# *Programming II (420-G20-HR)*

# *Assignment 3 – Searching and Sorting*

Date assigned: Fri. April 16, 2021

Test Plan, Class Diagram due: **Fri. April 23, 2021**

Completed assignment due: **Mon. May 10, 2021**

**Learning Objectives**

Upon successful completion of this assignment, the student will be able to:

1. Design test cases from requirements
2. Design a frame from requirements.
3. Declare and instantiate an ArrayList of objects in Java.
4. Add objects to an ArrayList in Java.
5. Write the code to find an element in an ArrayList using a binary search in Java.
6. Sort an ArrayList using a bubble sort.
7. Create a single level control break report.
8. Format output reports in a Java program.
9. Read a data file from a Java program.
10. Extract delimited fields from a string in Java using a **StringTokenizer**.
11. Use various Java **String** methods.
12. Write to a data file from a Java program.

**To be handed in:**

1. A test plan and class diagram must be handed in for review on Fri. April 23. The test plan should be a numbered list of test scenarios that need to be tested. The files, the frame, and the report logic all need to be considered in the test plan.
2. The final Java project called ***username\_*G20\_A03\_MusicSystem** should be zipped and uploaded to **Moodle** on Mon. May 10, 2021 andshould include the following in addition to the Java **src** folder:
   1. ***username\_*G20\_A03\_Test\_Plans.docx** containing the updated test plans for the system.
   2. ***username\_*G20\_A03\_ClassDiagram** containing the updated class diagram for the system.
   3. A self-assessment for the assignment. (Use the template from **Moodle**.)

**Programming Style:**

Marks will be deducted for poor style. The following criteria will be used:

* All Java source files must be formatted using the Eclipse formatter. (Right-click and select **Source🡪 Format**.),
* Your code must meet all the requirements of the Heritage College Computer Science Code Checklist.

**Organization:**

5 marks of the assignment mark will be for organization. The self-assessment must be completed and included in the assignment folder and the assignment must be:

* + handed in to the correct location,
  + be properly named
  + be complete according to the assignment specifications

***Marking Scheme:***

|  |  |
| --- | --- |
|  | **Out of** |
| Test Case Plan – includes tests for testing the frame (input validation and report output), the files, the search and sort functionality. | 15 |
| Class Diagram – all classes and methods included; correct use of access. | 10 |
| Song java code – has constructors, accessors and mutators; correct access on instance variables. | 15 |
| MusicSystem java code – code divided correctly between classes; constructors, accessors, mutators; ArrayList used to store songs from file; reading in from file works; writing to file in sorted order works; binary search used and working; bubble sort used;. | 55 |
| MusicSystemFrame java code - all frame components work; search validation errors report; search works as expected; reports displayed in correct format with correct data; total downloads displayed on report; quit functionality; menu’s work as expected; scroll bars used in report output. | 55 |
| Correct execution against requirements; thoroughly tested against test cases. | 15 |
| Documentation submitted on April 23, 2021 | 10 |
| Organization | 5 |
| **Total** | **180** |

# Music Reporting System Problem Specification

1. You are going to design a music reporting system that lists the top songs of the year. You will read in song information from a file named **music.txt** and store it in an **ArrayList**. You will then search through the **ArrayList** using binary search, sort the **ArrayList** using bubble sort and display reports based on this information. You will also create a new file called sortedMusic.txt. **You must use ArrayLists. You must write your own binary search and use it for searching. You must write your own bubble sort and use it for sorting.**
2. Your system must have three classes called **MusicSystemFrame**, **Song** and **MusicSystem**. The **MusicSystemFrame** class deals exclusively with displaying the input and output through a frame interface. The **MusicSystem** class handles the file processing, the searching and the sorting. The **Song** class stores individual songs.
3. Your **MusicSystem** class should have one ArrayList called **songArray**. The **Song** objects are read in from the **music.txt** file and stored in the **songArray**. The **music.txt** file will not contain the songs in the correct order, so the **MusicSystem** class must then sort the **songArray** using your own bubble sort method.
4. You can download the "**music.txt**" file from **Moodle**. The format of the **music.txt** data file is:

|  |  |  |  |
| --- | --- | --- | --- |
| **Position** | Contents | **Datatype** | **Delimiter** |
| 1 | year | Int | \* |
| 2 | ranking\* | Int | \* |
| 3 | artist | String | \* |
| 4 | songTitle | String | \* |
| 5 | number of digital downloads (in thousands) | int | \n |

