Christina Buffo and Midori Yang

Technical Report

Abstract Data Types

-Linked Lists:

We plan to use linked lists for several things: storing the meeting objects in a course object, and storing the collection of days and their information in the schedule.

We intend to use linked lists to store the meeting objects within the course object because the meeting objects need to be in a specific order (from meetings earlier in the week to later in the week) and we need to be able to potentially access any of the meeting times, rather than just the most recently entered or the first entered. Also, the number of meetings per course can be of variable length, so a data structure such as linked lists that has a mutable length would be ideal.

For similar reasons, linked lists would be most suitable for storing the days of the week in the overall schedule, because the days need to be accessed separately from each other and must be accessed at any point in time without worrying about enqueuing or popping from a stack.

-Trees:

Trees are the most fitting data structure to store the individual meeting times in each day, because there are default class times (e.g. 8:30-9:40) that a meeting instance could easily be added to. In addition, the absence of a meeting instance from a default class time could still be handled by the program without causing issues, and a meeting instance outside the default class times could be added as another leaf to the tree. Also, adding multiple classes at a time (as in, multiple children to the time parent) would be possible, so all possible class schedules could be stored in the same tree, with only one visible.

Classes:

-Meeting object:

The meeting object is a subclass of the course that stores information about the course name, meeting time, meeting day, etc., along with the specific day the meeting occurs. The day that the meeting occurs must be specified in the constructor, and cannot be changed.

-Main actions:

-getMeetingTime: Returns the meeting time of the class.
(There will be other getter methods for general information).
-setMeetingTime: Sets the meeting time of the class to the specified value.
(There will be other setter methods for general information).

-Course object:

The course object stores the meetings for each class (i.e. a class that meets once a week would have one meeting object, a class that meets twice would have two, etc.) in a linked list. It also contains information about the course in general (e.g. department and course number, meeting time), which allows general information of the course to be easily updated for all meeting times. Courses objects also contain a boolean describing whether the class is visible on the schedule.

-Main actions:

-addMeeting: Adds a meeting time (meeting of the course) to the class.
-removeMeeting: Removes a meeting time from the class, if it exists.

-getDepartment: Returns the department that contains the course, if it exists.
(There will be other getter methods for all other general class information.)
-setDepartment: Sets the department of the class for every meeting object.
(There will be other setter methods for all other general class information.)
-number: Returns the number of meeting times associated with the course.
-isVisible: Returns whether the course is displayed.
-show: Makes the course visible if previously hidden
-hide: Makes the course hidden if previously visible

-Day object:

The day object consists of an n-ary tree, with the root being a string of generic times as separate children of the day (the root), which would make adding and removing certain class times easier (for each day the class meets, go through that day's tree, find the time section, and remove the pointer from that time section to the meeting instance). It contains all possible course combinations, even multiple at the same time.

-Main actions:

-addTime: Adds a time parent to which course children can be added.
-removeTime: Removes only custom times.
-addEightThirty: Adds a class to an 8:30 time section
(Similar methods for all default times)
-removeEightThirty: Removes a class from an 8:30 time section
(Similar methods for all default time)
-getNumber: Returns the number of classes stored in the same time block

-Schedule object:

The schedule object stores three linked lists: one containing the Day trees with all possible course combinations, one storing the list of all courses, and one storing the list of all currently visible courses. The GUI will interact with each component for better display (see User's Manual and GUI description).

-Main actions:

-addCourse: Adds a course to the appropriate day and the list of all courses. If it conflicts with a previously added class, alert the user and automatically make the newly added course invisible.
-removeCourse: Removes the specified course.
-displayCourse: Displays the specified course.
-hideCourse: Hides the specified course.

-LinkedList:

Linkedlists will be used to store meetings within a course, and courses within the comprehensive list and visible list of courses within the schedule.

-LinkedNode:

LinkedNodes will be used to implement linkedlists.

