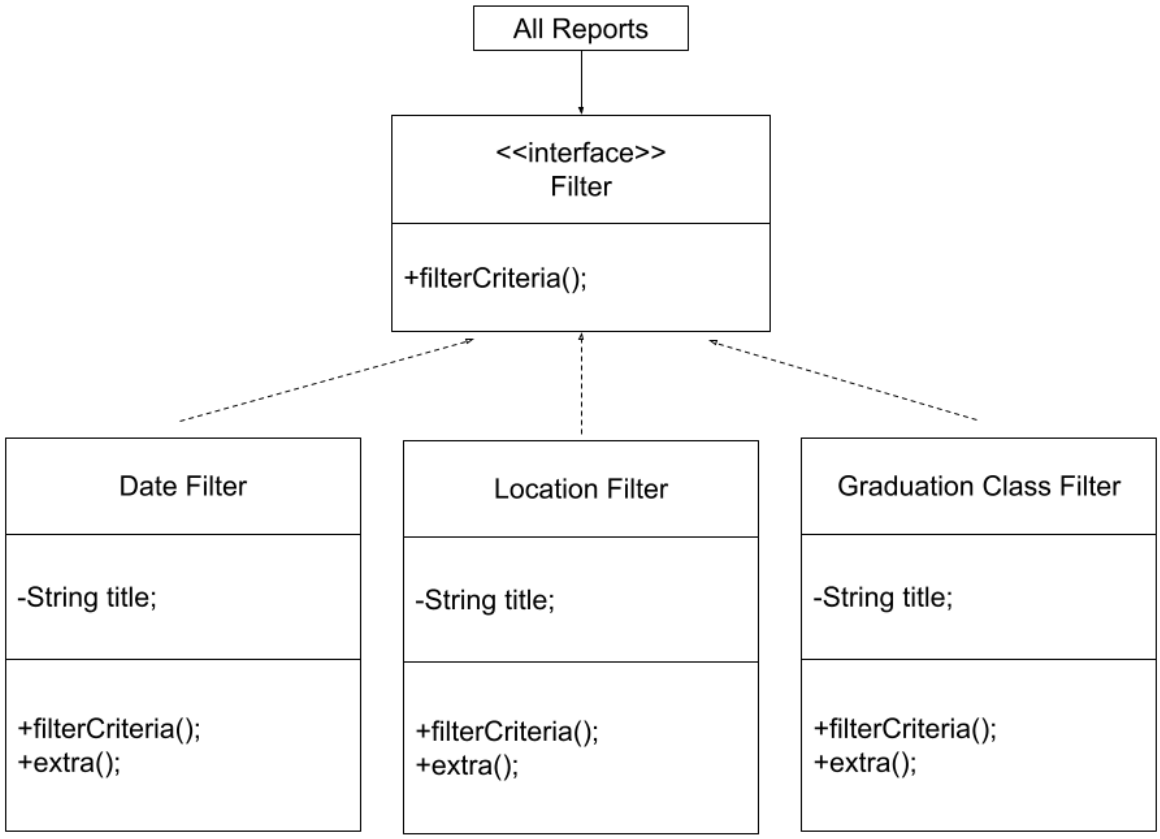


Stats

Stats is a class that will display default campus statistics and ask the user if they wish to further filter through our reports. They will then have the option to type in one of three filter options (i.e.

date, location, and graduation class). It will then get the filtered statistics from either the Date Filter class, the Location Filter class, or the Graduation Class Filter class.