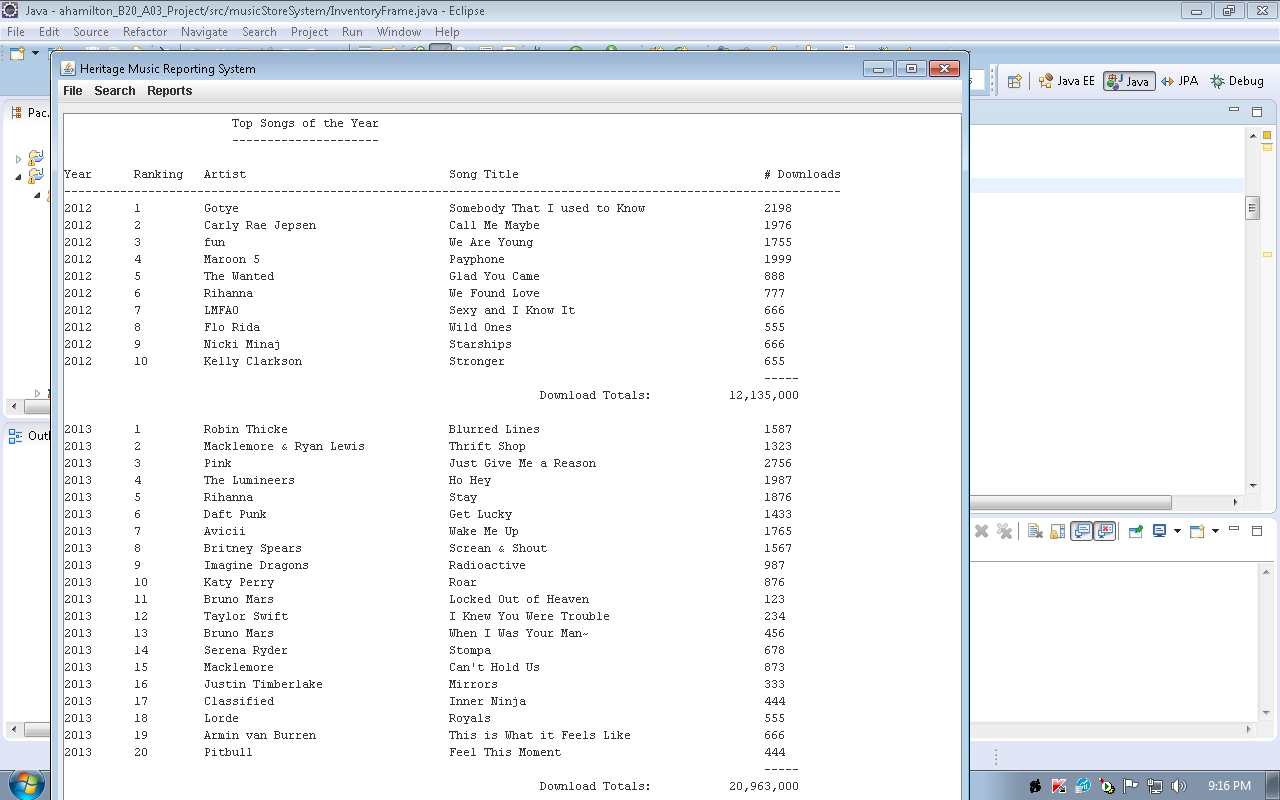
\*The ranking represents the spot on Billboard’s top songs of the year

1. The format of the **sortedMusic.txt** data file that you will create is the same as the **music.txt** file, but sorted by year and ranking.

