



GRADR

User and Technical Guide

Table of contents

Set up and using Gradr

Logging in	3
The welcome screen	4
The navigation bar	5
The classes page	6
The students page	7
The grades page	8

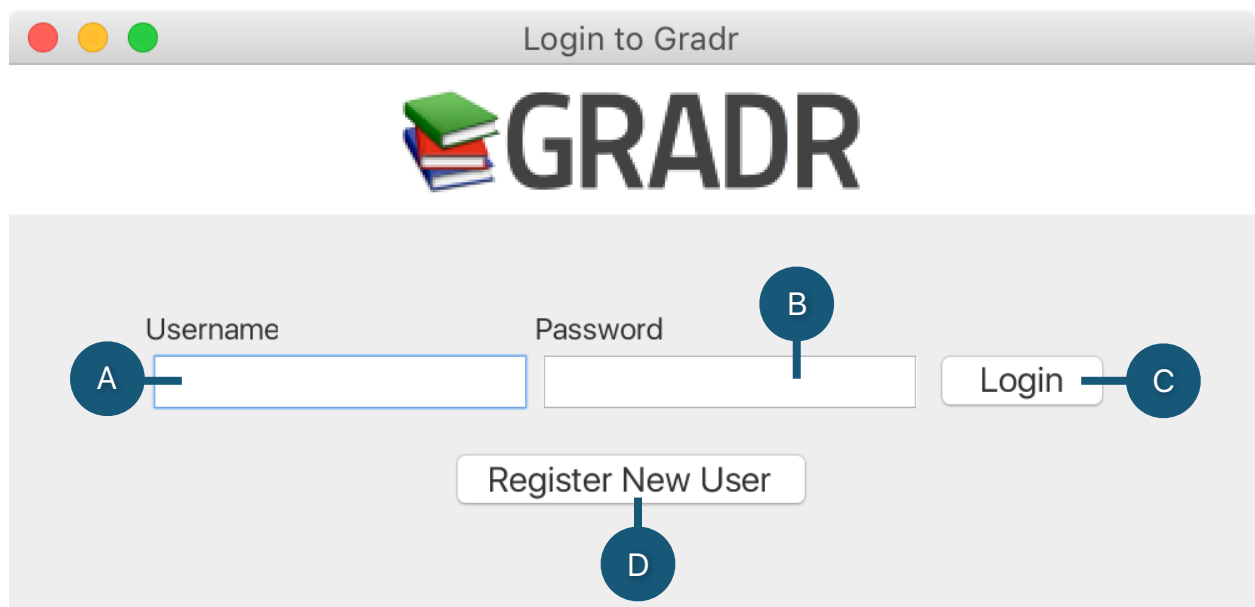
Technical details

Introduction to the Logic	9
Filesystem	10
Major classes in the Logic	11
student.java	
gradedAssignment.java	
classroom.java	
teacher.java	
fileIO.java	12
Major Methods of the File I/O	12
Introduction to the GUI	14
Major classes in the GUI	15
Login.java	
Gradr.java (Dashboard)	
AddButton.java and AddClass.java/AddStudent.java	16
Classes.java	
DataFileTable.java and DataFileTableModel.java	
Students.java	17
Grades.java	
Attendance.java	18

Set up and using Gradr

A quick overview

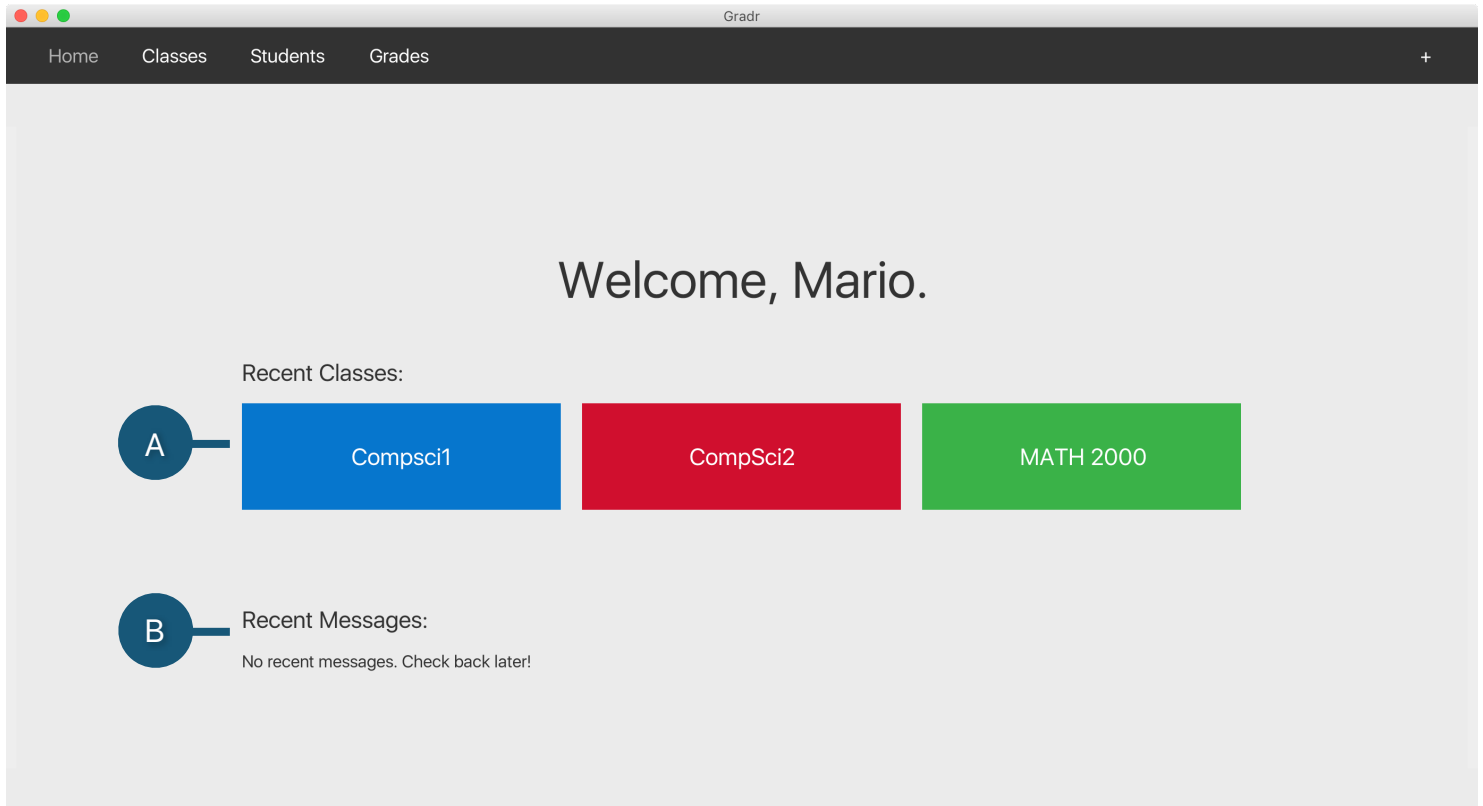
Logging in



The screenshot shows a web browser window titled "Login to Gradr". The page features the Gradr logo, which consists of a stack of three books (green, red, and blue) next to the word "GRADR" in a bold, sans-serif font. Below the logo, there is a login form with two input fields: "Username" and "Password". The "Username" field is labeled with a blue circle containing the letter "A". The "Password" field is labeled with a blue circle containing the letter "B". To the right of the "Password" field is a "Login" button, labeled with a blue circle containing the letter "C". Below the "Login" button is a "Register New User" button, labeled with a blue circle containing the letter "D".

