Cue Sheet App 2.0

CSC 260 - Large-Scale Software Development

Team Members: Patrick Aung, Kevin Ng, Woody Wu

March 14th, 2019

I. Project Description

The goal of Version 2 is to upgrade Version 1 cuesheet application by adding more features to the graphical user interface and improving design choices.

II. New Application Features and Design Patterns

Following are 11 additional application features and design patterns implemented for Version 2.0:

1. **Remove Segment**: Removes selected segments from the overall route if segment has been added to the route.
2. **Remove Multiple Segments:** On top to being able to remove segments, Version 2.0 also allows users to remove multiple segments at a time from the canvas display.
3. **Clear Route**: The user can now clear their canvas. Clearing the canvas clears the overall route display as well as the overall route itself. However, this state is saved in RouteState object for undo feature to work.
4. **Fancy Drag**: Once segments are added to the canvas display, individual segment vertices can be dragged, and the motion of the dragged segments (and accurately updated edges) are visible. Who does not want a fancy drag?
5. **Export in 2 file formats**: Users can now choose to export the final route in either .csv or .txt file formats from the canvas display.
6. **Export in Abbreviated Format:** The final route can now be exported in an abbreviated form (descriptive directions are abbreviated with a simple L for turn left and R for turn right).
7. **Export After Load:** In the initial de
8. **Import in 2 file formats:** Version 2 allows the users to import .csv and .txt segment files into the segment bank.
9. **Multiple Segment Imports:** In Version 1, users have to import one .csv file at a time into our segment bank. In Version 2, users now have the ability to select multiple .csv or .txt (new feature) files to import into the segment bank.
10. **Undo**: Allows users to undo the states of the overall route on the canvas display.

11. **Chain of Responsibility:** Instead of using hard-coded file handlers, we can now process files using a chain of file handlers to find the correct file handler.

The specific implementations of the additional features will be discussed in the model and GUI sections.

III. Model Classes and Interfaces

The model added a new RouteState class and revised Route class, FileHandler interface, CSVHandler class, and TXTHandler class.

RouteState: RouteState class uses an array of route objects to keep current of current and previous states of the canvas/overall route. This class also has an integer "pointer" which points at the current index of the routestate array. The route displayed on canvas is the route the pointer points at from the routestate array. By choice of our team, we set a limit to the number of times the user can undo moves.
Say, the limit of undo moves is set to 5. If the user has already made five consecutive moves and makes the sixth move, we remove the first element from the routestate array and add the sixth move/state of present route to the end our arraylist. The pointer is updated to the last index of the array list.
RouteState is updated if users change any properties of the overall route (i.e. add segments, remove segments, label segments and clear route). The RouteState object stores all the information and contents of the overall route on the canvas display when a route is added to its routeState arraylist.

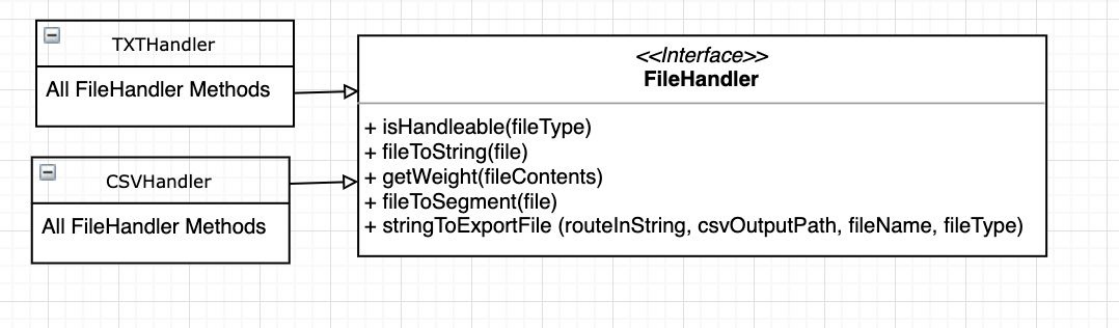
Route (revised): The fundamental design of the Route object has not changed. However, several new helper methods have been added in order to enable the new application features.