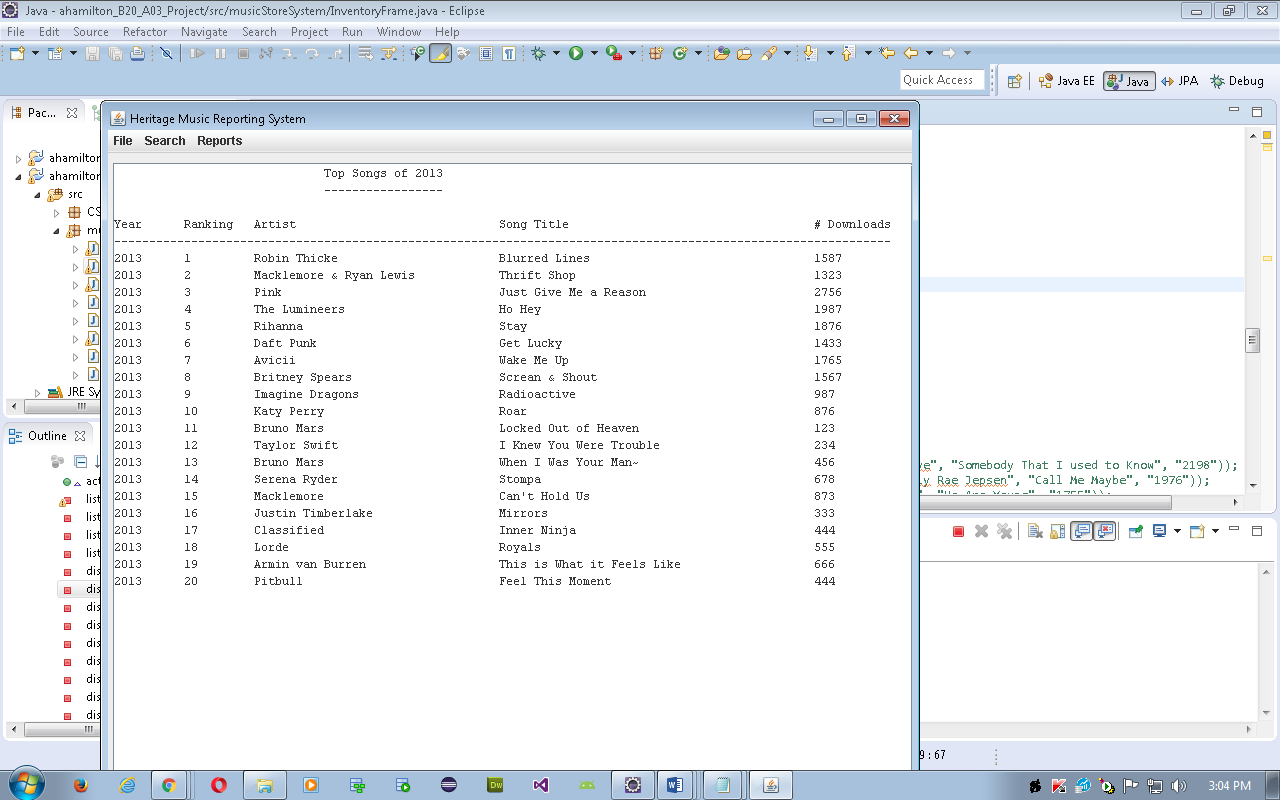
1. A JFrame must be created, that is menu driven. The menu functionality is as follows and sample output is located at the end of the assignment:

|  |  |  |
| --- | --- | --- |
| **Menu** | **MenuItem** | **Action** |
| File | Write | Write the music entries to a file called sortedMusic.txt, sorted by year and ranking. |
|  | Exit | System.exit(0); |
| Search | By Song Title | Find music entry whose song title is entered in JOptionPane |
|  | By Year/Ranking | Find music entry whose year and ranking is entered in JOptionPane |
| Reports | List All Music | Display report that lists all music, ordered by year and ranking. At the end of each year, the total digital downloads for that year should be reported. |
|  | List by Year | Display report that lists music for a specific year, ordered by ranking. |