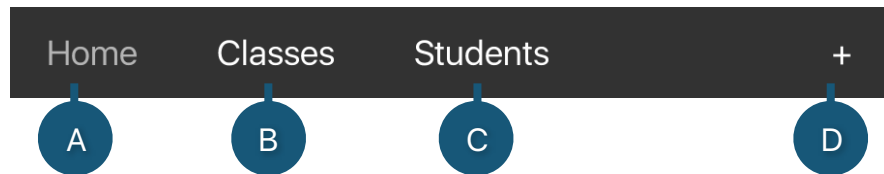
- | | |
|---|------------------------------|
| A | Username field |
| B | Password field |
| C | Login and launch application |
| D | Register a new user account |

The welcome screen



- A Quick links to most recent classes
- B Message center

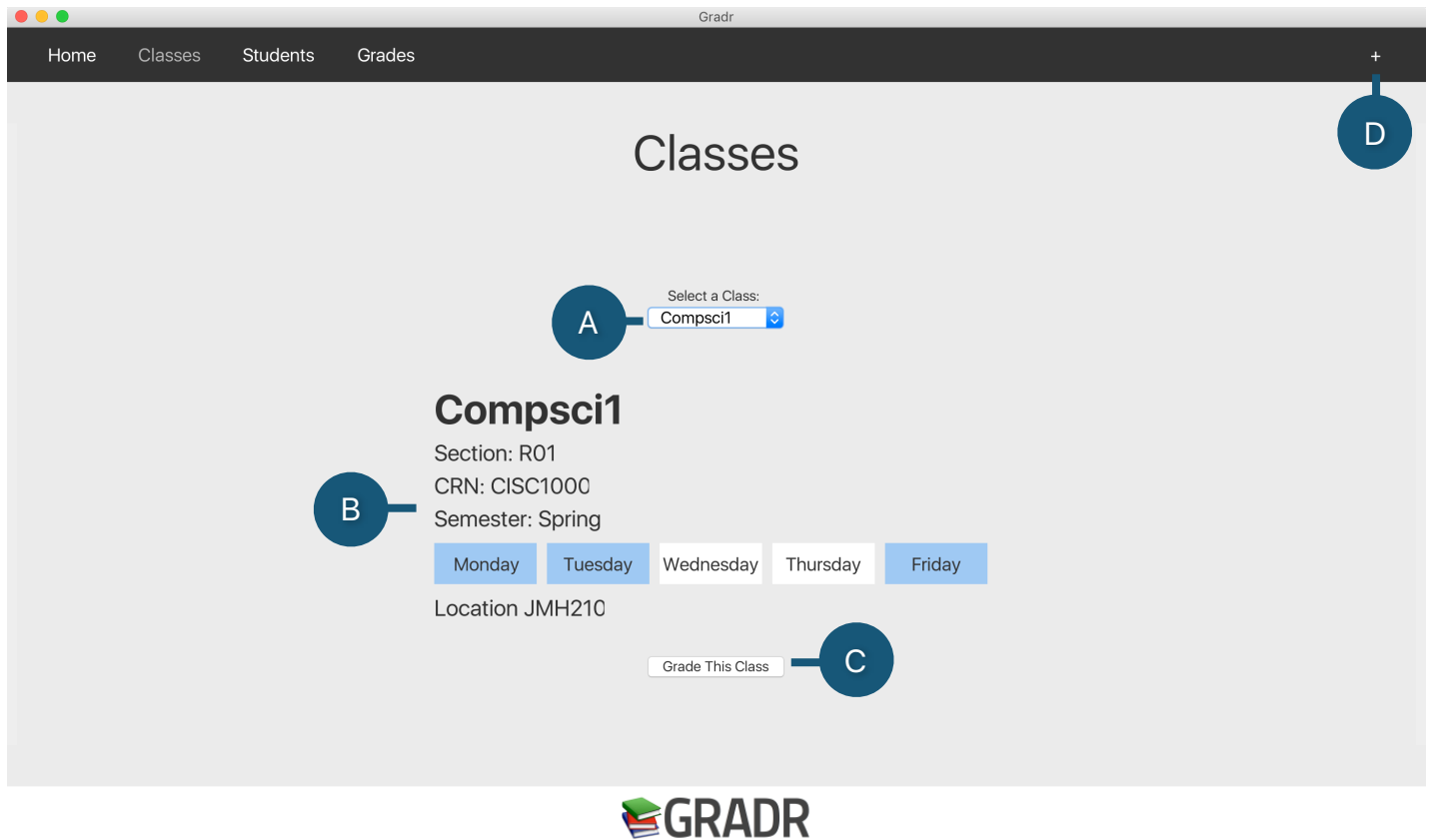
The navigation bar



Highlighted text indicates the current page being viewed.

A	Home page
B	Classes page
C	Students page
D	Add new class or new student

The classes page



A

Select which class to view

B

General information about the selected class

C

Link to the grade page for the selected class

D

Add new class

The students page

Gradr

Home Classes Students Grades

Master Students List

Master Students List

A

B

Id Number	Last Name	First Name	Age
A123	Merendino	Mario	20
A345	Cantwell	Brooke	30
A678	Merendino	Joseph	12
A010321	Subject	Test	30
A9648963	Student	Newest	532



A

Filter student list by specified class

B

Add a new student

The grades page






The screenshot shows the Gradr application interface. At the top, there's a navigation bar with 'Home', 'Classes', 'Students', and 'Grades'. The main header displays 'Compsci1' with a dropdown menu showing 'Compsci1' and a grade filter 'A'. Below this is a table of student grades:

Last Name	First Name	Age	Id Number	Test1	Test2	Test3
Merendino	Mario	20	A1242	96	90	98
Brooke	Cantwell	20	A5324	95	93	99
Carlo	Mazzoni	22	A1154	90	88	89
Joseph	Barry	21	A0977	84	82	87
Christine	Papadakis	19	A8832	79	88	90
Megan	Mattocks	21	A6411	64	55	65

At the bottom, there's a sidebar with buttons labeled A through E, each with a corresponding icon:

- A: Filter grades list by specified class
- B: Add or edit an assignment
- C: Take class attendance
- D: Display assignment and class average statistics
- E: Save all changes made to grades



-  Filter grades list by specified class
-  Add or edit an assignment
-  Take class attendance
-  Display assignment and class average statistics
-  Save all changes made to grades

Technical details

Introduction to the Logic

The logic behind our program was based around several classes, each representing different objects required for the gradebook: classes (or classrooms), assignments, students, and teachers. Additionally, we created a file input/output class so all of the methods for reading and writing to our file system were contained in one class. Instances of this file I/O class could be created in any other class, and these input and output methods could be called on the file I/O objects, passing any other objects as arguments to constructors or methods.

Before creating any GUI elements, we first developed a functional terminal application to test the back-end and logic. Creating a terminal application before adding GUI allowed us to link the front-end and back-end of our project more easily. Also, fixing problems in the logic of the program made it easier to isolate any issues in the user interface, so debugging mainly took place in two different stages.