buildAbbreviationRouteString(): This function is designed for one of our new functionalities, to export in abbreviated format. In order to export a txt or csv file that represent the Route object in an abbreviated format, this method is created for converting the route object to an abbreviated string.

updateRoute(ArrayList bankContent, ArrayList routeContent, SegmentBank theBank): As the undo functionality returns the information about previous route, a helper method is needed to update the overall route content and the segment bank content within the Route object.

updateSegment(SegmentVertex vertex, int newX, int newY, Color color): This helper method is required for establishing the data communication between the view and model because of fancy drag.

segmentToMove(int x, int y): The feature of fancy drag requires the view to pass the necessary information to the model for determining the selected segment vertex.



FileHandler: FileHandler is an interface that defines what methods specific file handler classes have to implement. Every file handler class has to have an isHandleable method in order to be able to estabilish a chain of responsibility. Every file handler class will also have to implement following methods necessary for the cuesheet app:

String fileToString(File file): take a segment file and turns into a string representation

getWeight(String fileContents): takes the string representation of a segment and finds its final distance.

fileToSegment (File file): takes a segment file and returns the file's respective segment object.

stringToExportFile (String routeInString, String fileOutputPath, String fileName, String fileType): this method takes the string representation of the final route that users have built on the canvas, the directory path where the user wishes to export the final route, the name of the export file, and the fileType (either .csv or .txt) in string, then exports the final route in user selected file format.

CSVHandler: CSVHandler implements the FileHandler interface. This handler handles all .csv files using a chain of responsibility. The CSVHandler's successor is set to TXTHandler so that if the input file is a .txt file, the CSVHandler passes the file to the TXTHandler to process it. CSCHandler also implements all the methods defined in FileHandler interface.

TXTHandler: TXTHandler implements the FileHandler interface. This handler handles all .txt files using a chain of responsibility. Any file that the CSVHandler cannot process, the TXTHandler handler it. The TXTHandler's successor is set to null, since we only have 2 file handlers. If there were another file type handler, the TXTHandler's successor would be that new file type handler. If neither CSVHandler nor TXTHandler can handle a file, the file will simply not get processed, and in the terminal, we will received a message that the file cannot be handled.

IV. Graphical User Interface Classes and Interfaces

For the graphical user interfaces for the cuesheet application 2.0, the visual of the both canvas display and the segment bank display remains the same, but a few modifications were made to enhance the functionalities of the displays. The following includes the descriptions of the new available buttons:

SegmentDisplay: With the 3 standard buttons from version 1.0, there is a new "remove segment" button, which allows user to remove the selected segment from the overall route. The visual of the button is done in the SegmentDisplay class, and the functionality is implemented within the SegDisButtonListener class.

CanvasDisplay: The basic visual of the canvas display remains very similar, except with the additional buttons for several new functionalities related to the control of the overall route. The new buttons include:

1. Clear Route: Clearing the overall route in the canvas display
2. Undo: Returns and shows the previous step made by the users.
3. Redo: Returns and shows the newer step made by the users.
4. Export Route: Allows to choose export file formats, between .csv and .txt

RouteControl: Although the RouteControl object does not affect the overall visual of the application, there are a few adjustments on the implementations in order to accommodate the fancy dragging motions. While mouseListener is used in version 1.0, mouseMotionListener is also added for keeping track of the dragging motion.