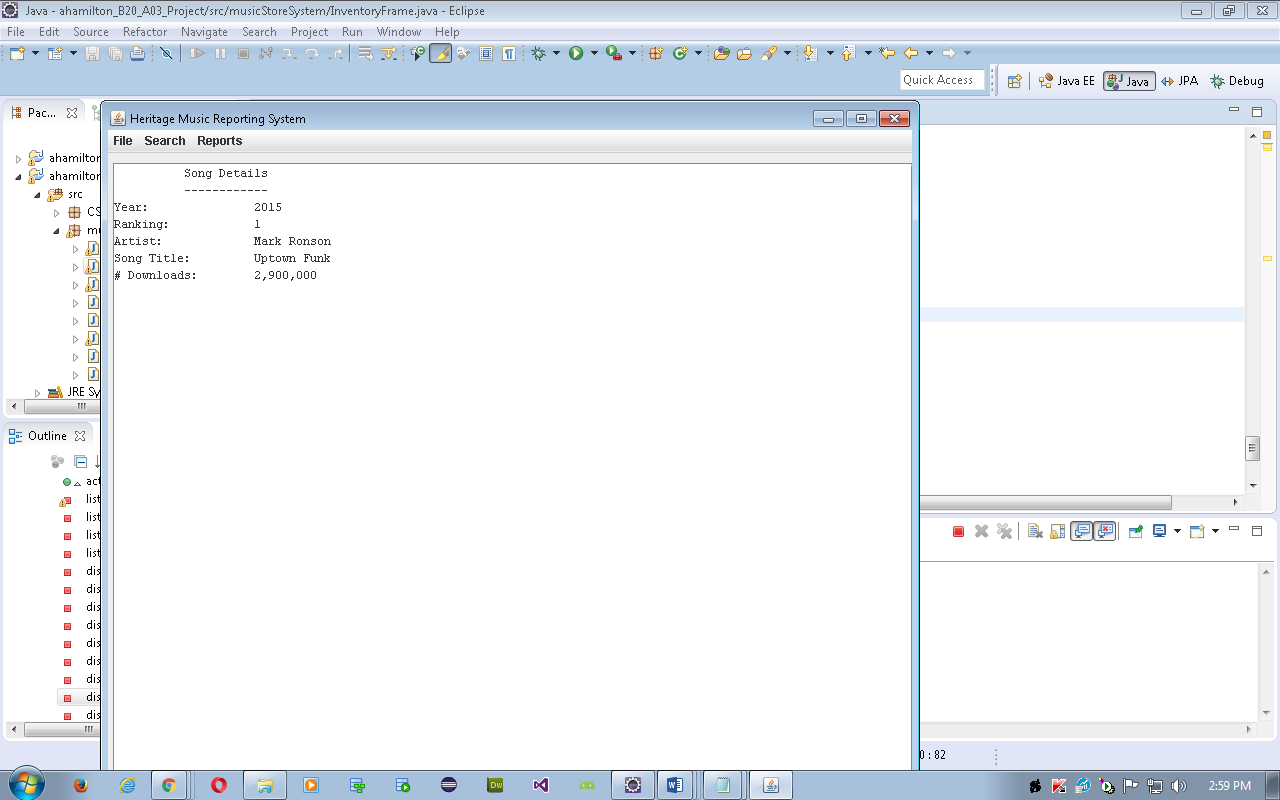
1. At least one of the searches must be performed using a **binary search** method that you write.
2. The text area to display the reports must have scroll bars.
3. If the song title is not found by the search, then a pop-up error message must be displayed stating that “The song <songName> was not found.”
4. If the year and song ranking are not found by the search, then a pop-up error message must be displayed stating that “The year <year> and ranking <ranking> were not found.”
5. If the year entered is not a valid year or the song ranking is not an integer, then a pop-up error message must be displayed.
6. When the Reports->List All Music menu option is chosen, the display should be similar to the following. You should have a scroll bar that allows you to scroll through the whole list of all years.



1. When the Reports->List by Year menu option is chosen, the display should be similar to the following. You should have a scroll bar that allows you to scroll through the whole year.



1. When the Search->By Song Title menu option or the Search->By Year/Ranking menu option is chosen, the display should be similar to:



### Your tasks:

1. Create test plans for testing the **Song** class, the **MusicSystem** class and the **MusicSystemFrame** class. This should be a numbered list of the scenarios to be tested. The scenarios should describe in a sentence what to test and the expected result.
2. Design the system. Draw a class diagram that includes all classes involved, including your frame class. .
3. Get approval of your frame design and test plans before you start coding.
4. Code the classes according to the specifications. Test as you go.
5. Completely test your program using the test plans you developed in the first step.
6. Document your code with comments; format your code.
7. Complete the self-assessment.