~~Filter Component~~



~~All Reports~~

~~All Reports is a base class that gets all of the reports that have been properly submitted to our program. This is what the Filter classes filter through.~~

~~Filter~~

~~Filter is the common interface that will be used by the specified Filter classes.~~

~~DateFilter~~

~~DateFilter is a class that will override what was stated in the Filter Interface to specifically filter through a date user input. It will ultimately display statistics for a specific date(s). Any additional additions fall under the extra() method.~~

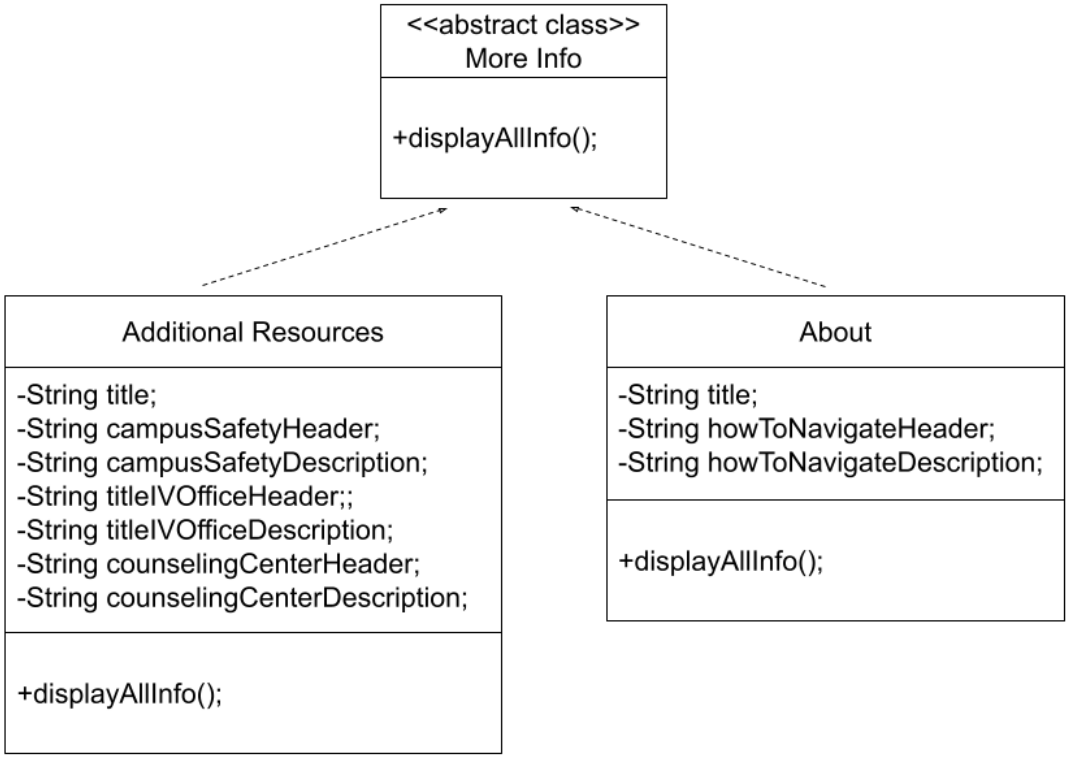
~~LocationFilter~~

~~LocationFilter is a class that will override what was stated in the Filter Interface to specifically filter through a location user input. It will ultimately display statistics for a specific location(s). Any additional additions fall under the extra() method.~~

~~GraduationClassFilter~~

~~GraduationClassFilter is a class that will override what was stated in the Filter Interface to specifically filter through a graduation class user input. It will ultimately display statistics for a specific graduation class(es). Any additional additions fall under the extra() method.~~

MoreInfo Component



MoreInfo

MoreInfo is an abstract class that holds the basic information used in our ‘additional resources’ and ‘about’ pages. String attributes were assigned to the various headers and descriptions since that is how users will be able to read through them.

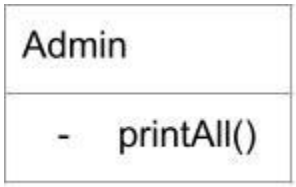
Additional Resources

Additional Resources extends MoreInfo and overwrites its original method. It should only display additional resources available for students.

About

About extends MoreInfo and overwrites its original method. It should only display general information about our program.

~~Admin Component~~



~~Admin~~

~~The Admin class is relatively simple, it contains a method to print all of the information stored in the csv document.~~