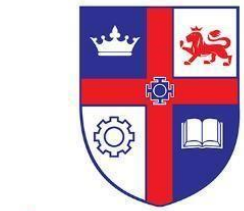
LONDON METROPOLITAN UNIVERSITY



Islington college
(इस्लिङ्टन कलेज)

CS5003 Data Structure and Specialist Programming

30% Individual Coursework

2023-24 Autumn

Student Name: Trisana Gurung

London Met ID: 22067339

College ID: NP01AI4A220055

Group: L2AI2

Assignment Due Date: Friday, January 12, 2024

Assignment Submission Date: Friday, January 12, 2024

Words Count: 4811

I confirm that I understand my coursework needs to be submitted online via MySecondTeacher under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and marks of zero will be awarded.

Table of Contents

Table of Figures

Table of Tables

1. Introduction:

1.1. A brief introduction:

The project uses Java, Draw.io, and Apache NetBeans IDLE to make a software system called “Dramaflix” which gives information on dramas in Korea which is popular. It helps to give a time update on the drama.

It makes us aware of the actual situation and the history of the drama that is represented through the data. It has implemented merge sort and binary search. Also, has an add, update, and delete button that can be used in real-time to get and see the information of the drama. The software system helps to provide the user with the experience and features of the drama listing in Korean dramas that have captivated audiences globally. Through the help of the add, update, and delete it helps to keep the information of the current scenario of Korean dramas.

It is a programming language that helps to complete the task with the help of draw.io to create the architecture system and use Java to develop in Apache NetBeans IDLE.

1.2. Aims and Objectives:

It main aim of Dramaflix is to give information on Korean drama and make its growing demand. It is user-friendly and gives a detailed list of the drama that is shown in real time for the user experience.

- To provide updated information on drama
- Implementing Merge Sort and Binary Search helps to give efficient data
- It is a user-friendly interface
- For accurate present information
- Adding new drama, updating the drama, and deleting outdated drama records make it user-friendly.

2. Algorithms Utilized

2.1. Algorithm 1

Algorithm:

Algorithms solve problems sequentially using a defined action producer. The term "data processor" refers to the role played by automated systems and is commonly accepted. Algorithms typically begin with instructions that transform initial inputs into words in certain ways (Gillis, 2023).

Merge sort:

Merge sort is a sorting method that divides an array into small arrays by sorting each small array and merging them back into the final form of the array. The operation continues until the entire array has been sorted. Merge sort, like the rapid sort algorithm, sorts the elements using a divide-and-conquer strategy (GeeksforGeeks, 2023).

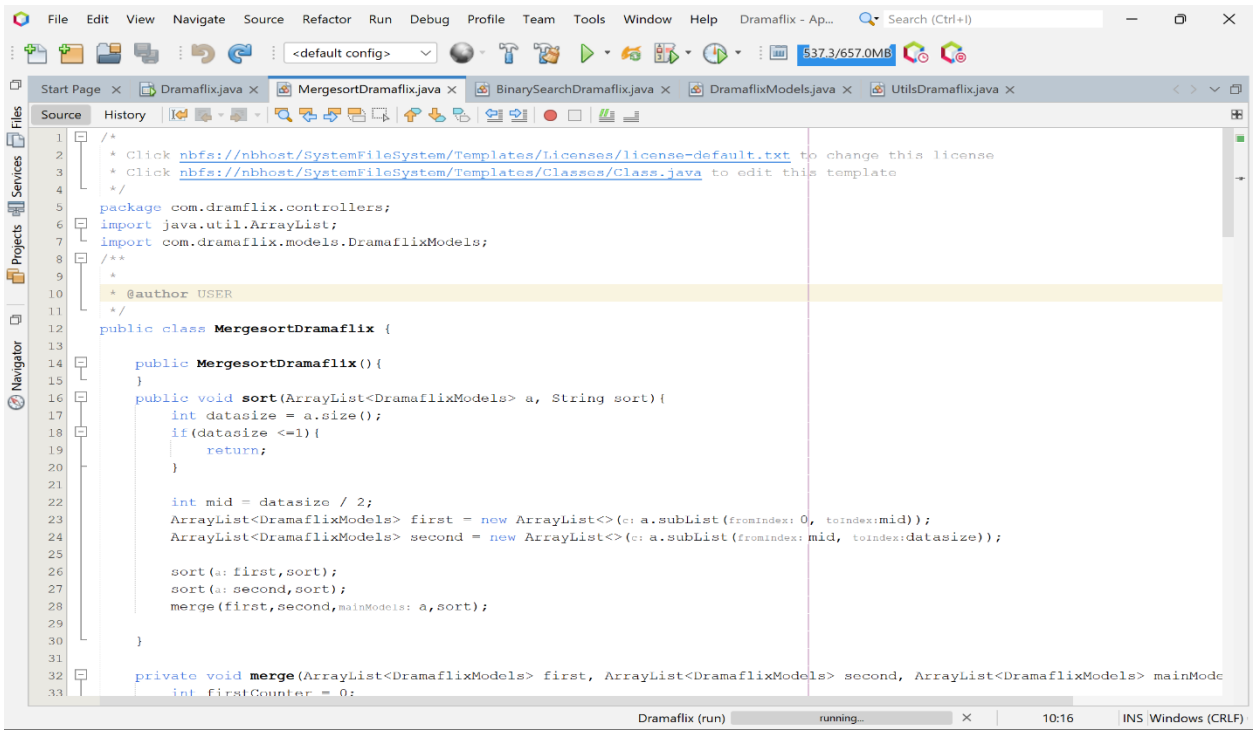


Figure 1 MergesortDramaflix

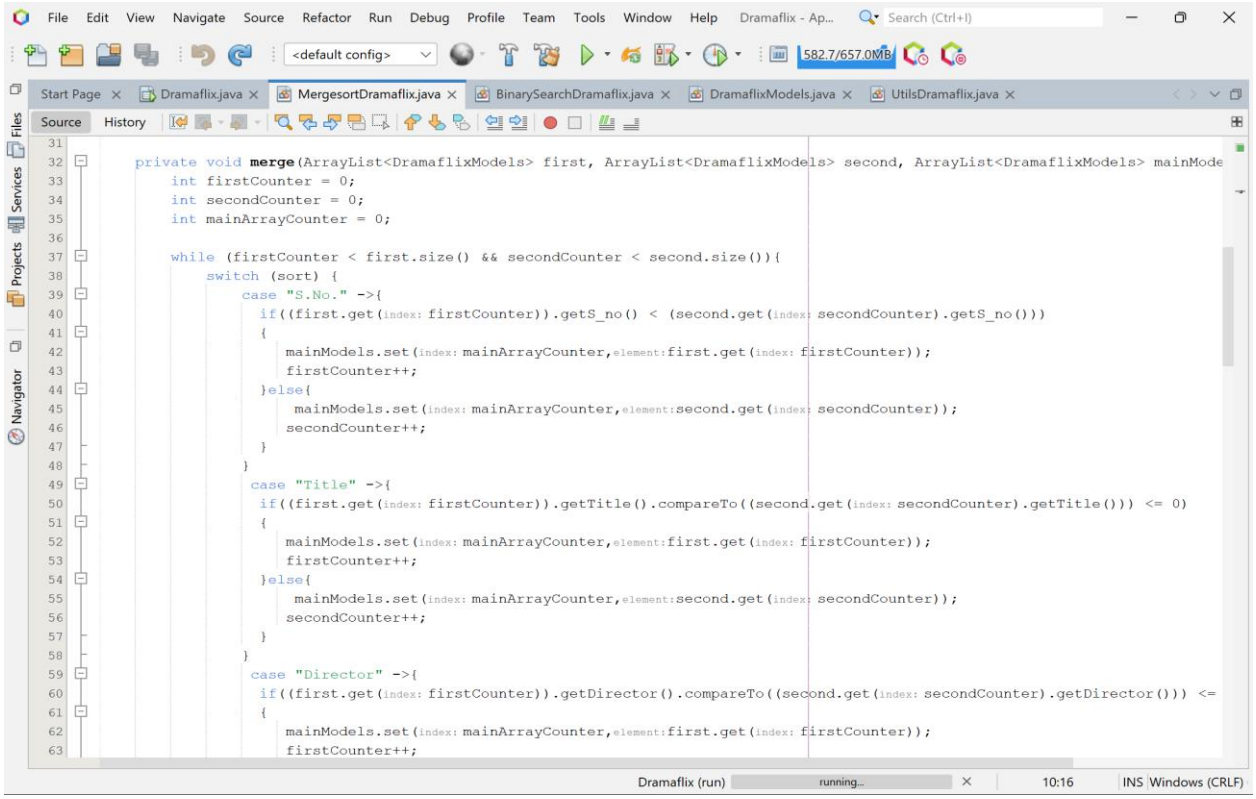


Figure 2 MergesortDramflix

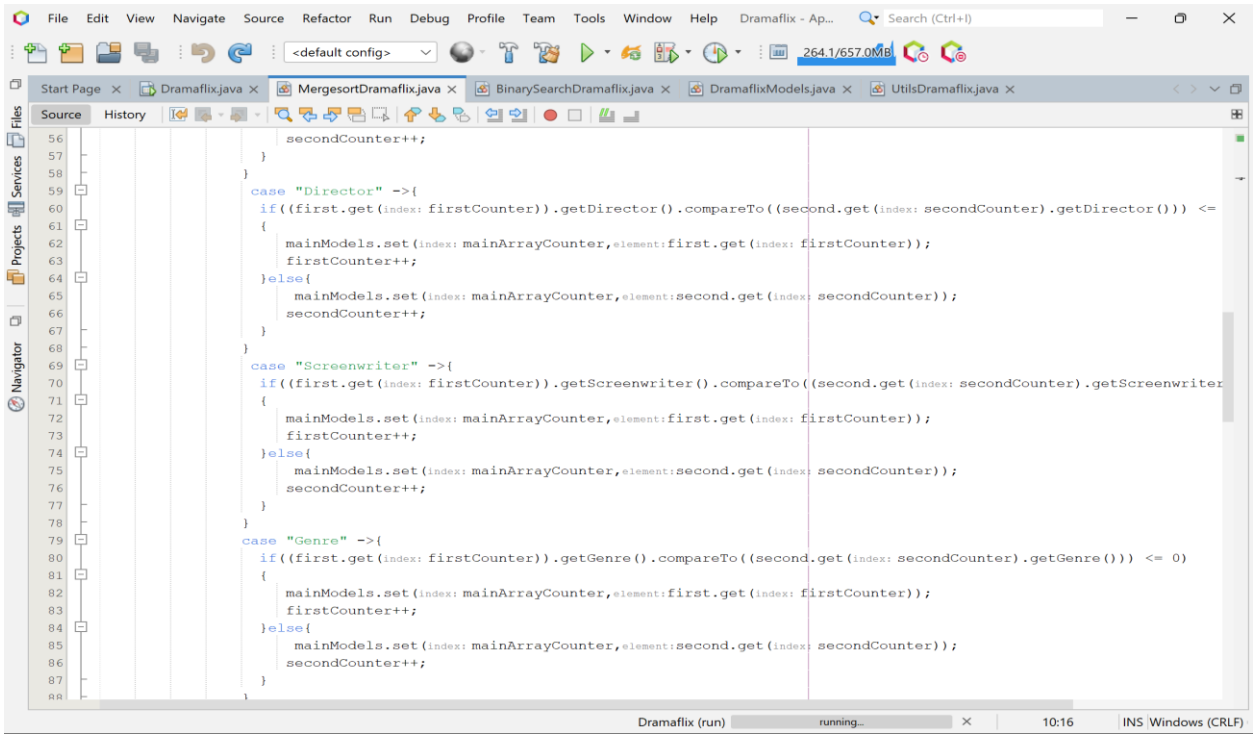
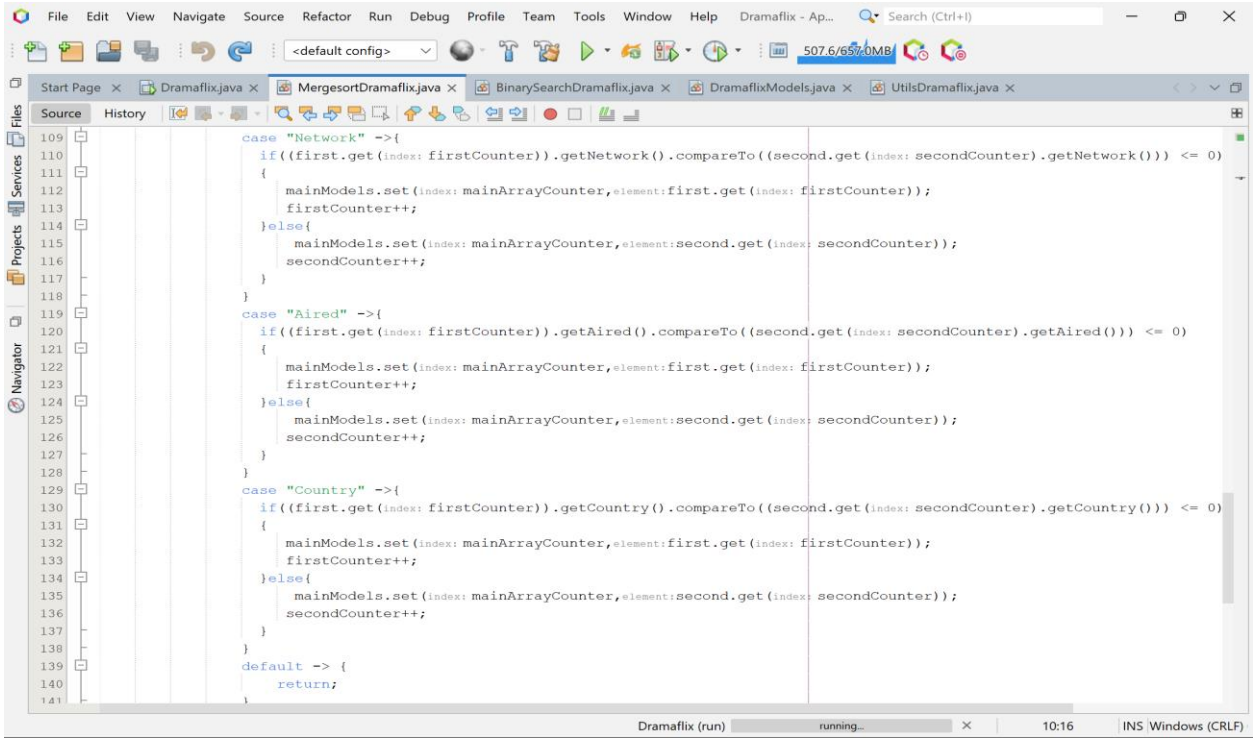
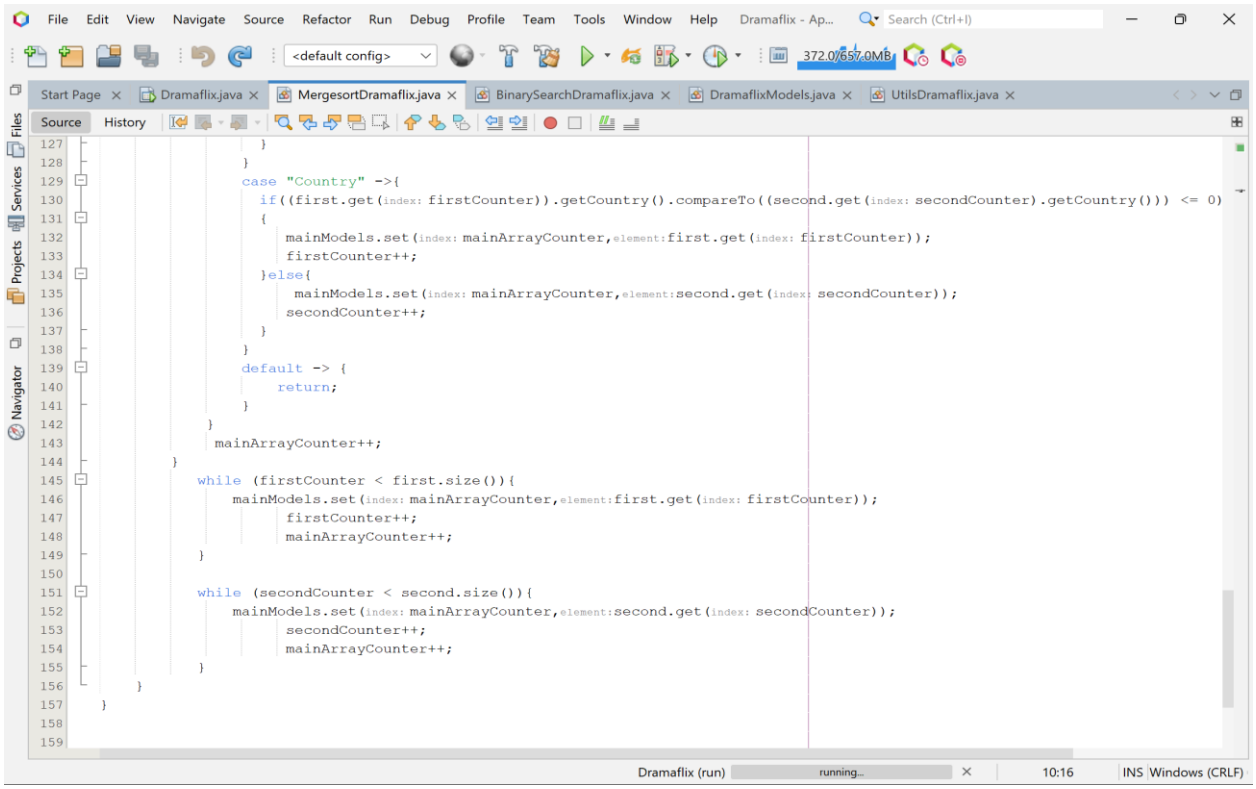


Figure 3 MergesortDramaflix



```
case "Network" ->{
  if((first.get(index: firstCounter)).getNetwork().compareTo((second.get(index: secondCounter).getNetwork())) <= 0)
  {
     mainModels.set(index: mainArrayCounter,element:first.get(index: firstCounter));
     firstCounter++;
  }else{
     mainModels.set(index: mainArrayCounter,element:second.get(index: secondCounter));
     secondCounter++;
  }
}
case "Aired" ->{
  if((first.get(index: firstCounter)).getAired().compareTo((second.get(index: secondCounter).getAired())) <= 0)
  {
     mainModels.set(index: mainArrayCounter,element:first.get(index: firstCounter));
     firstCounter++;
  }else{
     mainModels.set(index: mainArrayCounter,element:second.get(index: secondCounter));
     secondCounter++;
  }
}
case "Country" ->{
  if((first.get(index: firstCounter)).getCountry().compareTo((second.get(index: secondCounter).getCountry())) <= 0)
  {
     mainModels.set(index: mainArrayCounter,element:first.get(index: firstCounter));
     firstCounter++;
  }else{
     mainModels.set(index: mainArrayCounter,element:second.get(index: secondCounter));
     secondCounter++;
  }
}
default -> {
    return;
}
```

Figure 4 MergesortDramaflix



```
case "Country" ->{
  if((first.get(index: firstCounter)).getCountry().compareTo((second.get(index: secondCounter).getCountry())) <= 0)
  {
     mainModels.set(index: mainArrayCounter,element:first.get(index: firstCounter));
     firstCounter++;
  }else{
     mainModels.set(index: mainArrayCounter,element:second.get(index: secondCounter));
     secondCounter++;
  }
}
default -> {
    return;
}
mainArrayCounter++;
while (firstCounter < first.size()){
   mainModels.set(index: mainArrayCounter,element:first.get(index: firstCounter));
     firstCounter++;
     mainArrayCounter++;
}
while (secondCounter < second.size()){
   mainModels.set(index: mainArrayCounter,element:second.get(index: secondCounter));
     secondCounter++;
     mainArrayCounter++;
}
```

Figure 5 MergesortDramaflix

Figure 6 MergesortDramaflix GUI

Step 1: Start

Step 2: Input the array List Dramaflixmodels in the unsorted list of “a”

Step 3: Then again input the string by “sort”

Step 4: Then sort the output array list in DramaflixModels

Step 5: Create the size if “a” is 1 or less then to be returned to the already sorted “a”

Step 6: To calculate the midpoint(‘mid’) of “a”

Step 7: Create new two array lists first and second from the splitting of “a” and ”mid”

Step 8: Calling the mergeSortDramflix to recursive from the first and second

Step 9: Calling the merge of first, second, a, and sort to sort and merge them into a

Step 10: After merging the input of first is in the first sorted list, the second is in the second sorted list, mainModels is the merged list stored in the result, and sort is in the string

Step 11: Through the counters from firstcounter, secondcounter, and mainArrayCounter into 0.

Step 12: The size of the first and second is bigger than the firstCounter and secondCounter which are compared based on sort that updates the mainModels with smaller element incremented counters accordingly.

Step 13: Copying remaining element first to mainModels and second to mainModels.

Step 14 The array list is sorted based on mainModels

Step 15: run the code

Step 16:Open the GUI

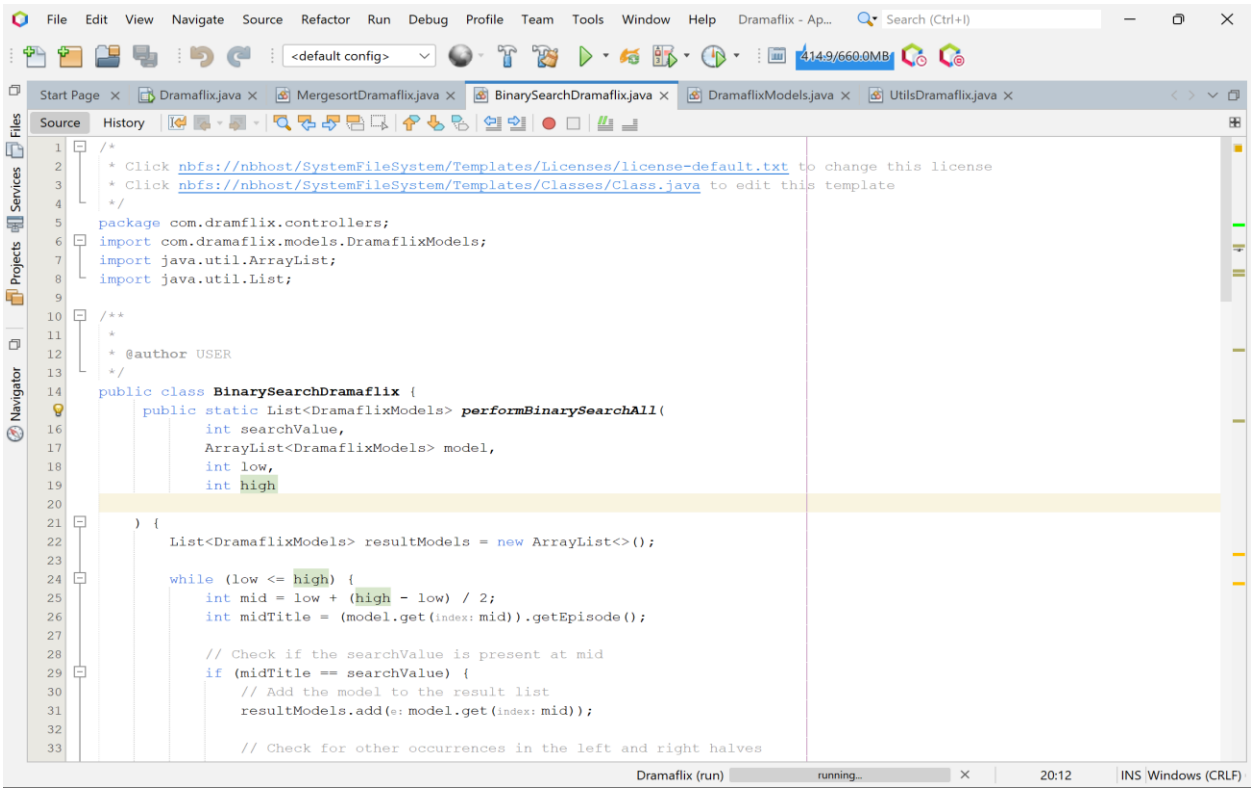
Step 17: Selecting Combo box and choose and click the button to sort

Step 18: The table of titles will come in alphabetical order of title then again to S.N. to come in numbers lowest to highest

Step 19:Stop

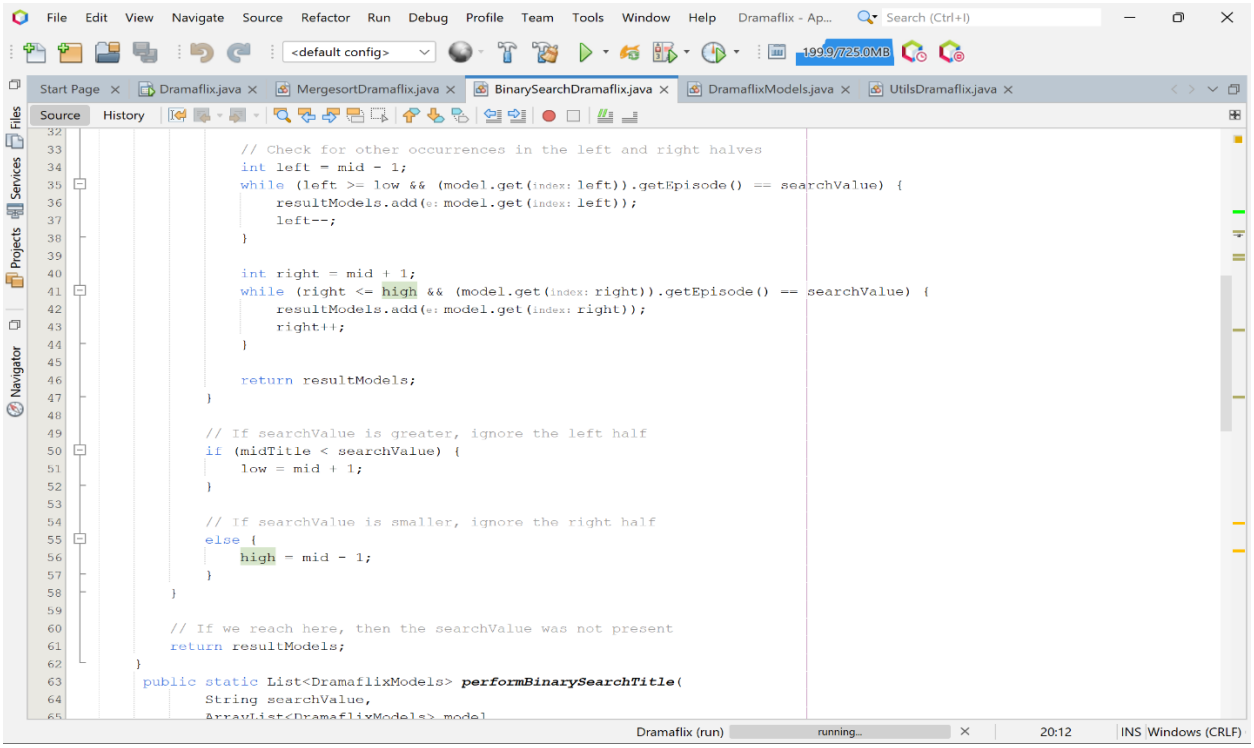
Binary search Algorithm:

Binary search is a search technique that uses an array to sort and divide the search results continuously. It is useful to look for the middle member of an array when comparing the target. Only if the elements are in the list will the process be successful, and the element will be returned to its original location; otherwise, the search will fail (GeeksofGeeks, 2023).



```java
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package com.dramflix.controllers;
import com.dramaflix.models.DramaflixModels;
import java.util.ArrayList;
import java.util.List;

/**
 *
 * @author USER
 */
public class BinarySearchDramaflix {
    public static List<DramaflixModels> performBinarySearchAll(
            int searchValue,
            ArrayList<DramaflixModels> model,
            int low,
            int high

    ) {
        List<DramaflixModels> resultModels = new ArrayList<>();

        while (low <= high) {
            int mid = low + (high - low) / 2;
            int midTitle = (model.get(index: mid)).getEpisode();

            // Check if the searchValue is present at mid
            if (midTitle == searchValue) {
                // Add the model to the result list
                resultModels.add(e: model.get(index: mid));

                // Check for other occurrences in the left and right halves
```

Figure 7 BinarySearchDramflix



```java
                // Check for other occurrences in the left and right halves
                int left = mid - 1;
                while (left >= low && (model.get(index: left)).getEpisode() == searchValue) {
                    resultModels.add(e: model.get(index: left));
                    left--;
                }

                int right = mid + 1;
                while (right <= high && (model.get(index: right)).getEpisode() == searchValue) {
                    resultModels.add(e: model.get(index: right));
                    right++;
                }

                return resultModels;
            }

            // If searchValue is greater, ignore the left half
            if (midTitle < searchValue) {
                low = mid + 1;
            }

            // If searchValue is smaller, ignore the right half
            else {
                high = mid - 1;
            }
        }

        // If we reach here, then the searchValue was not present
        return resultModels;
    }
    public static List<DramaflixModels> performBinarySearchTitle(
            String searchValue,
            ArrayList<DramaflixModels> model
```

Figure 8 BinarySearchDramaflix

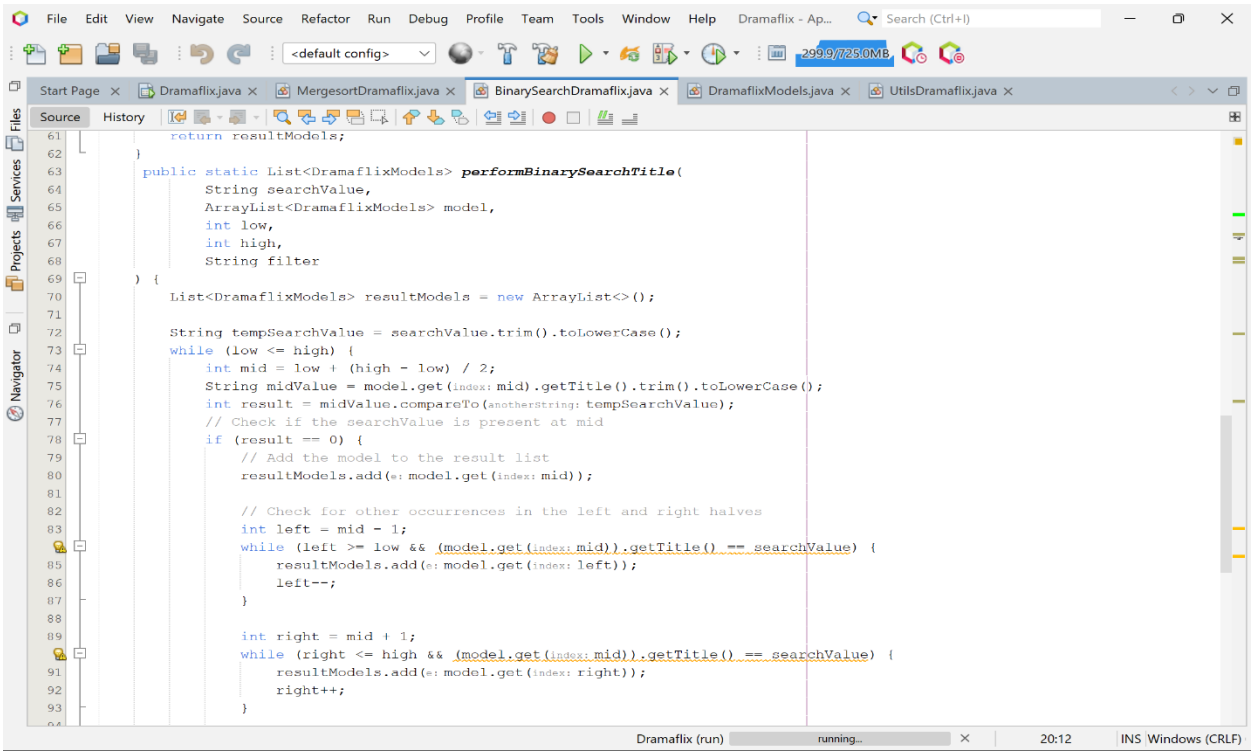


Figure 9 BinarySearchDramaflix

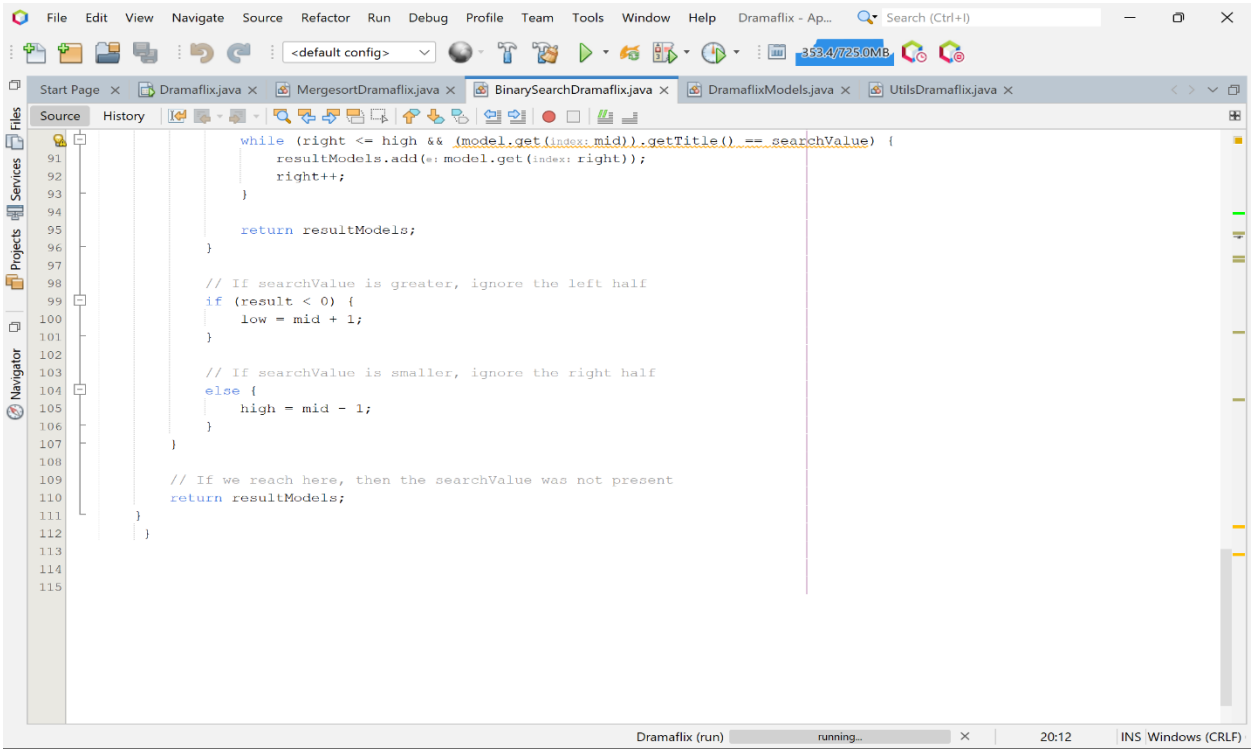


Figure 10 BinarySearchDramaflix

Figure 11 BinarySearchDramaflix

Step 1: Start

Step 2: input the searchValue for episode int and model for array list DramaflixModels

Step 3: Starting index low for int and Ending index high for int

Step 4: List of DramaflixModels output resultModels

Step 5: Empty the list for resultModels and calculate low is less or equal to the high by the middle index mid to obtain the episode number.

Step 6: If midEpisode is equal to the search value it will be added mid to resultModels from Dramaflix.

Step 7: Then check left to mid-1 and right to mid + 1 where left is equal or grater to low and and right is less than or equal to high which adds the index of DramaflkixModel to resultModels.

Step 8: Then midEpispde is greater than searchValue, which is set high to mid – 1 and if it is less the searchValue, which is set low to mid and the loop will continue until the resultModels are matched into DramaflixModels.

Step 9: input the searchValue for title String and model for array list DramaflixModels

Step 10: Starting index low for int, String to filter, and Ending index high for int

Step 11: List of DramaflixModels output resultModels

Step 12: Empty the list for resultModels and calculate low is less or equal to the high by the middle index mid to obtain the episode number.

Step 13: If midEpisode is equal to the search value it will be added mid to resultModels from Dramaflix.

Step 14: Then check left to mid-1 and right to mid + 1 where left is equal or grater to low and and right is less than or equal to high which adds the index of DramaflkixModel to resultModels.

Step 15: Then midEpispde is greater than searchValue, which is set high to mid – 1 and if it is less the searchValue, which is set low to mid and the loop will continue until the resultModels are matched into DramaflixModels.

Step 16:Run the code

Step 17: Select the combo box title and search the title name and the display will be shown where the history of the drama will be seen.

Step 18: Stop

2.2. Algorithm 2

Flowchart:

A flowchart depicts a step-by-step procedure, including sequences and decisions. This application enables quick visualization and analysis of complex ideas. The flowchart is widely used in numerous industries for documenting, planning, processing, and visualizing tasks (Asana, 2023).

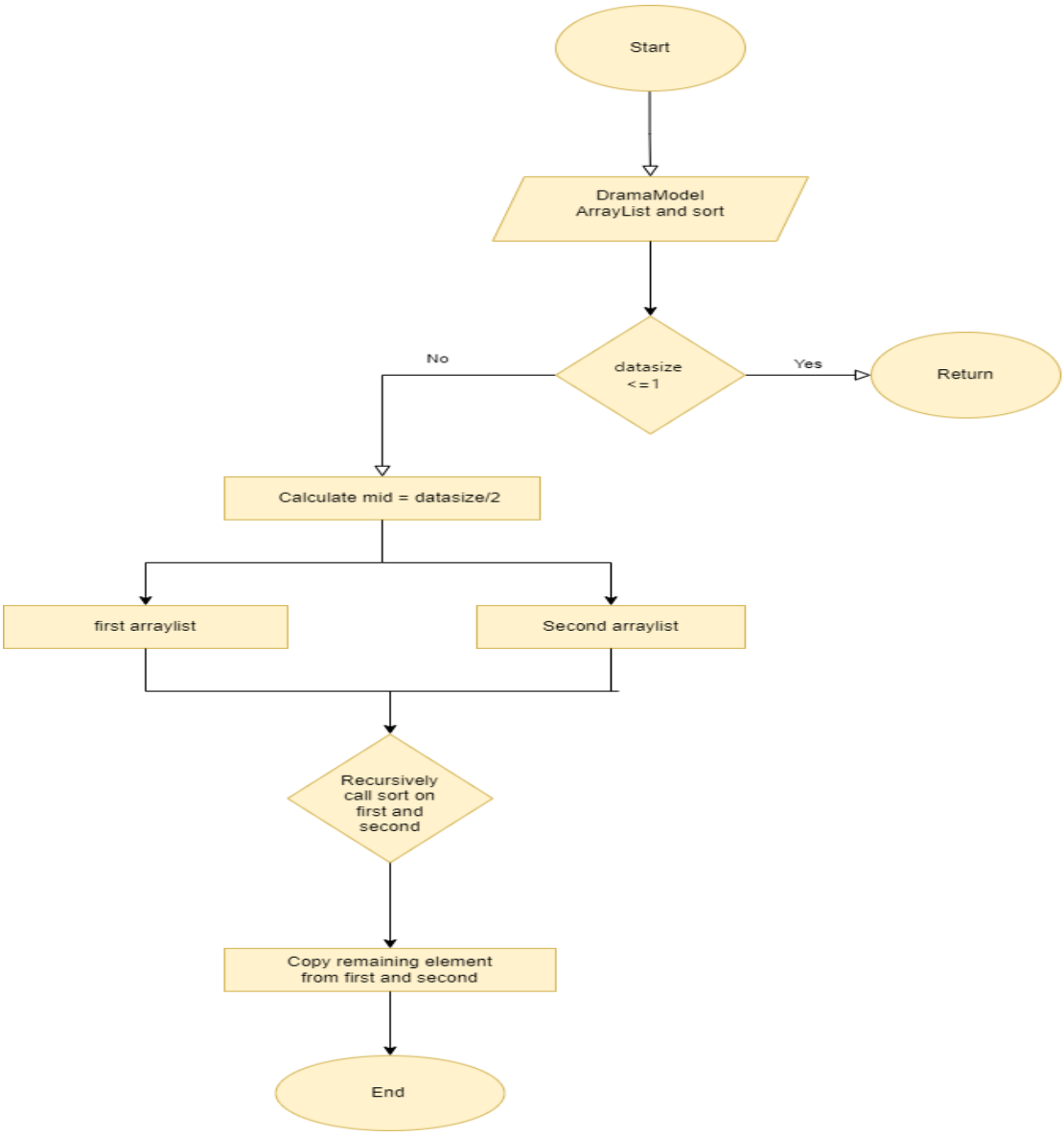


Figure 12 MergesortDramflix

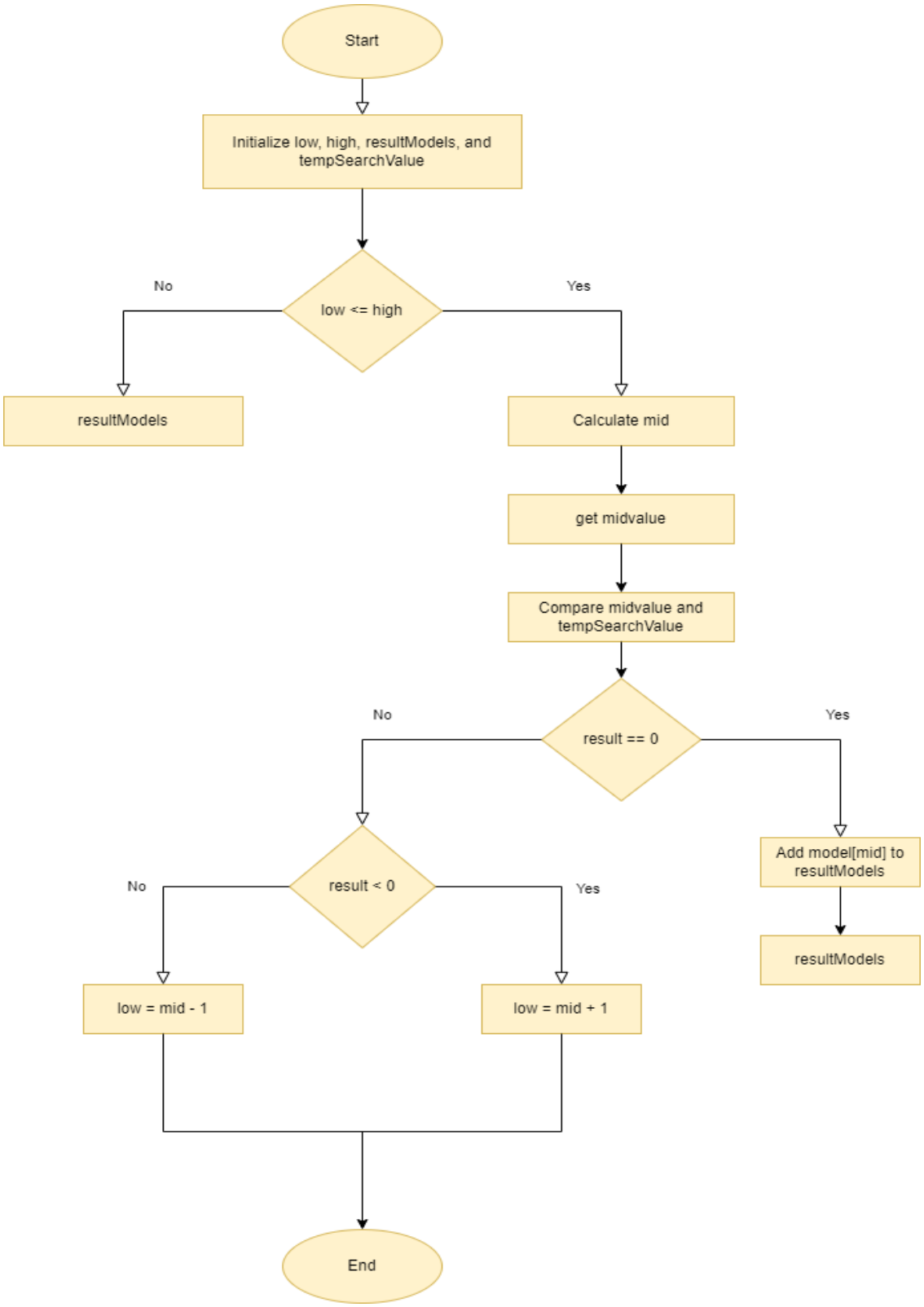


Figure 13 ***BinarySearchDramaflix***

Pseudocode of merge sort:

CREATE a class MergesortDramaflix

DO

CREATE an ArrayList DramaflixModel where it is declared a and string is sort

IF the datasize is geater and equal to 1

RETURN

CREATE an instance variable mid equals to datasize/2

CREATE array list first of sublist.a from Index to Index mid

CREATE array list second of sublist.a from mid Index to datasize

DECLARE mergesort first and sort

DECLARE mergesort second and sort

DECLARE mergesort first, second, a and sort

DECLARE Merge(first: ArrayList<DramaflixModels>, second: ArrayList<DramaflixModels>, mainModels: ArrayList<DramaflixModels>, sort: String)

DECLARE firstCounter = 0

DECLARE secondCounter = 0

DECLARE mainArrayCounter = 0

WHILE firstCounter < size of first and secondCounter < size of second

```
            SWITCH sort

           CASE "S.No."

        IF first[firstCounter].getS_no() < second[secondCounter].getS_no()

        SET mainModels[mainArrayCounter] = first[firstCounter]

         DECLARE firstCounter++

        ELSE

            SET mainModels[mainArrayCounter] = second[secondCounter]

            DECLARE secondCounter++

           DECLARE mainArrayCounter++

        WHILE firstCounter < size of first

            SET mainModels[mainArrayCounter] = first[firstCounter]

            DECLARE firstCounter++

            DECLARE mainArrayCounter++



        WHILE secondCounter < size of second

            SET mainModels[mainArrayCounter] = second[secondCounter]

            DECLARE secondCounter++

            DECLARE mainArrayCounter++

END DO
```

Pseudocode of Binary search:

CREATE a class BinarySearchDramaflix

DO

CREATE static method perfpormBinarySearch that has Search value be the object, array list DramaflixModels be the model initializing low and high and string to field in the List

SET resultModels = empty ArrayList<DramaflixModels>

WHILE low <= high:

DECLARE mid = low + (high - low) / 2

DECLARE midValue = getFieldValue(model[mid], field)

SET result = compareValues(midValue, searchValue)

IF result == 0:

DECLARE resultModels.add(model[mid])

SET expandResultList(model, low, mid - 1, resultModels, field)

SET expandResultList(model, mid + 1, high, resultModels, field)

RETURN resultModels

IF result < 0:

DECLARE low = mid + 1

ELSE

DECLARE high = mid - 1

RETURN resultModels

SET static method expandResultList(

DECLARE array list model DramaflixModels instance variable start and end dor string field with resultmodels list DramaflixModels

WHILE start <= end:

// Add adjacent elements to the result list

SET resultModels.add(model[start])

SET start++

DECLARE static method getFieldValue(

SET model: DramaflixModels,

SET field: String

SWITCH field:

CASE "Title":

RETURN model.getTitle().trim().toLowerCase()

CASE "Episode":

RETURN model.getEpisode()

SET default:

RETURN null

DECLARE static method compareValues(value1: Object,value2: Object)

IF value1 is Comparable and value2 is Comparable:

RETURN value1.compareTo(value2)

RETURN 0

DECLARE static method performBinarySearchAll(

SET searchValue: int,

DECLARE model: ArrayList<DramaflixModels>, low: int, high: int) List<DramaflixModels>

RETURNperformBinarySearch(searchValue, model, low, high, "Episode")

DECLARE static method performBinarySearchTitle(

SET searchValue: String,

DECLARE model: ArrayList<**DramaflixModels**>, low: int, high: int) List<DramaflixModels>

RETURN performBinarySearch(searchValue.trim().toLowerCase(), model, low, high, "Title")

END DO

3. Class Diagram

A software model called a class diagram is used to draw the design and explain the relationships between the classes. It displays the relationships and classifications between names and attributes. Since it's a high-level software model, you can view the class diagrams without having to view the code (Tutorialpoints, 2023).

3.1. Overall Diagram

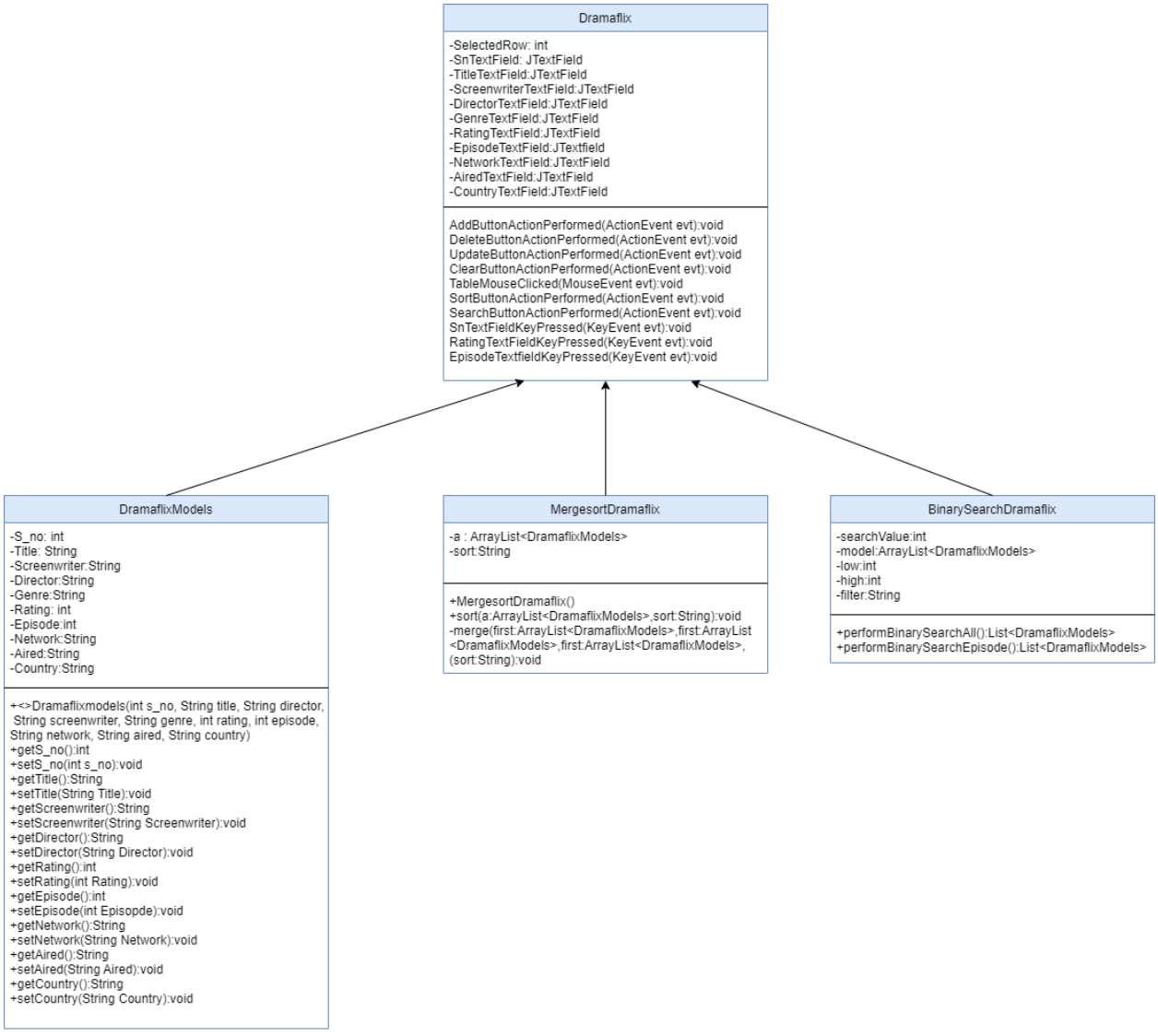


Figure 14 overall class diagram

3.2. Individual Diagrams

3.2.1. Dramaflix Class 1

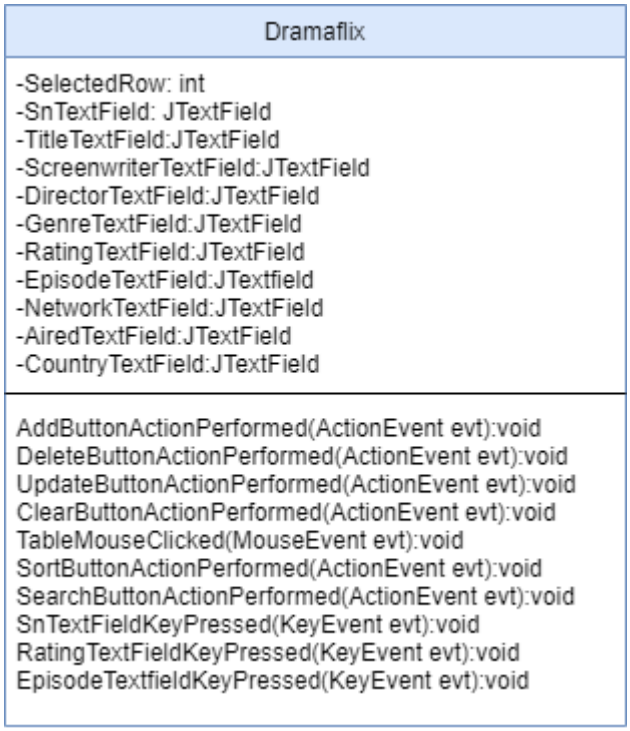


Figure 15 Dramaflix class diagram

3.2.2. DramaflixModels Class 2

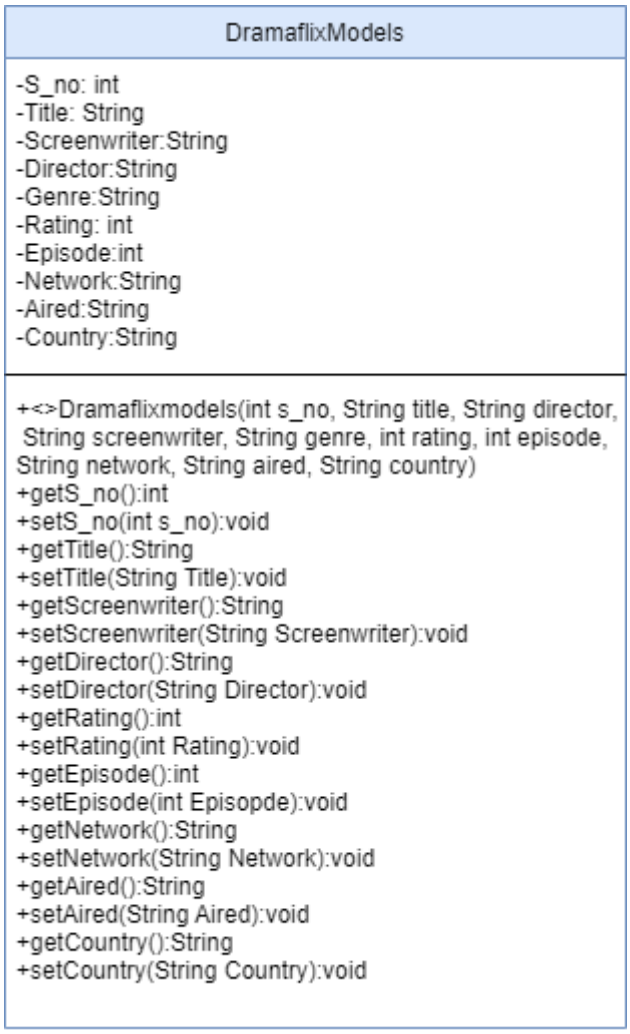


Figure 16 DramaflixModels class

3.2.3. MergesortDramaflix class 3

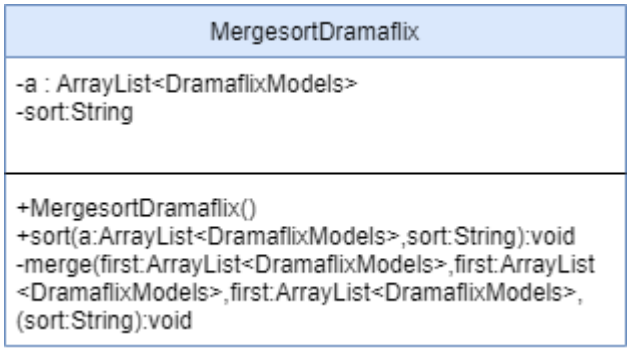


Figure 17 MetgesortDramaflix class

3.2.4. BinarySearchDramaflix

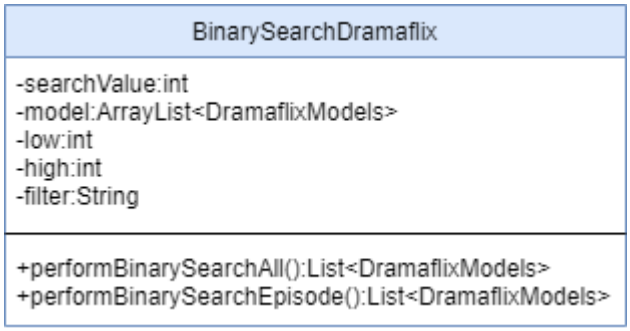


Figure 18 BinarySearchDramaflix diagram

4. Method Description

The Java code block's specified action is made clearer by the method description. The arguments or values in the methods aid in the method's execution. Thus, it aids in the written instructions, such as those included in the technique. For its method support, the method description has a special name of its own (Simplilearn, 2023).

4.1. Method description of Binary Search

4.1.1. performBinarySearchAll:

The binary search perform method in array list for all search to DramaflixModels. Where we could search the array list within the DramaflixModels from low starting index and high ending index for the search where episode number could be searched.

4.1.2. performBinarySearchTitle:

The binary search performs a method in the array list to search titles for DramaflixModels. The title searchValue is within the array list DramaflixModels. From low starting index, high ending index, and filter to string with the list being matched to the title.

4.2. Method description of Merge Sort

4.2.1. DramaflixModels

In the merge sort entry point for the array list to the DramaflixModels is to be sorted. All the sorting attributes are specified based on the DramaflixModels where a is sorted by the ArrayList.

4.2.2. Merge

The merge has two array lists first and second in the mainModels based on the sorting specified attributes on DramaflixModels. The first and second ArrayList is sorted and the main ArrayList is merged through the result.

4.2.3. Sort

The array list od the DramaflixModels has been sorted and string sort is specifying the attributes based on the sorting performed. It is a sorted sublists of the merge in the sort method.

4.3. Method Description of Models

4.3.1. Get S_no and Set S_no

The method for the serial number attributes the provided access to DramaflixModels. The getter method is used for serial number retrieving of the number which allows retrieving and modify it. And setter method is used for setting the serial number drama.

4.3.2. Get Title and Set Title

The method for the title attributes the provided access to DramaflixModels. The getter method is used for title retrieving of the number which allows to retrieve and modify it. And setter method is used for setting the title drama.

4.3.3. Get Director and Set Director

The method for the Director attributes the provided access to DramaflixModels.The setter method is used for setting the director drama. The getter method is used for director retrieving of the number which allows to retrieve and modify it.

4.3.4. Get Screenwriter and Set Screenwriter

The method for the screenwriter attributes the provided access to DramaflixModels.The setter method is used for setting the Screenwriter drama. The getter method is used for screenwriter retrieving of the number which allows to retrieve and modify it.

4.3.5. Get Genre and Set Genre

The method for the Genre attributes the provided access to DramaflixModels.The setter method is used for setting the genre drama. The getter method is used for genre retrieving of the number which allows to retrieve and modify it.

4.3.6. Get Rating and Set Rating

The method for the Rating attributes the provided access to DramaflixModels.The setter method is used for setting the rating drama. The getter method is used for rating retrieving of the number which allows to retrieve and modify it.

4.3.7. Get Episode and Set Episode

The method for the Episode attributes the provided access to DramaflixModels.The setter method is used for setting the episode drama. The getter method is used for episode retrieving of the number which allows to retrieve and modify it.

4.3.8. Get Network and Set Network

The method for the network attributes the provided access to DramaflixModels.The setter method is used for setting the network drama. The getter method is used for network retrieving of the number which allows to retrieve and modify it.

4.3.9. Get Aired and Set Aired

The method for the aired attributes the provided access to DramaflixModels.The setter method is used for setting the aired drama. The getter method is used for aired retrieving of the number which allows to retrieve and modify it.

4.3.10. Get Country and Set Country

The method for the country attributes the provided access to DramaflixModels.The setter method is used for setting the country drama. The getter method is used for country retrieving of the number which allows to retrieve and modify it.

4.4. Method Description of Dramaflix

4.4.1. AddButtonActionPerformed

The add button method is executed when the user click the button and fill the textfield and update on data from the dramaflixList with new model DramaflixModels. It also clear the text filda and have a successful JOptionPane.

4.4.2. DeleteButtonActionPerformed

The delete button method is executed when the user click the button and select the row. It ask for the conformation to either delete or no and if we delete it will automatically update according to the dramaflixList.

4.4.3. UpdateButtonActionPerformed

The update button method is executed when the user click the button and select the table and do some changes in variable. It clears the text filed and also show the success JOptionPane message.

4.4.4. ClearButtonActionPerformed

The clear button method erase the writhing in the text field when the user click the button. It helps to clear all the textfield.

4.4.5. TableMouseClicked

The table mouse clicked method is used on the table of rows being used. It helps to select the row data from the table and retives the corresponding text.

4.4.6. SortButtonActionPerformed

The sort button method select the sort and update the table using the dramaflixList. It use the mergesortDramaflix sorter to sort the dramaflixlist.

4.4.7. SearchButtonActionPerformed

The search button method helps to search the code from the combo box and search button to search the certain variable. It is performed using binary search algorithm and display and prints the JOptionPane.

4.4.8. SnTextFieldKeyPressed

The serial textfield checks the event and helps invalid the wrong input by showing the error messege.

4.4.9. RatingTextFieldKeyPressed

The rating textfield checks the event and helps invalid the wrong input by showing the error messege.

4.4.10. EpisodeTextFieldKeyPressed

The episode textfield checks the event and helps invalid the wrong input by showing the error messege.

5. Test Case

Test no	1
Objectives	To add the value in an existing table.
Action	We should fill the text field of all the given text and click the add button
Expected Result	Clicking the button the table will be added and the dialogue box should be added.
Actual Result	It showed it had been updated in the dialogue box and added to the table too.
Conclusion	The test is successful.

Table 1: Add the value

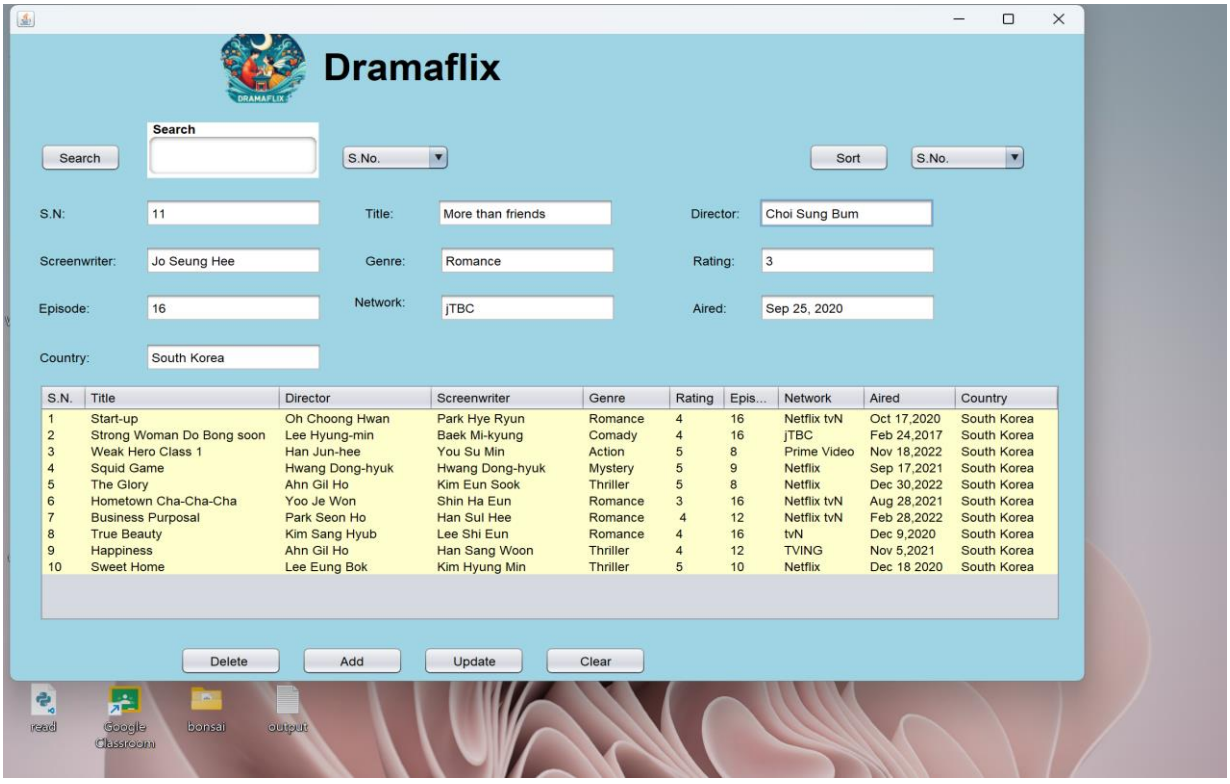


Figure 19 Running the program

Figure 20 Add success

Figure 21 Add shown in the table

Test no	2
Objectives	To update the value in an existing table.
Action	Select the random table text change some of its variables and click the update button.
Expected Result	Clicking the button the table will be updated and the dialogue box should be added.
Actual Result	It showed it had been updated in the dialogue box and updated to the table too.
Conclusion	The test is successful.

Table 2 Update Button

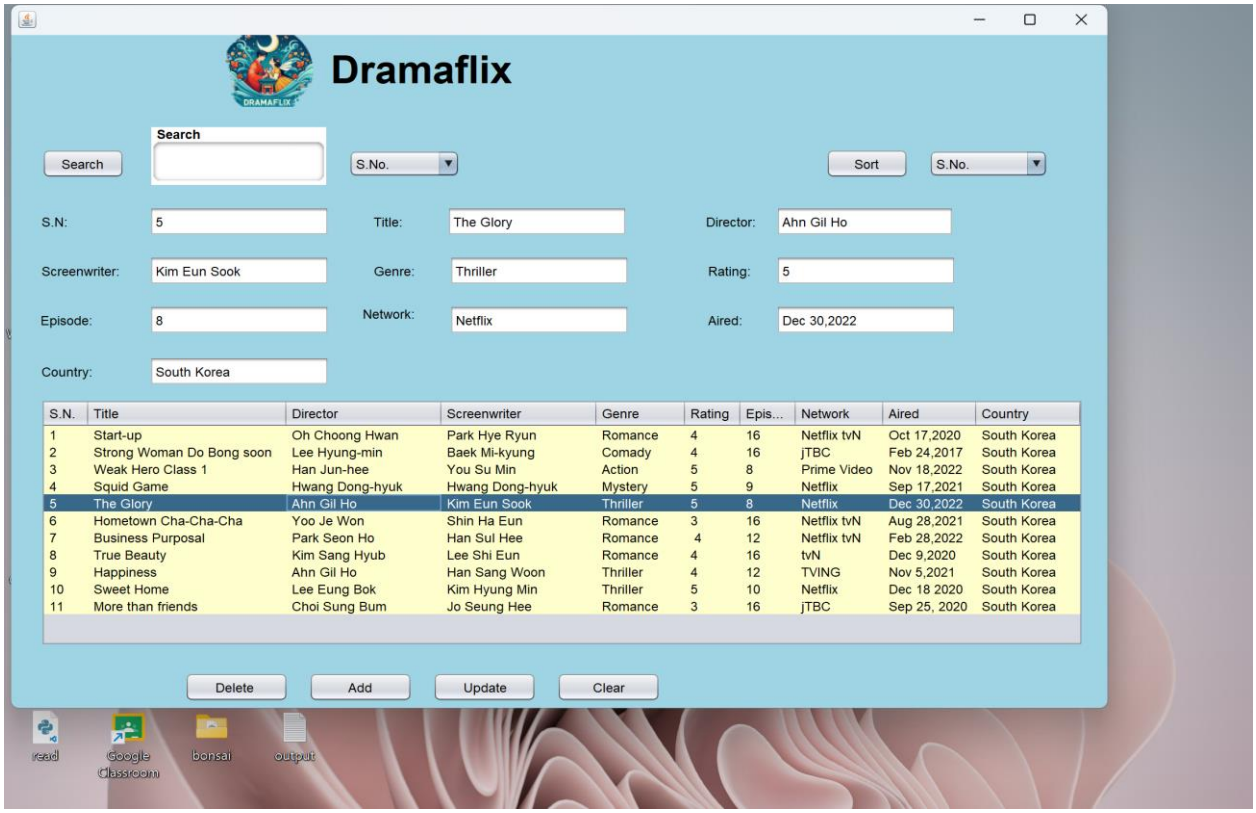


Figure 22 Selecting update

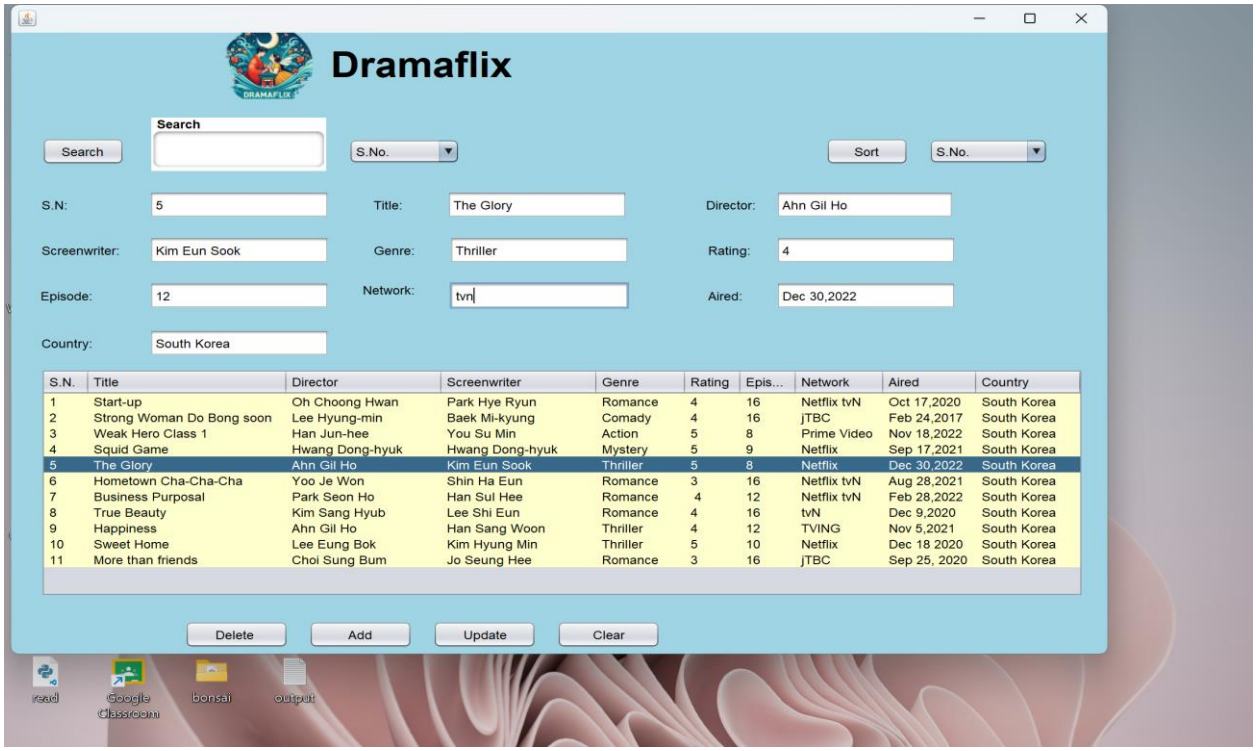


Figure 23 Editing the update

Figure 24 Sucess validation of update

Test no	3
Objectives	To delete the value in an existing table.
Action	Select the random table text and click the delete button.
Expected Result	By clicking the button delete the verification will come if we want to delete or not if yes then it will delete
Actual Result	It showed the verification to delete or not and it was deleted by clicking the button.
Conclusion	The test is successful

Table 3 Delete Button

Figure 25 Selecting to delete

Figure 26 Conforming the delete

Figure 27 Delete Completed

Test no	4
Objectives	For sorting the value in an existing table.
Action	Select the combo box choose the item that we want first and select the sort button.
Expected Result	By selecting a certain item from the combo box and clicking the sort button it will sort.
Actual Result	It showed the sort by the combo box and clicking the sort button.
Conclusion	The test is successful.

Table 4 Mergesort

Figure 28 Selecting from combobox for the sort

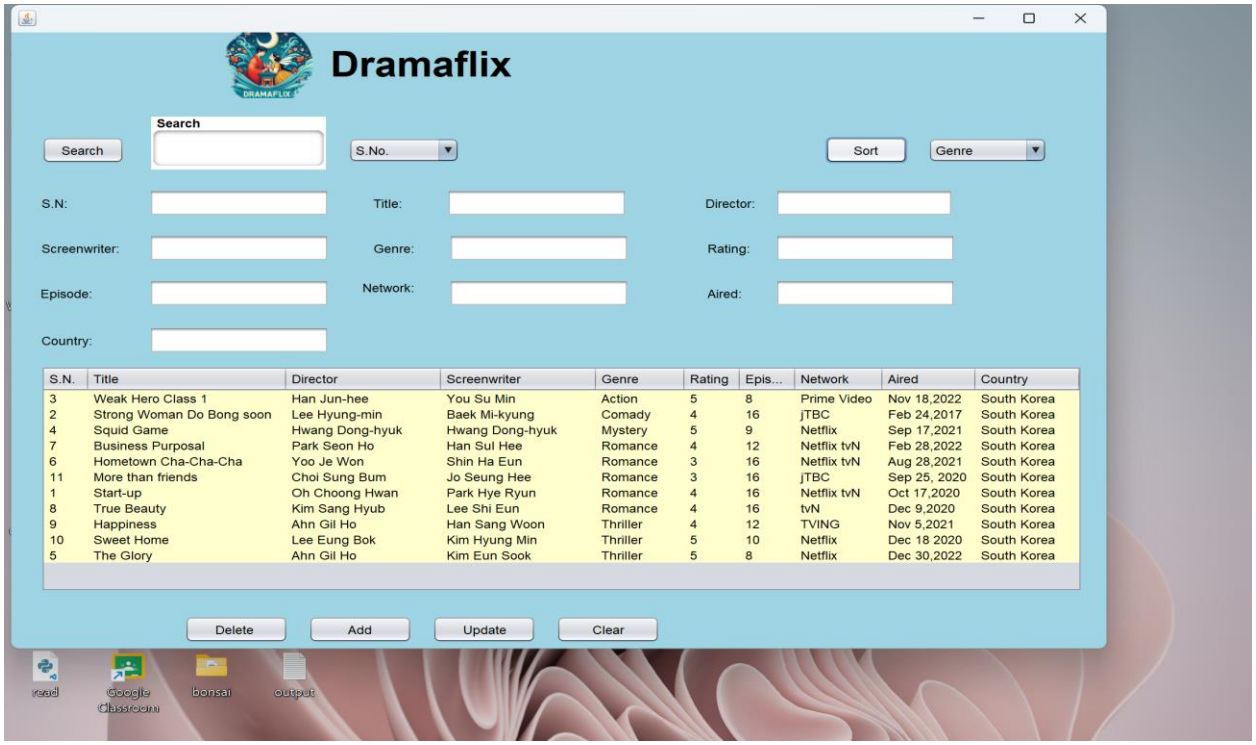


Figure 29 Sort of genre

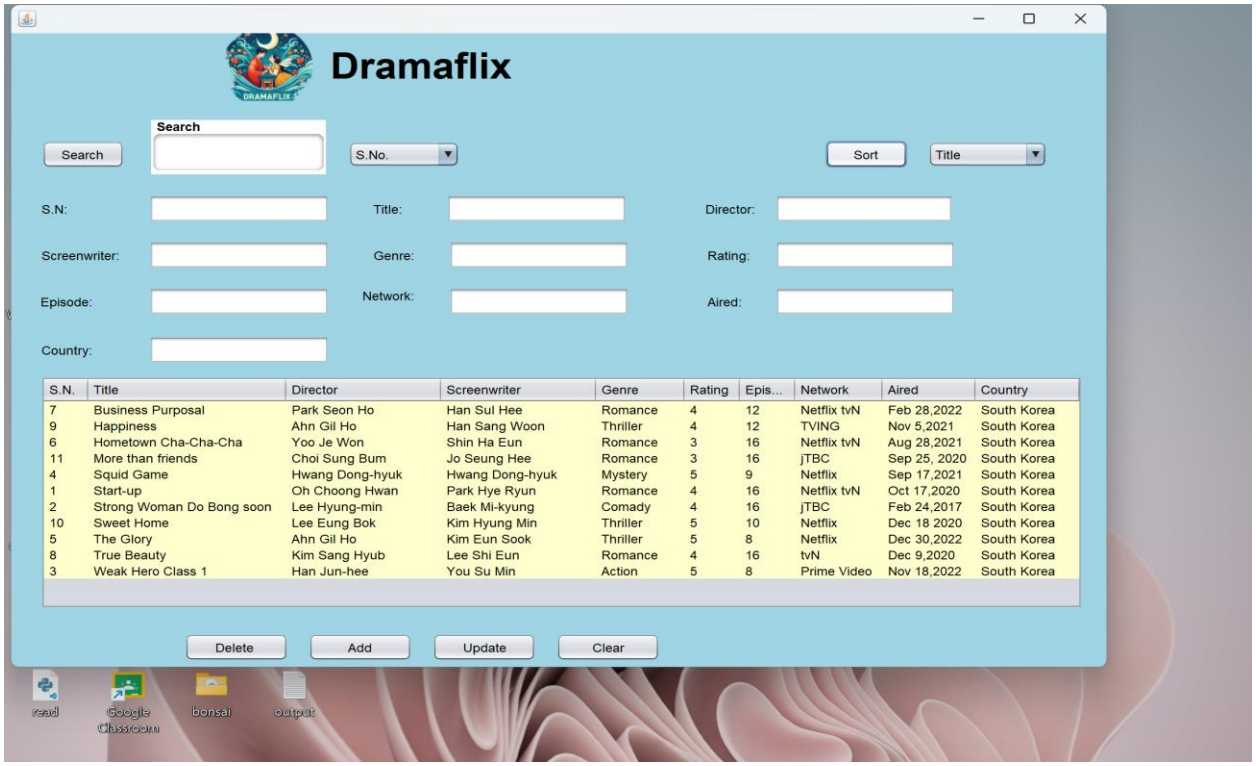


Figure 30 Sort of title

Figure 31 sort of S.No.

Test no	5
Objectives	For searching the binary search from the table.
Action	Select the combo box search the item and use the search button.
Expected Result	Select the combo box item search for the item related and use the search button.
Actual Result	It showed the item from selecting the combo box and searching it displayed.
Conclusion	The test is successful.

Table 5 Binary search

Figure 32 Selecting from combobox for Binary search

Figure 33 Display of Binary search of title

Figure 34 Display of Binary search of episode

Test no	6
Objectives	For the validation of a rating of more than 5.
Action	Select one of the table contents and put a rating of more than five.
Expected Result	Select the table of one content and put a rating of more than five In the dialogue box we could not put more than 5 come.
Actual Result	It showed by clicking the content and putting more than five the error message occurred.
Conclusion	The test is successful.

Table 6 Validation of rating

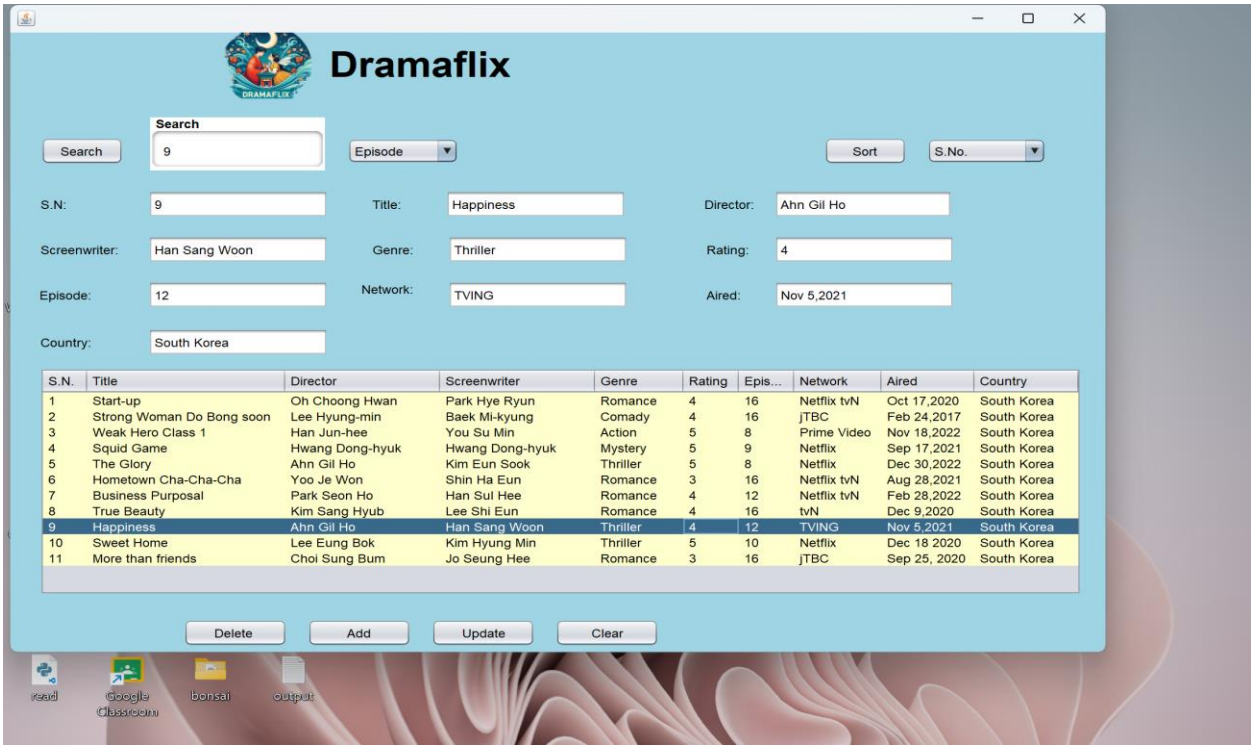


Figure 35 Selecting for the rating validation

Figure 36 Validation of rating

Test no	7
Objectives	For the validation of an int only write the number, not the alphabet.
Action	Select the int and write the alphabet error will occur.
Expected Result	Fill in the text field and write the alphabet in int type then the error message will occur.
Actual Result	It showed error message validation while filling in the int type using the alphabet.
Conclusion	The test is successful.

Table 7 Validation of int

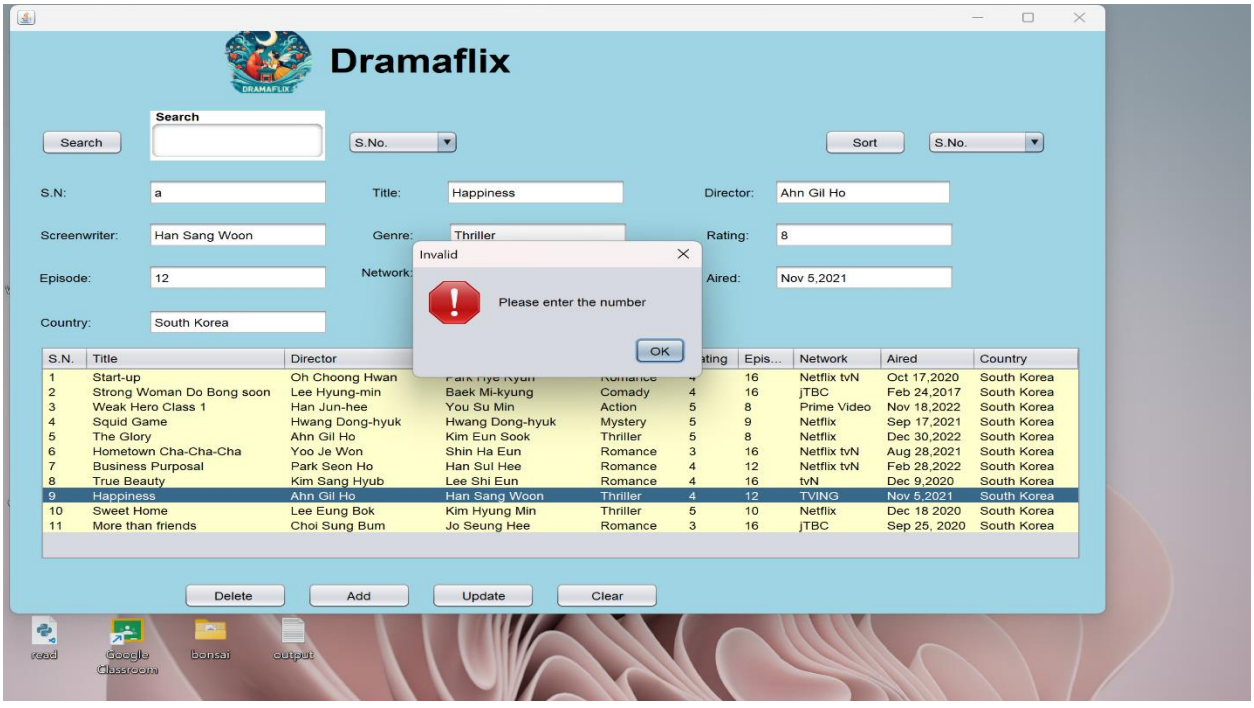


Figure 37 Validation from S.N.

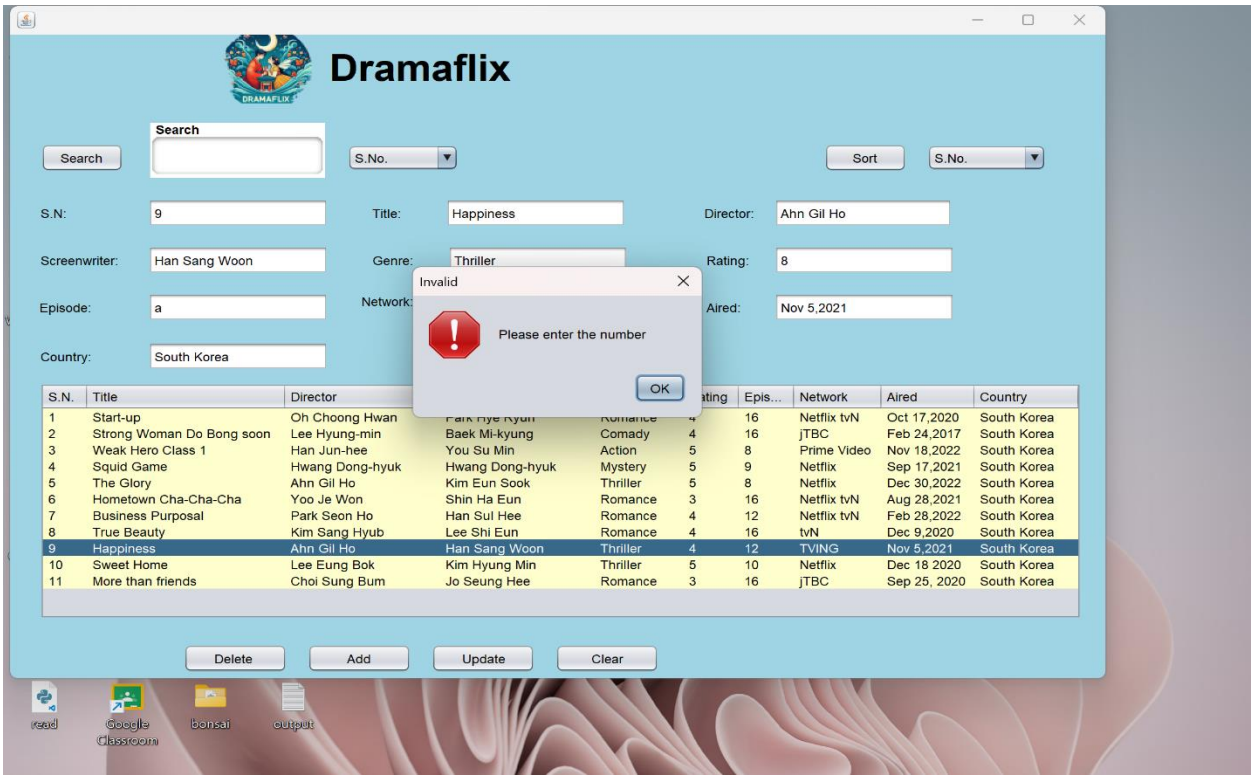


Figure 38 Validation from Episode

Figure 39 Validation from Rating

Test no	8
Objectives	For the update success validation
Action	Adding the update the validation will show Update success
Expected Result	After updating the row the changes will appear and the Validation will shown
Actual Result	By clicking the update button and changing the row validation will be shown
Conclusion	The test is successful.

Table 8 Validation of update

Figure 40 update validation

6. Development Process

6.1. Tools and Techniques Implemented

Draw.io:

A line graph diagram called Draw.io is useful for software development. It is an excellent free graph service for the user software. Making a lot of diagrams, such as flowcharts, wireframes, UML diagrams, network diagrams, and organizational diagrams, is helpful. You can export and save Draw.io in PNG, JPEF, PDF, and other formats. Because it makes it easier for users to construct web apps for the somewhat complicated charts and diagrams

Java:

Java is an object-oriented programming language that facilitates code creation, debugging, and compilation. James Gosling created Java in 1995 while working at Sun Microsystems. It supports references, threads, and interfaces and offers a large variety of classes with various inheritances. Java features built-in thread support and is an independent platform.

Apache Net Beans :

Apache Net Beans are software development application that address the help to the business, user and developer to develop their product quickly and effectively. It has strengthened the java platform and other platform for providing one of the best development product.

6.2. Challenges

In the process of completing the coursework, I have faced lots of challenges. First while doing the update button it did not automatically come while clicking the row after doing some research. I had to solve my first challenge it needed to be the first action performed on the table mouse clicked. The second challenge while doing the merge sort all my data from the table was gone from reviewing the code sir had provided we had not declared the array list. I had also a problem finding the color pattern that could fit the Dramaflix.

The final challenge faced was when BinarySearch was not converting into the string and had many displays with the help friend I could complete all the programs.

6.3. Problem Faced

Error no	1
Type of error	Syntax error
Error	Missing the parentheses of the getText of Titletextfiled
Resolving error	Adding the parentheses of the getText of Titletextfield.
Conclusion	Error Resolved

Table 9 Error no 1

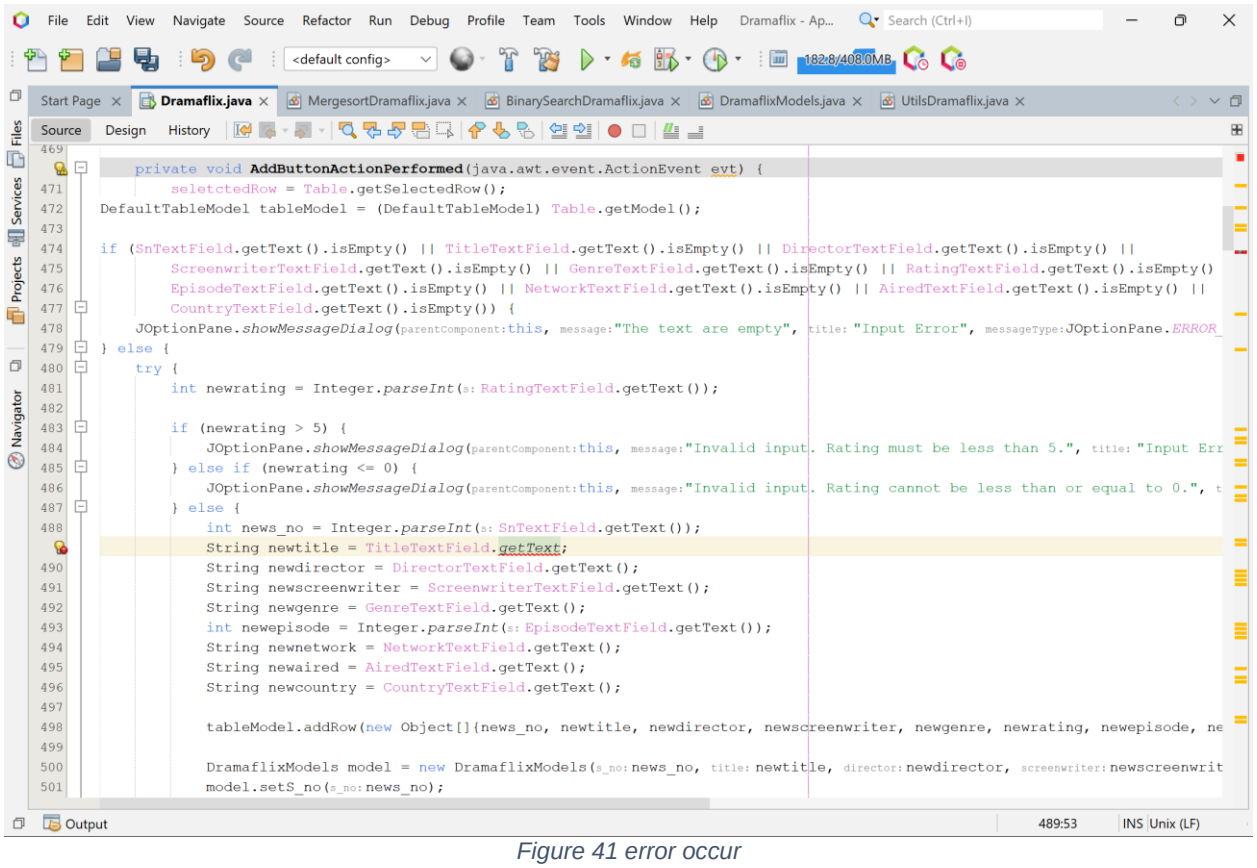


Figure 41 error occur

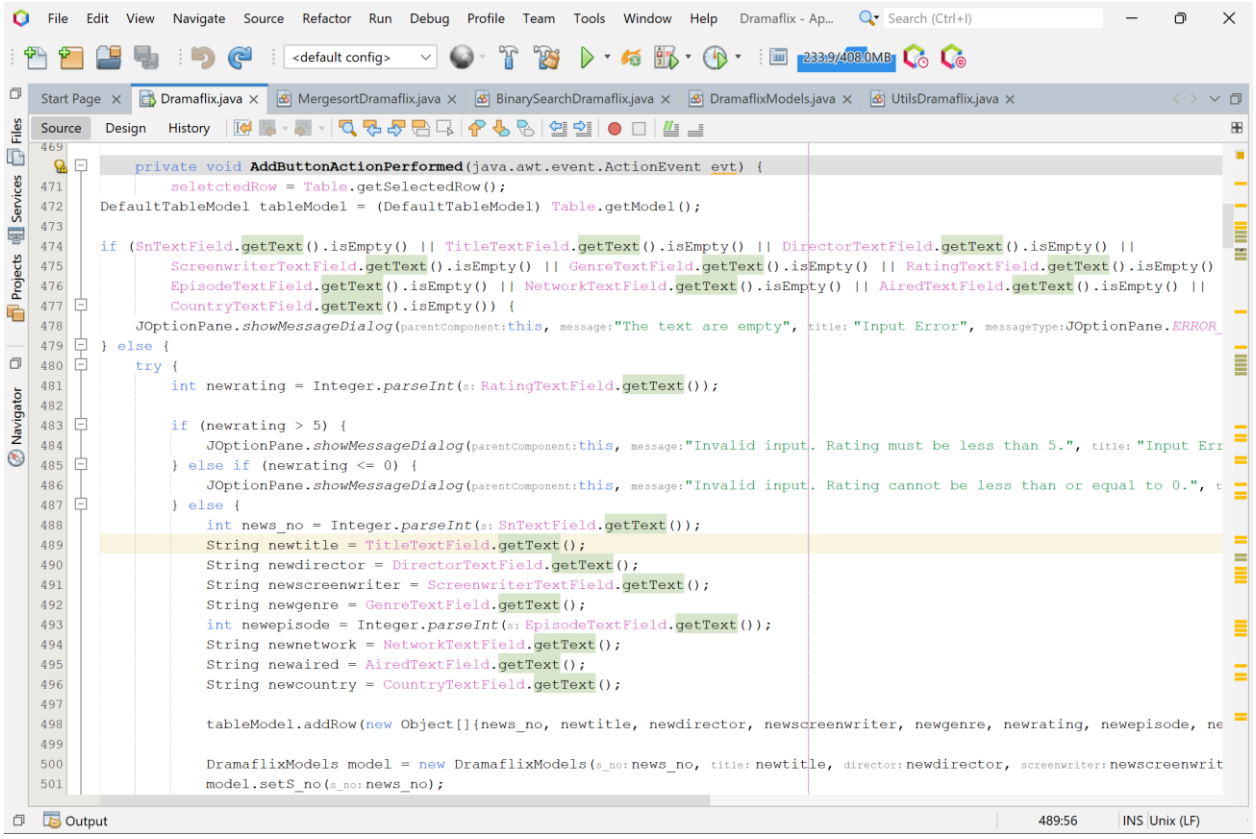


Figure 42 error solved

Error no	2
Type of error	Semantic error
Error	Parameter value is different variable than the declared variable. Where it should be Table but used table.
Resolving error	Declaring the parameter value same variable to the declared variable by changing the table into Table.
Conclusion	Error Resolved

Table 10 Error table 2

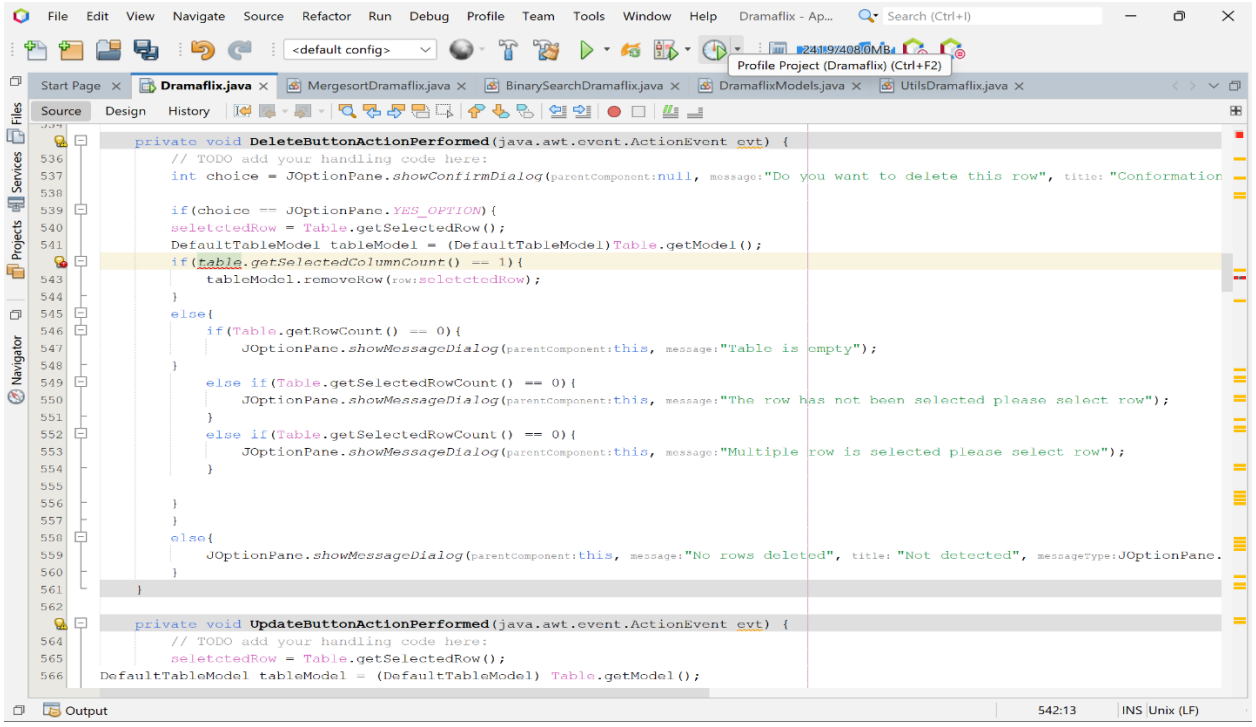


Figure 43 Error occur

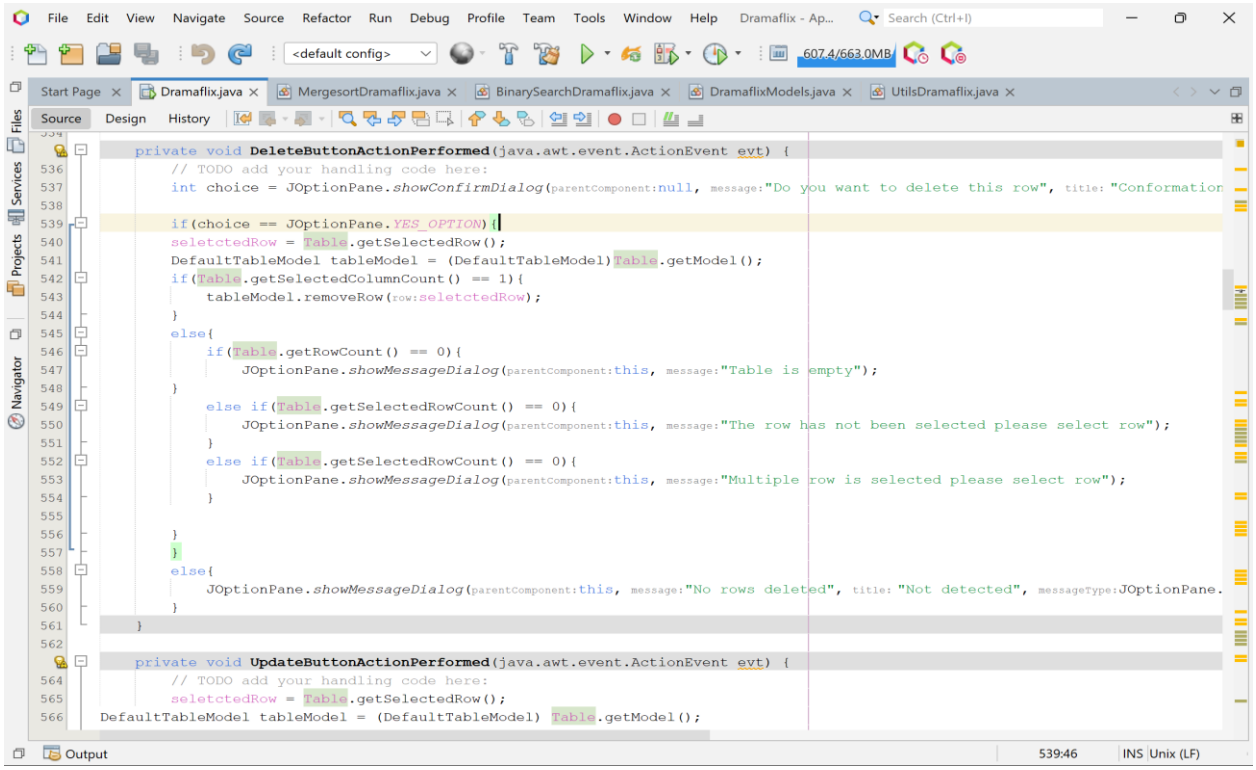


Figure 44 Error solved

Error no	3
Type of error	Logical error
Error	Parameter value is different variable than the declared variable. Where both variable were same.
Resolving error	Declaring the parameter value same variable to the declared variable changing one low into high.
Conclusion	Error Resolved

Table 11 Error table 3

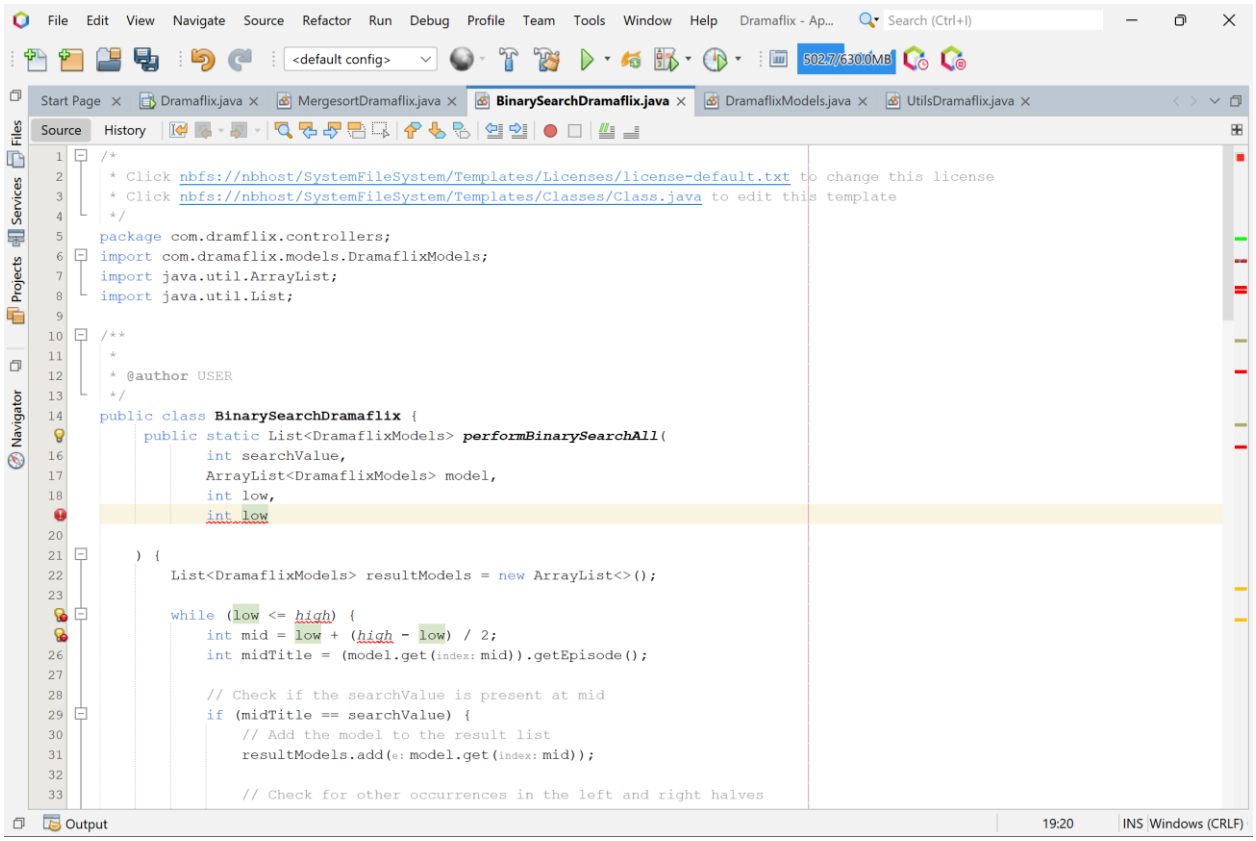


Figure 45 Error occur

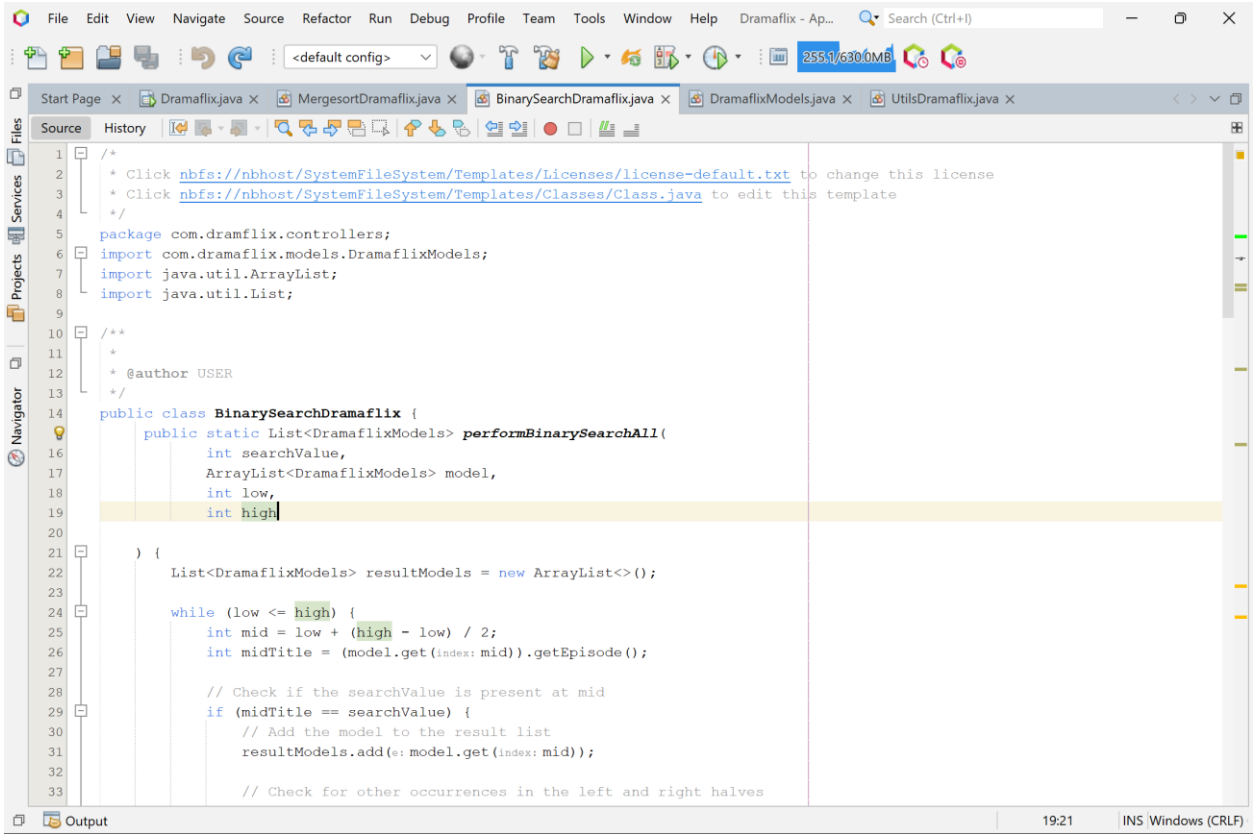


Figure 46 Error solved

6. Conclusion

From this course work I have gained an extensive knowledge of the Java program, which was centered on creating a software system “Dramaflix” which helps to store information about the drama. It has helped me to learn programming by encountering a lot of errors that we had to deal with. As, I learned more about Java, draw.io, and Apache NetBeans IDLE.

As, a part of the project, I learned how to use Apache NetBeans IDLe to create a GUI where we could update, delete, and add the information of the drama. We have also learned to use Binary Search Algorithms and sort. While creating the sort and Binary Search Algorithm I faced lots of problems, such as my sort array list could not work and the binary search algorithm kept on failing. But with the help of sir and friends, I could face the problem and complete my task. In addition, I was able to use the internet and deduct the problems I faced.

Finally, Learning how to make a software system was a fun, challenging, and exciting opportunity to learn more about the programming language.

7. References

References

Asana. (2023, 12 20). *What is a flowchart? Symbols and types explained*. Retrieved from Asana: https://asana.com/resources/what-is-a-flowchart

GeeksforGeeks. (2023, 12 19). *Merge Sort – Data Structure and Algorithms Tutorials*. Retrieved from GeeksforGeeks: https://www.geeksforgeeks.org/merge-sort/

GeeksofGeeks. (2023, 12 19). *Binary Search – Data Structure and Algorithm Tutorials*. Retrieved from GeeksofGeeks: https://www.geeksforgeeks.org/binary-search/

Gillis, A. S. (2023, 12 19). *What is an algorithm?* Retrieved from TechTangent: https://www.techtarget.com/whatis/definition/algorithm#:~:text=An%20algorithm%20is%20a%20procedure,%2D%20or%20software%2Dbased%20routines.

Simplilearn. (2023, 12 27). *An Introduction to Methods in Java with Examples*. Retrieved from Simplilearn: https://www.simplilearn.com/tutorials/java-tutorial/methods-in-java

Tutorialpoints. (2023, 12 22). *UML - Class Diagram*. Retrieved from Tutorialpoint: https://www.tutorialspoint.com/uml/uml_class_diagram.htm

8. Bibliography

Bibliography

Asana. (2023, 12 20). *What is a flowchart? Symbols and types explained*. Retrieved from Asana: https://asana.com/resources/what-is-a-flowchart

GeeksforGeeks. (2023, 12 19). *Merge Sort – Data Structure and Algorithms Tutorials*. Retrieved from GeeksforGeeks: https://www.geeksforgeeks.org/merge-sort/

GeeksofGeeks. (2023, 12 19). *Binary Search – Data Structure and Algorithm Tutorials*. Retrieved from GeeksofGeeks: https://www.geeksforgeeks.org/binary-search/

Gillis, A. S. (2023, 12 19). *What is an algorithm?* Retrieved from TechTangent: https://www.techtarget.com/whatis/definition/algorithm#:~:text=An%20algorithm%20is%20a%20procedure,%2D%20or%20software%2Dbased%20routines.

Simplilearn. (2023, 12 27). *An Introduction to Methods in Java with Examples*. Retrieved from Simplilearn: https://www.simplilearn.com/tutorials/java-tutorial/methods-in-java

Tutorialpoints. (2023, 12 22). *UML - Class Diagram*. Retrieved from Tutorialpoint: https://www.tutorialspoint.com/uml/uml_class_diagram.htm