-N-ary tree:

N-ary trees will be used to hold all possible courses within a Day object (see Day object).

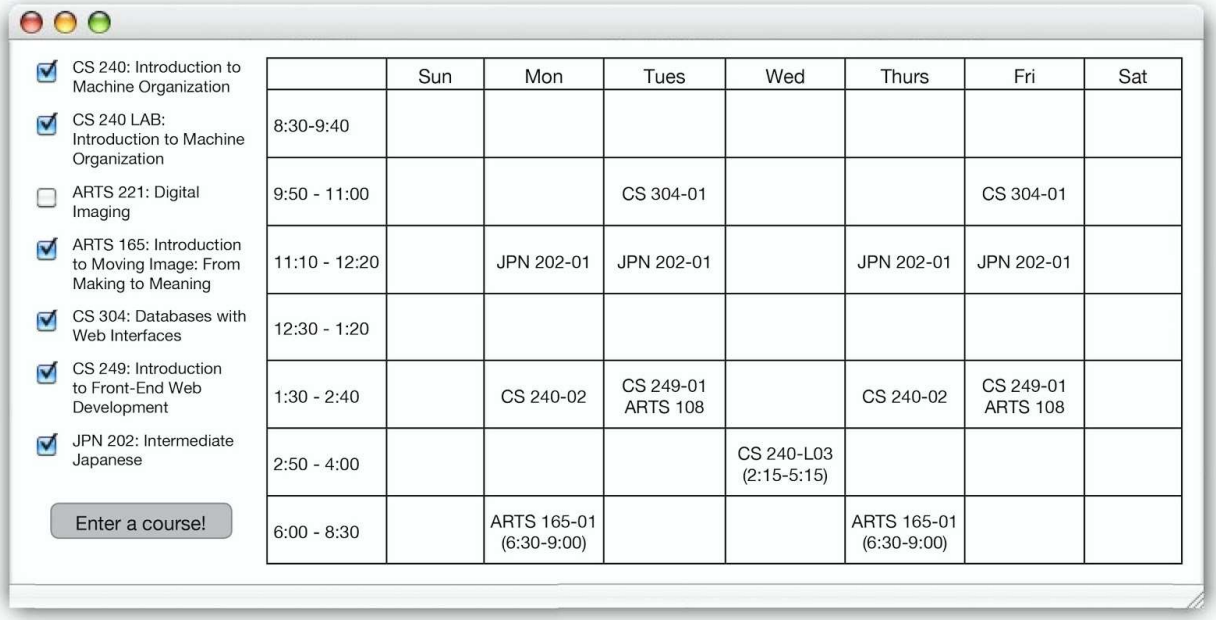
-JFrame:

JFrames will be used to implement the GUI, along with JButton, ButtonListener, JPanels, and other components from java.swing and.

-GUI:

-Main schedule panel:

The main scheduling panel will include two main components, the schedule and the course list: information for both will be obtained from the lists in the Schedule object. The schedule component will access the courses stored in the tree and, based off of which courses are visible, will display courses. The course list component will be from the list of all courses, each with a check box next to it that allows the user to change the visibility settings of the course. An example of the main schedule panel is shown below:



	Sun	Mon	Tues	Wed	Thurs	Fri	Sat
8:30-9:40							
9:50 - 11:00			CS 304-01			CS 304-01	
11:10 - 12:20		JPN 202-01	JPN 202-01		JPN 202-01	JPN 202-01	
12:30 - 1:20							
1:30 - 2:40		CS 240-02	CS 249-01 ARTS 108		CS 240-02	CS 249-01 ARTS 108	
2:50 - 4:00				CS 240-L03 (2:15-5:15)			
6:00 - 8:30		ARTS 165-01 (6:30-9:00)			ARTS 165-01 (6:30-9:00)		

-Adding class panel:

The adding class panel will consist of various user input options for the course department and number, the course name, prespecified time or custom time, and notes. An example of the adding class panel is shown below: