Project 6 Write-Up

Kyle Douglas, Wyett MacDonald, Paige Hanssen, Tia Zhang

The goal of this project was to enhance the previous project by adding a number of new features. Before we were able do that that, however, we had to do a lot of refactoring from our last project. We ultimately decided to add a few new features including toggling comments, handling indentation, auto-closing parenthesis and brackets, and implementing a find menu item. In order to keep our IDE as elegant as possible we had to properly place all of these new implementations into their respective files. This meant including tabbing, indenting, and find in the EditMenuController, and adding auto-closing parenthesis and brackets in JavaCodeArea. Because commenting, indenting, and find all required buttons within the edit menu, whereas auto-closing parenthesis was a built-in addition, we thought this was the best way to go about splitting the features.

Refactoring from project 5 made our code more elegant, accurate and simpler in project 6. Unused fields and System.out.println statements were removed and all fields are either private or public final. Duplicate code found in files, especially CompilationController.java and FileController.java, has been removed and simplified to be in one place only. Some other refactoring that helped make our code elegant was removing Object[], which was an inelegant way of storing components. Our way is much more elegant because by only storing elements of the same type in an array, it prevents the need for typechecking and typecasting. It also is elegant because all the unused objects that were being stored in the Object arrays were not ever used, so now they do not exist. This allowed for some simplification further on, especially when setting the bindings for the menu items, because we were able to put all the menu items into one array together (instead of two different ones with multiple different kinds of objects), and then no longer need to check the class type for each object when setting bindings, and we no longer need to cast it to a MenuItem type. Our code was able to go from 18 lines to 5.

Additionally, we were able to refactor to make our code more elegant by using inheritance in both the JavaCodeArea and TabPaneInfo. We changed TabPaneInfo to be called CodeAreaTabPane, where we extended TabPane and included the two methods that existed in

TabPaneInfo. Finally, comments were added throughout our code to make it more readable and easier to understand. For example, the call() and handleCompileAndRunAction() methods are two complicated methods that previously had few to no comments, and so more comments were added to make it much easier for someone to understand and, therefore, more elegant.

The new features work as follows: If you wish to add some comments to your code, the toggle comments button in the menu item will comment out highlighted code if it isn't already commented out. Additionally, this button will also uncomment highlighted code if it was already highlighted. When you want to indent text, simply click on the button in the edit menu and the highlighted text will automatically be indented one tab to the right (4 spaces). If you wish to un-indent the text, go ahead and highlight the text you wish to un-indent and if there is at-least one tab in a line, it will be detabbed to the left (4 spaces). Entabbing and detabinng work very similarly to the indenting, except they transform spaces into tabs (as long as there are 4 consecutive spaces). If you are looking for something specific within the text file, now you can find it easily using our new find menu item. Click on the find button in the edit menu and a search bar will pop up in the toolbar. There you can type in whatever you are looking for and if there is matching text in your text file it will be found. For this feature, we hoped to highlight the found text, however we were only able to add it to an ArrayList, because it was more difficult than anticipated to highlight text within a CodeArea. This feature is not fully completed, however it functions properly, and finds the proper words to be highlighted. Our last addition was auto-closing parenthesis and brackets, which will occur automatically within the body of the text file. Whenever you type in an open parenthesis, a close parenthesis will automatically appear to the right of it and your cursor will go in between the two. In the context of an open bracket, whenever you type in a left bracket the IDE will automatically create a close bracket on the next line and will keep the cursor where it is. Unfortunately, we did not have time to fully perfect this, as it only works if the open bracket or open parenthesis being typed is the last element of the codeArea. Otherwise it does not add the closing item.

Finally, while refactoring did not add any new features, it did make some other features work or work better. The console in the program will now display both runtime errors (like NullPointerExceptions), as well as any program output, while also taking in user input. It also still displays any compile errors, or displays a “Compilation Successful!” message if compile completes. This was done by just redirecting the error stream from the process builder in the ProcessBuilderTask class. This got rid of a conditional to check if the command was to compile or to run, and all compile and runtime errors and messages were accessed through the input stream. Along with the runtime errors, the button disabling was fixed, in that the buttons in the toolbar will all be disabled if there are no tabs open, if there are tabs open, only Compile and Compile and Run will be enabled, and if the program is either compiling or running, Halt will be the only button available. This prevents the problem of being able to run multiple programs, because it is not possible to click the Compile or Compile and Run buttons when one program is already running.

One way our code is inelegant is when words are found and gathered into an ArrayList of words, we couldn’t figure out how to highlight the matched words because of the way text editing is done. Then, for our adding parentheses and brackets feature, it is not very elegant because it just appends the bracket and the parentheses to the end. We were unable to correct these two things because we couldn’t figure out how to fix them and we ran out of time to keep trying. Additionally, there was a bug fix that we did for the detab feature for replacing a selected area of text with “” instead of a space, that may not be completely elegant, but was necessary for the functionality of our feature. Finally, some of the code for our un-indenting code features doesn’t account for the first line, because the first line doesn’t contain a new line character, so the new line had to be handled separately. However, by doing it this way, we were able to avoid using a for loop to go through the lines, which we believed was better to do.

We decided to split the work up four ways, with Paige working on refactoring, Wyett working on find, Tia working on toggling comments and indentation, and Kyle working on auto-closing parenthesis and brackets. While this was how we originally split up our work, some finished earlier than others and helped with other portions of the project. Overall the work was split up fairly evenly.