4-2 Milestone 3 Narrative

Michael P Clisbee

CS 499

Southern New Hampshire University

**Briefly describe the Artifact. What is it? When was it created?**

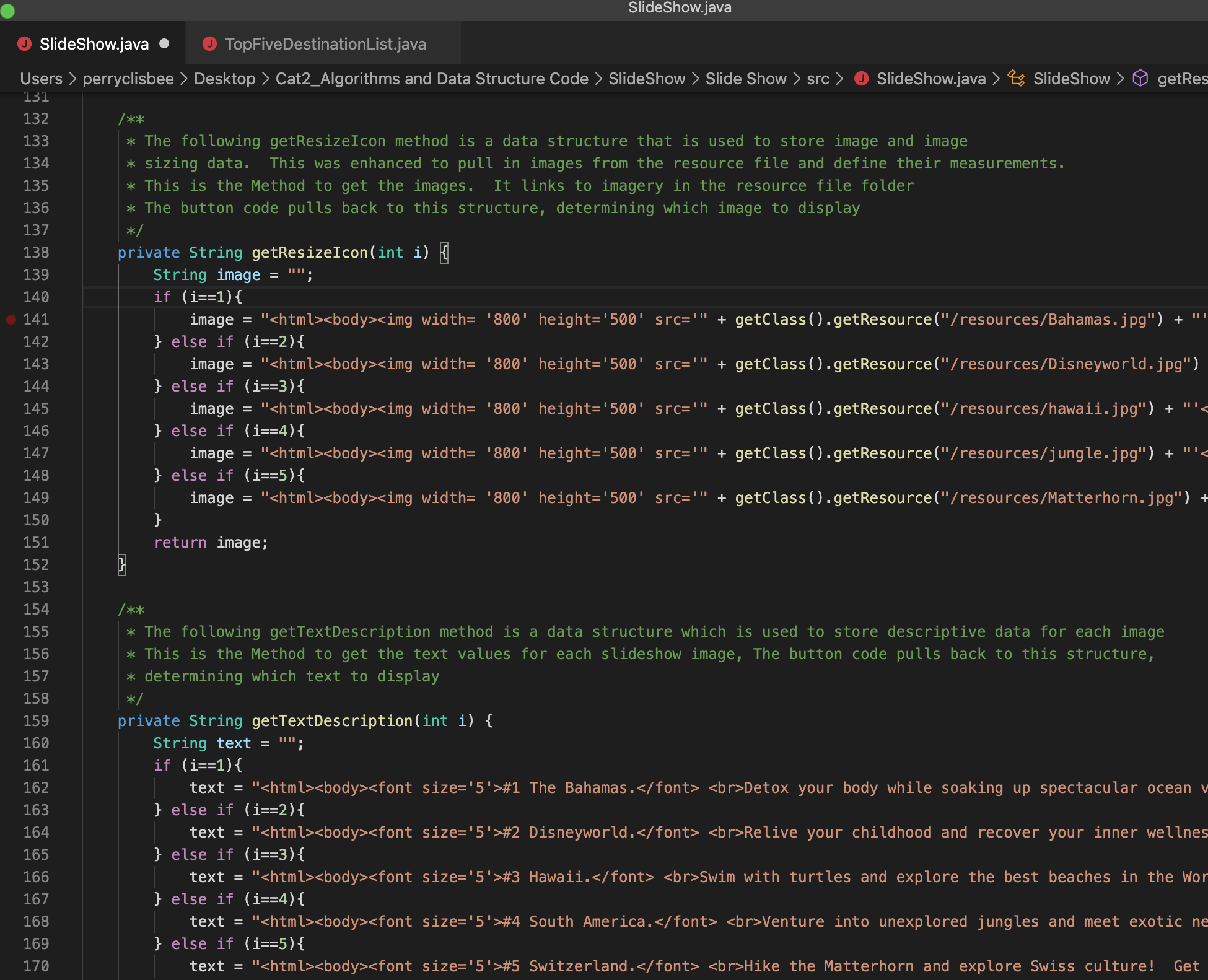
For Artifact Two in Category Two: Algorithms and Data Structures, I selected work done in CS250 – Software Development LifeCycle (SDLC), which focused on the Agile team roles and how their teamwork resulted in an overall rounded functional Java application where the code included added functionality and imagery for a better user experience. My enhancement plan included expanding the current API complexity of my TopFiveTravelDestination.java and SlideShow.java files with added resource folder imagery to give the user a better visual experience viewing destination information through a simple GUI that is user friendly. This was done through the enhancement of current Java code to include better code documentation to elaborate on code functionality resulting in minimal errors and a fluid transition between screens. This demonstrated knowledge and skills gained in the use of Java coding to create simple applications that the user can easily manipulate but not corrupt. Code was altered and tested using Java IDE, XCode application and Terminal applications for Mac.