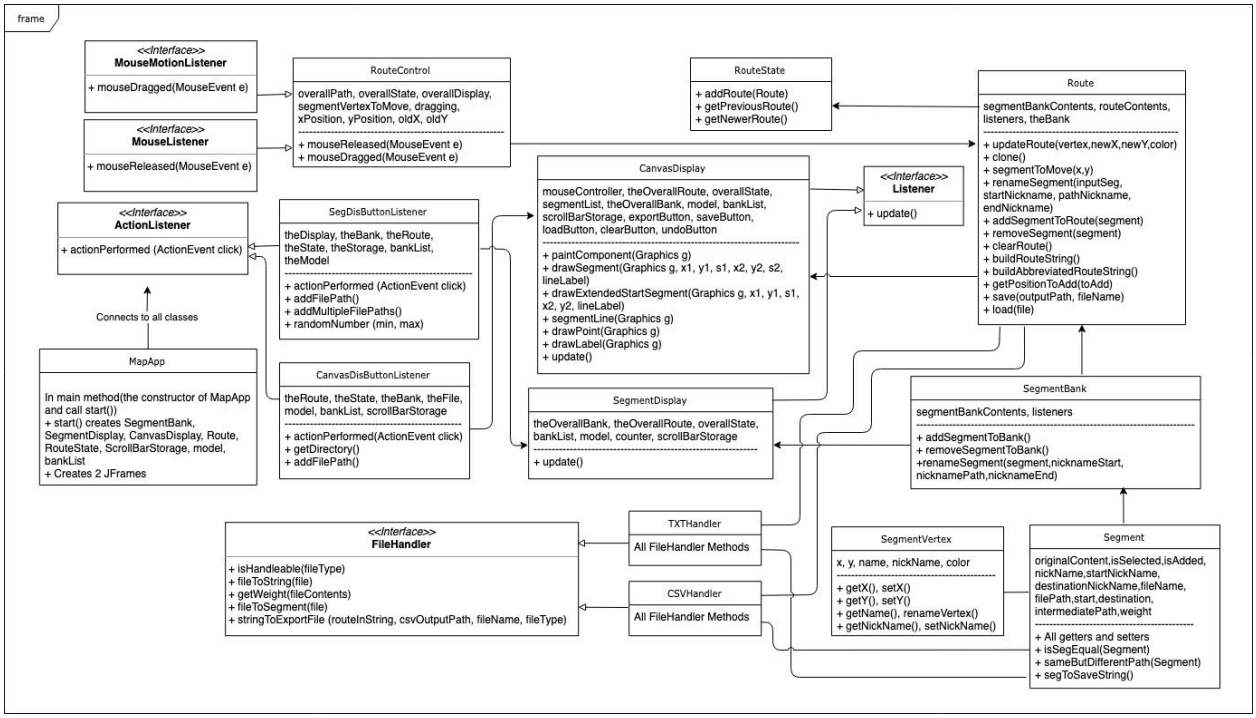
V. Work Distribution

The work distribution of this project is as follows:

Patrick and Woody were responsible for the FileHandler, CSVHandler, TXTHandler, RouteState and the modifications of Route.

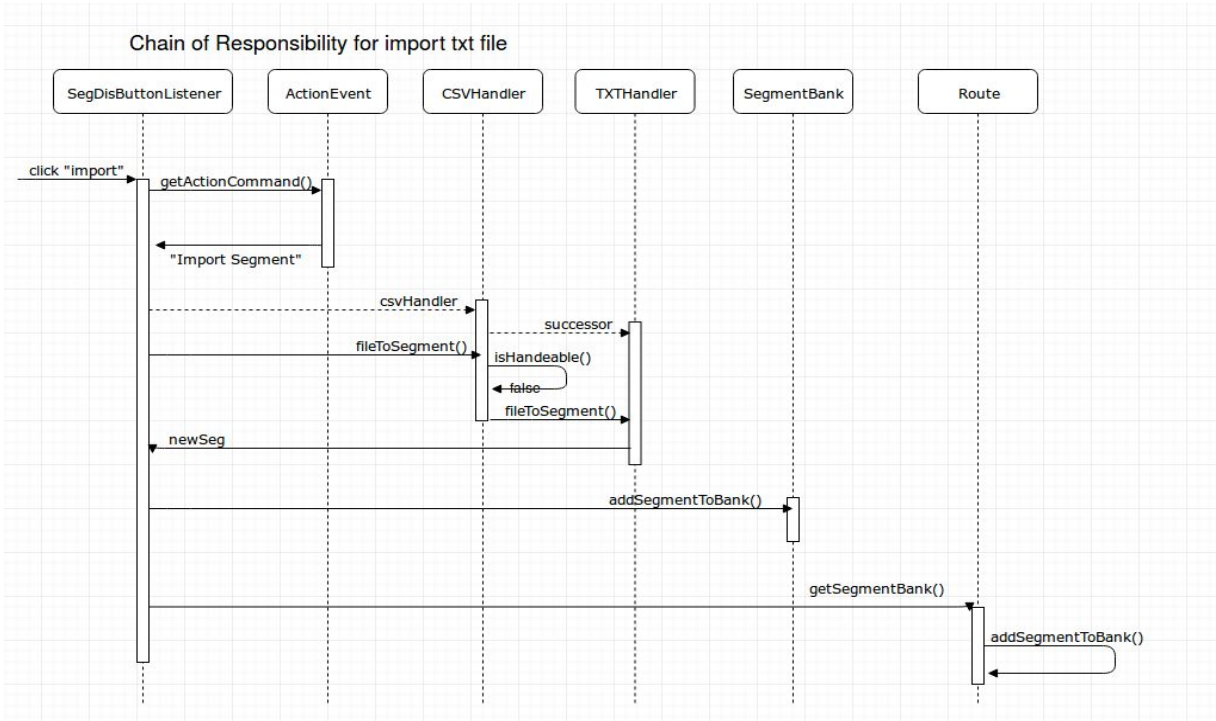
Kevin was responsible for most of the GUI-related classes, with the help from Woody for lighting up the dragging segment vertex.