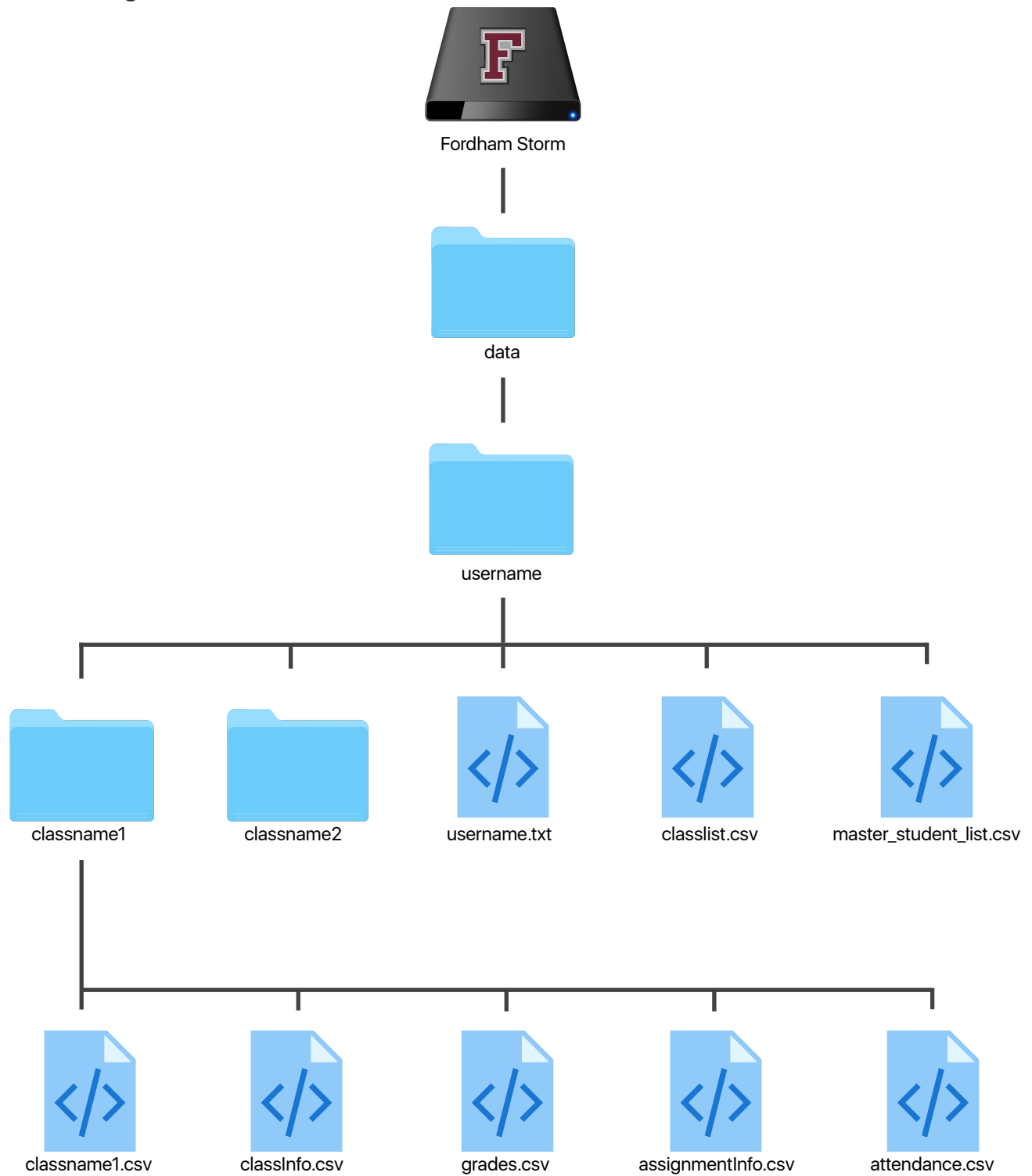
The file input and output class is one of the more important aspects in the logic or back-end functionality of our program. We organized our file system to create one main directory for any data generated by file output methods. There is a directory for each class created. Each class directory contained five different CSV files: one for attendance, another for class information (dates, meeting times, location, etc.), assignment information, a file with the name of the class containing a list of student information, and a file containing grades.

```
[brookecantwell@storm gradebook]$ ls
CISC1000      classroom.java  gradebook.java  student.java
CISC2000      directory_file gradedAssignment.class test.txt
CISC3000      FileIO.class   gradedAssignment.java
class_list.csv FileIO.java     gui
classroom.class gradebook.class student.class
[brookecantwell@storm gradebook]$ java gradebook
Welcome to the gradebook
What would you like to do?
    1)Add Class
    2)Display List of Classes
    3)Edit Class
    4>Delete Class
    0)Exit
1
Enter class name: CISC3598
```

The terminal application before adding a graphical user interface.

Filesystem

A visual guide



Major classes of the Logic

student.java

Data Members:

- firstName
- lastName
- average
- age
- idNumber

gradedAssignment.java

Data Members:

- score
- weight
- name
- weightedScore

classroom.java

Data Members

- currentTeacher
- className
- crn
- section
- semester
- meetingTime
- daysOfTheWeek
- location

teacher.java

Data Members

- password
- username
- firstName
- lastName

fileIO.java

As described above methods for reading in from files and writing to files are not contained in the classes they operate on but are instead aggregated into the file I/O class and called on a fileIO object passing the objects they operate on as arguments. The file I/O class is composed of methods that accept objects and strings as arguments to perform input and output to the different files in the file system. Most of the methods are either void (for writing) or return lists of objects of the other classes. There are some more general methods that, for example, create directories and files, and more specific methods used to write to specific files using objects of other classes, or read in CSV files, splitting the lines and assigning the values to data members of specific classes, and returning objects of the class or lists of objects.

Major methods of the File I/O

createDirectory(String path)

Uses the path argument to create directories and files in the file system.

teacherRegistration()

Creates a directory for a new teacher upon registration and saves the username, password, first name, and last name to a file in the directory.

addToClassList(String username, String className)

Writes the names of classes to a file in the teacher directory that is used to read in class names to create and edit classes, create drop-downs, and specify files to read in.

readClassList()

Reads in class names written to the class list.

appendStudent(student s, String className)

Appends information from a student object to the file belonging to whichever class the student is being added to.

addToMasterStudentList(student s)

Adds information from a student object to a master student list and checks for duplicate student IDs.

readMasterStudentList()

Reads in the master student list.

readStudent(String className)

Reads students from the file corresponding to the class name passed as the argument.

login(String username, String password)

Reads information from the teacher's CSV file created when registering to check passwords against usernames.

readTeacher()

Creates a teacher object using information from the login method to be used in all other classes to display information and specify directory paths.

createClassInfoFile(classroom c)

Uses information from a classroom object to create a file containing class information such as meeting days, time, and location.

getClassInfo(String className)

Reads in information from the file created by the createClassInfoFile method.

saveAssignments(Vector obj, String className)

Reads in assignments as a vector to be used for storing information from tables.

assignmentsNames(String className)

Stores information about assignments (name, number, weight) to be used for displaying grades and calculating statistics.

getAssignmentAverage(String className, String assignmentName, String type)**getStudentAverage(String className, String studentNameCombo)****getTotalPoints(String className, String assignmentName)**

Used for calculating and storing grade statistics.

printAttendance(String className, String amountAbs, String studentName)

Stores information about attendance and updates the file if the amount of absences is incremented or decremented.

getAttendance(String className, String studentName)

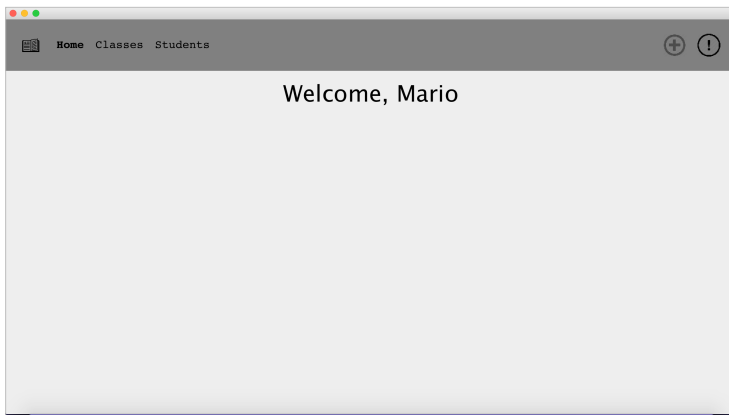
Reads in the attendance file.

Introduction to the GUI

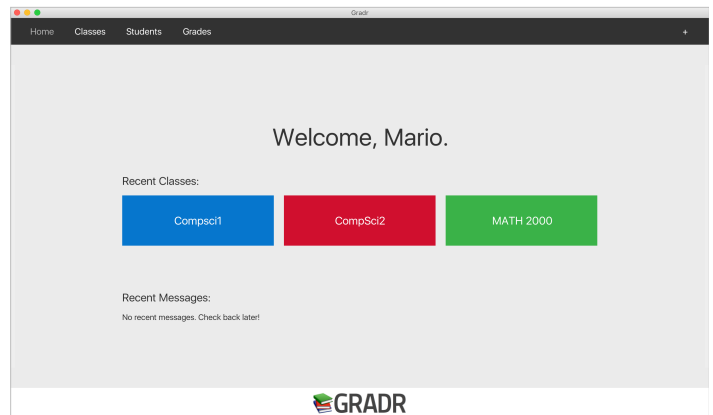
The GUI was designed with the user in mind. It's easy to follow with clear fonts, exciting colors, and large buttons and labels. We started with a very minimalistic design at first, only implementing the core features, then slowly cleaned up the UI to bring it to its current state.

The UI started out as only a terminal application, with the barebones. It only allowed the user to add classes, add students, and edit some information. We slowly implemented a GUI around the terminal application, until it was redundant to work on the inline application. The main class of the GUI is the Dashboard(Gradr.java). The UI started out looking very ugly, it was just enough to start working on applets, and frames to work on the UI. Then overtime we made the UI look nicer and nicer until we almost matched what we said we were going to do in the proposal.

The beauty of our GUI is that we programmed it with modularity in mind. Anytime we needed information from the CSV files or other classes, FileIO class is called to retrieve the data. This enables us to get information very easily through out the entire UI. Additionally, a function was leveraged to swap out the center panel of the main Dashboard to update the screen with new information.



October 29, 2016



Today

Major classes in the GUI

Login.java

The login screen is a very basic JFrame with a border layout. The north panel just displays our logo, and the center panel has a JTextField for the username input, a JPasswordField for password input, and two JButtons, one for logging in, and one for registering. The two important functions in the Login class are registerButtonActionPerformed and loginButtonActionPerformed. They both create objects of FileIO, and in the register button function all it does is call the FileIO function teacherRegistration(); and in the login button function all it does is calls the FileIO function login(username, pass), and if it returns true it opens the dashboard, and it closes the login screen.

Gradr.java (Dashboard)

The dashboard is where the GUI lives. All of the functionality of the program lies within the dashboard. Its not all too complex either, the whole thing is a border layout, with the North portion dedicated to the navigation bar which is its own JPanel, and the South portion containing our logo. The navigation bar is really nothing too special, it just has a few labels with listeners on each one, upon click it makes that one bold, and makes the rest have plain text. Also it calls a function to change the panel in the center to open the corresponding page. The way this is created is in the function createNavigationBar() that returns a JApplet. The center portion is the most interesting however. At default, it loads in the homepage, which is its own JPanel. That function is called showHomePage() that creates the panel with a label that welcomes the user and displays the 3 most recent classes. The best part about the dashboard is the swapCenter(JPanel) function, that takes in a JPanel and removes the current one, and replaces it with the one that was passed in, making it effortless to update the center screen whenever needed.

AddButton.java and AddClass.java/AddStudent.java

After clicking the plus button in the navigation in the the swapCenter function is called being passed the AddButton constructor, because the class extends JPanel it has all the properties of a JPanel. The user is now prompted to add a class or student. That page just has a JLabel saying "CreateNew..." with two JButtons underneath saying student or class. Clicking the student button or the class button will launch the AddClass or AddStudent classes and because both extend the JFrame class it will open a popup prompting the user to input class, or student information. In both, there are mostly JTextFields to input class or student information. In both classes the most important function the the submit button. In each class they have a submit button, that takes the information from the textfields, and creates either a classroom object or a student object. That object is passed to the file IO functions to save the student and class information to the appropriate files. The student object gets saved to three files, the master student list, the appropriate class list, and the appropriate class grades file.