The artifact is a compilation of five vacation locations displaying an image along with a description that is supposed to entice the client to go there. The artifact includes a TopFiveDestinationList.java and SlideShow.Java file that are two variations on the same project, that implements JFrames, JLabels and JPanels along with Button Controls to assist in maneuver.

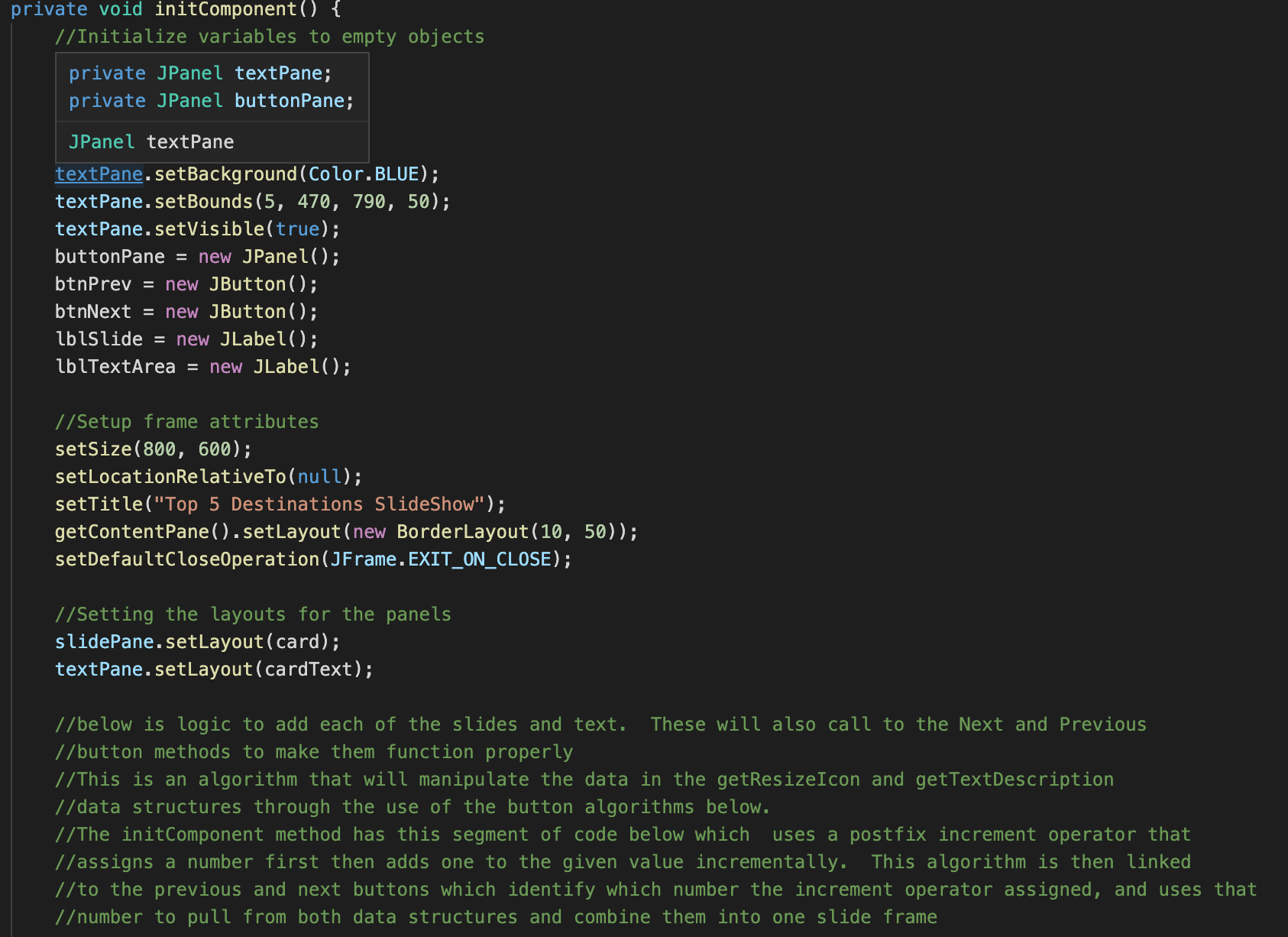
**Justify the inclusion of the Artifact in your ePorfolio. Why did you select this item? What specific components of the Artifact showcase your skills and abilities in software development? How was the artifact improved?**