VI. Class Diagram

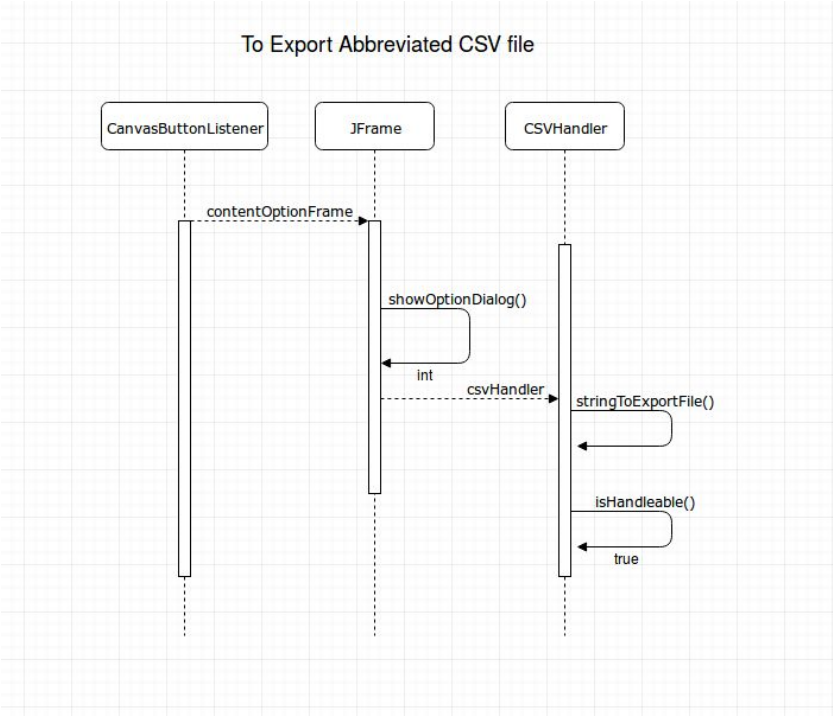


VII. Interaction Diagram

Interaction Diagram 1:



Interaction Diagram 2:



VIII. Testing

In order to ensure the functionality of classes in the model before merging them with other classes, we wrote RouteTest and SegmentTest to test if our model behaved as expected. In this version, we also wrote tests for new functions in Route class.

RouteTest:

- testAddSegmentToRoute
    - Tests for testing addSegmentToRoute method in Route class. It tests if the method add the segment in the correct order.
- testRemoveSegment
    - Tests for testing removeSegment method in the Route class. It tests if the method removes the correct segment from the route.
- testGetPositionToAdd

        - Tests for getPositionToAdd method in the Route class by testing if the method return the correct index to add for adding a segment.
    - testClearRoute
        - Tests clearRoute method in the Route class by checking that instance variables including segmentBank and route should be empty.
    - testBuildToString
        - Tests buildRouteString method in Route class, making sure that the method returns the correct string representing the route object.
    - testAbbreviatedString
        - Test buildAbbreviatedRouteString method in the Route class, making sure the method return the correct abbreviated string that represents route object.