Classes.java

The classes page, is opened when the user clicks the Classes button on the navigation bar. The classes page has a constructor that takes in a classname, which allows for the recent class buttons on the homepage to be used to pass in the correct information. The classes class extends the JPanel class, so the swap center function is used from the main dashboard. The page loads information from the class information file using the file IO function getClassInfo(classname). All the class information is displayed using JLabels. There is also a JComboBox (dropdown) that has a list of all the classes that allows the user to change the class being viewed. The JComboBox gets the information from the FileIO function readClassList(). The update(classname) function updates all the JLabels to to have the corresponding class information.

DataFileTable.java and DataFileTableModel.java

These classes work together and are primarily used to create JTables from CSV files. The best part about this class is it is used for converting any CSV file into a JTable, making it useful for the Students page, and the Grades Page. The DataFileTable class is used for determining which CSV file to load. The DataFileTableModel class is where the actual code is, and the important function in this class is initVectors() which loads the appropriate CSV, and turns it into a vector called data. Once the data is stored into a vector it is easily converted into a JTable.

Students.java

The students page is opened from the navigation bar, and the swap center function swaps the current center page with the students page, this is possible because the class extends the JPanel class. The students page is super simple, as it only contains three things, a JLabel, a dropdown containing the class list, and a JTable. The whole panel has a border layout and at the north position it contains the JLabel and dropdown, and the center contains the JTable. At first it just uses the DataFileTable to create a JTable that by default loads the Master Student list file. The master student list contains every student the teacher has. There is a listener on the drop down, and if the user changes the drop down to a different class, the JTable displays the appropriate class list. The listener calls the update(classname) which calls a swap center function for the center panel of the Students JPanel. The center panel was just the JTable, so we swapped it with the appropriate JTable for the correct class.

Grades.java

The grades page is opened from the navigation bar, and the swap center function swaps the current center page with the grades page, this is possible because the class extends the JPanel class. The grades page is very similar to the students page. Just like the students page, the JPanel has a border layout and at the north position it contains a JLabel and dropdown, the center contains a JTable, and the south position has a row of buttons of JButtons, and each one has a different function. The center panel creates a DataFileTable object to create a JTable that by default loads the first class in the class list file. Just like the students page there is a listener on the drop down, and if the user changes the drop down to a different class, the JTable displays the appropriate class list. The listener calls the update(classname) which calls a swap center function for the center panel of the Grades JPanel. The center panel was just the JTable, so we swapped it with the appropriate JTable for the correct class. The table allows for edits for the grades, and you can save the edits by clicking the save button. That listener function takes the JTable, and converts it to a vector. That vector is then passed through the File IO function saveAssignments(vector, classname). That saves all the data. All the other buttons on the bottom bar all have listeners that all call a JFrame popup that allow the user to add assignments, check attendance, check class statistics, and edit assignment information.

Attendance.java

The attendance page is a JFrame, so it acts like a mini pop-up in the application. The page allows you to add or remove days that a student has missed. The page has a drop down that has a list of the students, then a JLabel that acts like a counter, and two buttons that allow the user to increment or decrement the students attendance. There is also a save button that saves the attendance information. That save button calls the FileIO function `io.printAttendance(className, number of Absences, students Name)`. The page is constantly converting integers to strings, and vice versa because JLabels only have support for strings.



Created and edited by Mario Merendino, Brooke Cantwell, and Carlo Mazzoni.

© 2016 Gradr Inc. All rights reserved.

Gradr and the Gradr logo are trademarks of Gradr Inc, registered in the U.S. and other countries.