The TopFiveDestinationList.java and SlideShow.Java files were included in the ePorfolio as both showed the potential to demonstrate multiple displays of code functionality for the creation of working button controls, frame and panel attributes, and the insertion of images and text using if/else loops, the addition of colored backgrounds, borders, and the conversion of these files into a runnable jar file. The files included base code structures to allow for the insertion of images and text, with no declared variables to allow for panes and framework. The artifact was improved by declaring these variables, initializing them, setting up frame attributes and panel layouts, adding functionality for the previous and next buttons, and adding imagery and text with code to allow them to be displayed through a method that included if/else loops.

When modifying the artifact, I once again did a lot of code cross referencing online through stackoverflow.com and other sites to better understand the purpose of the different assert methods, and how to implement them. I used available software such as XCode to view and alter as necessary. This code was then placed into a downloaded version of Eclipse IDE for Mac, where I could further alter code, then create a runnable jar file in addition to the saved java file.

The code below demonstrates the insertion of imagery and text linked to images in the resource file, using if/else statements to determine which image to display. This is code from SlideShow.java. The two data structures shown are used to store image and image sizing data. This was enhanced to pull in images from the resource file and define their measurements. These are the Methods to get the images and text. It also links to imagery in the resource file folder. The button code pulls back to this structure, determining which image and text to display 

The code below displays Previous and Next Button algorithms, along with logic that is uses to determine the image and text slides to display:



Text

Description automatically generated

Text

Description automatically generated

The algorithm created to add each of the slides and text as seen here sets parameters for the Previous and Next buttons, adding to their functionality. this segment of code which uses a postfix increment operator that assigns a number first then adds one to the given value incrementally. This algorithm is then linked to the previous and next buttons which identify which number the increment operator assigned and uses that number to pull from both data structures and combine them into one slide frame. The initComponent method has this segment of code below as an algorithm which identifies which number the increment operator assigned, by using both the previous and next buttons, determine the line number to pull from both data structures and combine them into one slide frame. Previous Button Functionality Is linked to the previous code to add functionality. It will use the increment operator to go back one number from the currently assigned one, then go back to the previous pane.

**Graphical user interface, website

Description automatically generated**

Above are examples of the results of the code, with various vacation destinations displayed, **Graphical user interface

Description automatically generated**along with descriptive text, a title for the program, and functional buttons allowing the client to scan through all the slideshow pages.

Graphical user interface, text, application

Description automatically generated

The code above shows that Eclipse IDE environment was used to enhance data structure code for the TopFiveDestinationList.java file. This was also used to convert both java files to runnable jar files that you can open without the necessity of using an IDE or other environment.

**Did you meet the course objectives you planned to meet with this enhancement in Module One?**

The enhancements increased functionality in several different areas as determined in the Code Review by identifying, enhancing, and commenting on data structures in the code and their associated algorithms. Data structures were also used to store and organize data. The identified algorithms were used to manipulate the data in their associated structures.

**The two data structures** in the **SlideShow.java file** were identified as:

a. The getResizeIcon method, which is used to store image and image sizing data. This was enhanced to pull in images from the resource file.

b. The getTextDescription method, which is used to store descriptive data for each image.

**The two associated algorithms** in the **SlideShow.java file** were identified as:

a. The initComponent method, which has a segment of code which uses a postfix increment operator that assigns a number first then adds one to the given value incrementally. This algorithm is pulls from the two data structures to determine which data to pull from it and display.

b. The second associated algorithm originated from the same initComponent method, where the code has the button action identify which number the increment operator assigned and used that to pull from both data structures and combine them into one slide frame. The button algorithm determines whether to subtract one number from the postfix increment operator or add one number, depending on the button activated, then go to the next or previous slide and text pane.

**Reflect on the process of enhancing and/or modifying the Artifact. What did you learn as you were creating it and improving it? What challenges did you face?**

When modifying the artifact, I once again did a lot of code cross referencing online through stackoverflow.com and other sites to better understand the purpose of the different assert methods, and how to implement them. For this class, I could not access a virtual environment previously supplied by SNHU, so instead again used available software such as XCode to view and alter as necessary. This code was then placed into a downloaded version of Eclipse IDE for Mac, where I could further alter code, then create a runnable jar file in addition to the saved java file.