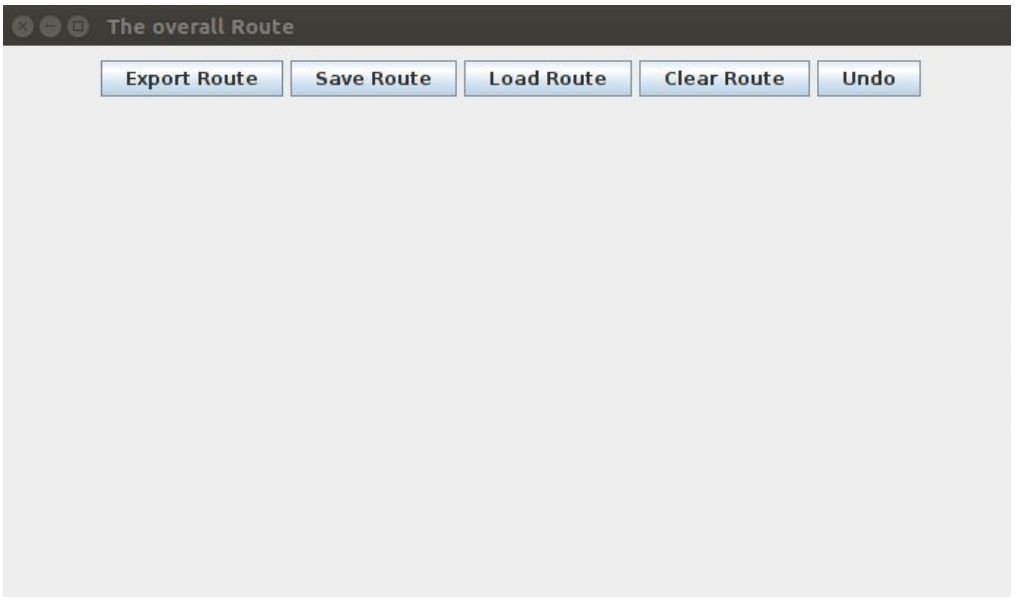
SegmentTest:

- testSegmentGetters
    - Tests if the getters methods of the segment class works
- testSegmentSetters
    - Tests whether the setters methods of the segment class works
- testSameButDifferentPath
    - Tests if two segments with same source and destination can be replaced if the second segment has different intermediate path.
- testCloneMethod
    - Tests whether the segment can be cloned correctly.

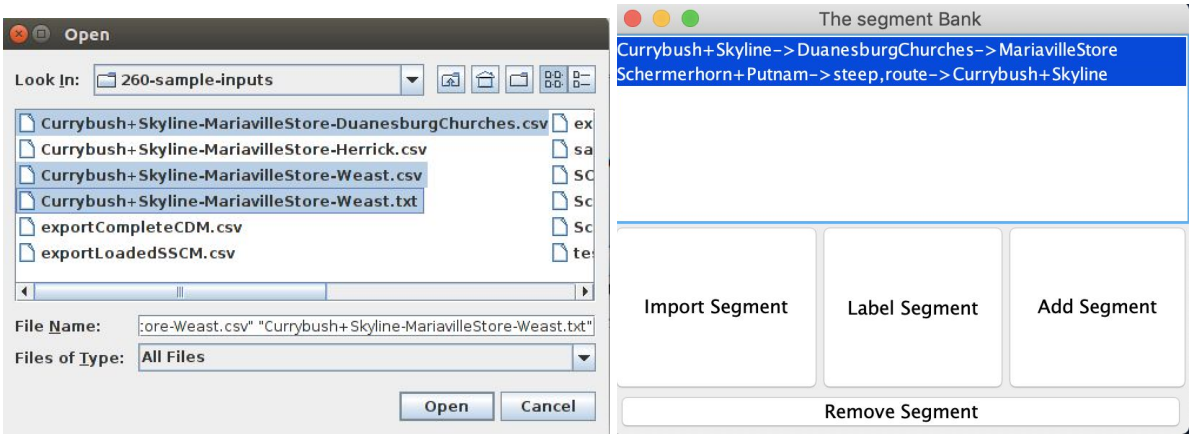
IX. Results

Following are features we have successfully implemented in Version 2.0 of the cuesheet application:

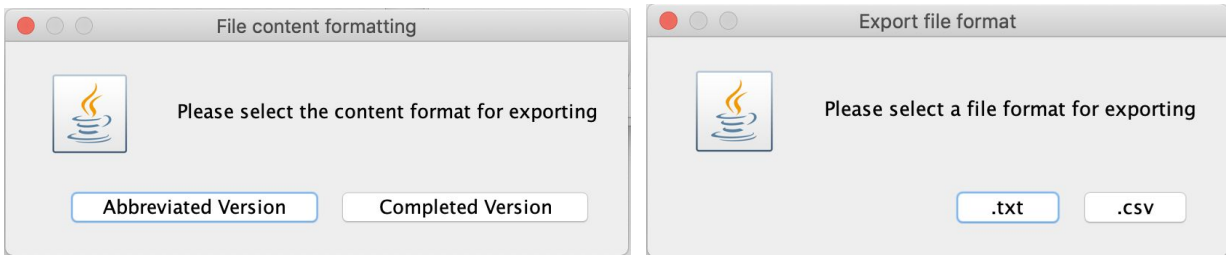
Firstly, we have added new buttons to our canvas display: Clear Route and Undo. The Clear Route button clears the canvas display and its initial uncleared state is added to our RouteState object, that stores a limited number of route states. Using this RouteState object, which contains an array of Routes, we can undo moves by indexing previous states of routes.



Multiple segment files in both .txt and .csv formats can be imported to the segment bank at a time. Later, multiple segments from the segment bank can be added to the canvas display if those selected segments can be added to the route.



Version 2.0 allows users to export routes in abbreviated and unabbreviated versions. When the user clicks on export, users first select what version (abbreviated or complete) of export they want, then choose what export file format (.txt or .csv) they want, they get the routes they build exported.



Using the buildAbbreviationRouteString() method, we are able to abbreviate both individual segment files as well as final routes built by users.

Type	Notes	Distance (miles) From Start
Start	Start of route	0
Left	Turn left onto Skyline Dr	0
Right	Turn right onto Duanesburg Churches Rd	4.44
Right	Turn right onto Batter St	7.37
End	End of route	10.07

Type	Notes	Distance (miles) From Start
Start	Start of route	0
Left	L:Skyline Dr	0
Right	R:Duanesburg Churches Rd	4.44
Right	R:Batter St	7.37
End	End of route	10.07

Figure X: Individual segment file abbreviated

Type	Notes	Distance (miles) From Start
Start	Start of route	0
Right	R:Mohawk-Huds	1.44
Left	L:Rice Rd	2.60
Left	L:Rice Rd	2.62
Right	R:Schermerhorn Rd	3.15
Right	R:Putnam Rd	4.13
Left	L:NY-159 E	5.73
Right	R:Currybush Rd	6.49
End	End of route	10.15

Figure X: Abbreviated final route built by user

X. Lessons Learned

Throughout the second version of the cuesheet application, our group has learning the following:

- All the working computer/macbook in the group should obtain the correct version of the Java compiler before coding and testing.
- Comprehensive and robust testing of the application was not straightforward and time-consuming.
- Implementing the functionalities of undo/redo was not as smooth as anticipated.

XI. Version 3.0

There are several crucial updates we hope (or wish) to implement in version 3.0. First of all, we would like to give users the ability to modify the actual segment content within the canvas display. The current version does not support any modification of the segment content, except export abbreviated version of the overall route.

We would also like to allow users to set different themes of the canvas with various coloring. As of now, version 2.0 cannot change its background color of the canvas.