Program 2 – Shape Database 2.0

The purpose of this assignment is to practice utilizing the various concepts learned in Chapters 10, 11, 12 (from CPS141) and 13 of our text. Specifically, this assignment will require use of the following concepts.

- Functional Interfaces (10.11)
- Exception Handling (11.1)
- Label Control (CPS141 - 12.2)
- Scene, VBox, HBox Controls (CPS141 - 12.3, 12.5)
- Event Handlers (CPS141 - 12.8)
- JavaFX and CSS (13.1)
- ListView Controls (13.4)
- ComboBox Controls (13.5)
- Menus (13.8)

The program will implement a simple graphical user interface for the Shape database from Program 1 and will allow the user to display Shape information for all valid shape records in the database and records for specific shape types.

1 User Interface Specifications

This section corresponds to Item 3 of the "Program Rubric" and specifies how the user should interact with the program. The user interface will be implemented using JavaFX and will consist of a single page with two different "modes" or "skins" as shown in the figures below. Part of the project will be to stylize certain aspects based on the developer's preference to support a "light" and "dark" mode for the interface. The shape information will be loaded from a CSV (comma, separate, variable) file that will be provided.

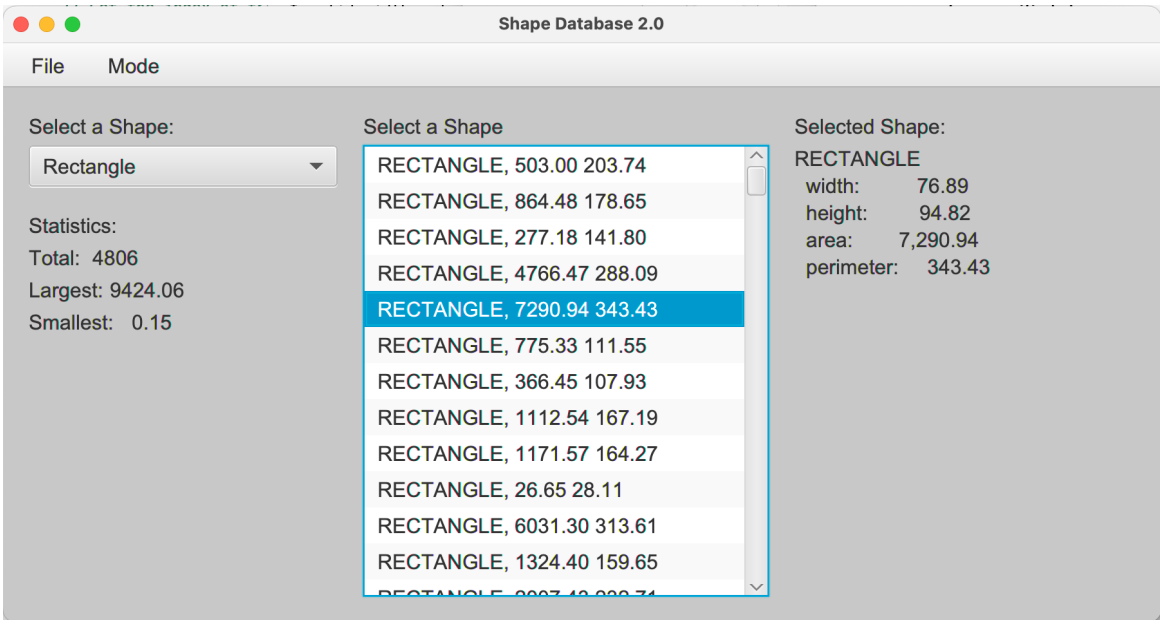


Figure 1, Shape Database 2.0 - Light Mode

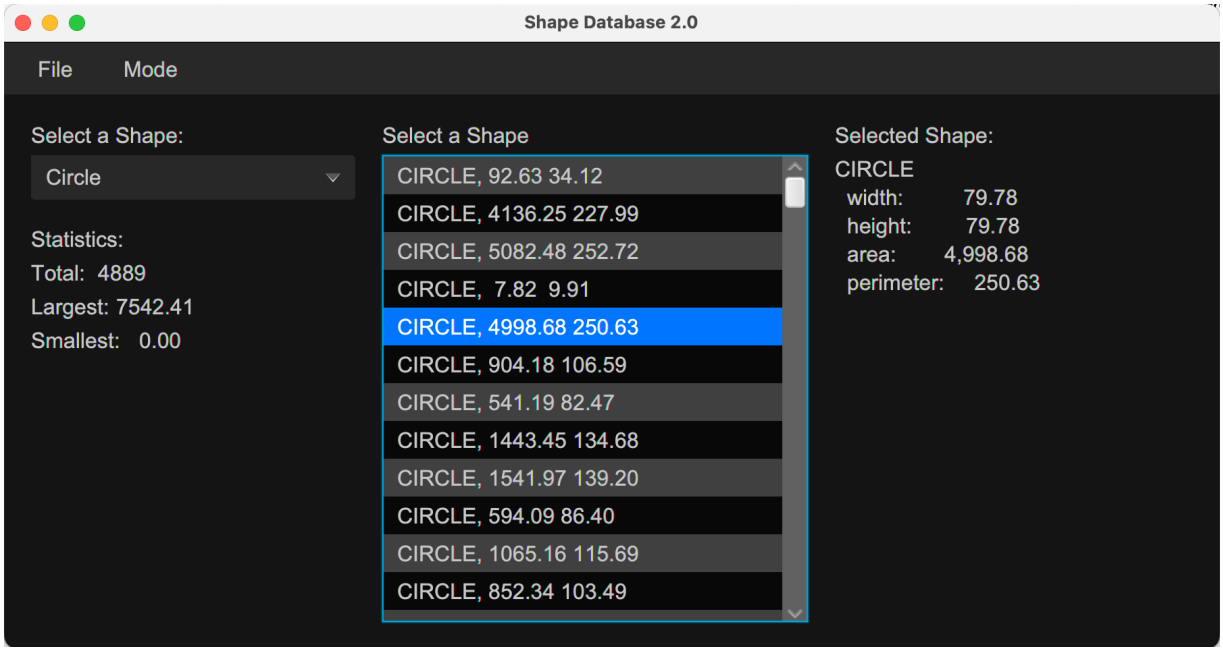


Figure 2, Shape Database 2.0 - Dark Mode

1.1 Main Page

The Main Page, for both Light and Dark Modes, should consist of the JavaFX Controls shown in the figures above. User interaction will be implemented using some of the JavaFX controls from Chapters 12 (CPS141) and 13. Controls will be grouped using VBox, HBox, and or GridPane layout containers into three columns and a menu bar as follows:

- Left-Side Control Column - Filter Controls
  - a ComboBox to select shape types including, All, Circle, and Rectangle.
  - a group of Label controls in which to display basic statistics of the filtered set of shapes including the total number, the value of the largest shape area, and the value of the smallest shape area, based on the selected Shape type.

- Center Control Column - Filtered Set Display and Selector
  - a ListView control in which to display the filtered set of shapes and to allow selection of a particular shape

- Right-Side Control Column - Selected Shape Display
  - a group of Label controls to display information about the shape selected in the ListView Control.

- Top MenuBar with two Menus including a
  - File Menu, with a single MenuItem to exit the app,
  - Mode Menu to with RadioMenuItem objects to select the light or dark mode.

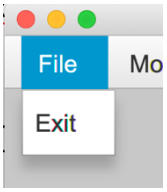
1.1.1 Program Start

The following conditions should apply when the program starts:

- The Shape Selector should be set to “All”.
- Total: should be the total number of valid shape records found in the database.
- Largest should show the value of the area of the shape with the largest area in the current selection.
- Smallest should show the value of the area of the shape with the smallest area in the current selection.
- The Select Shape ListView should contain a list of all shapes in the database.
- The Selected Shape information should be empty.
- The program may start in either Light or Dark mode depending on your preference. The associated radio menu item should be checked under the Mode menu (see below).

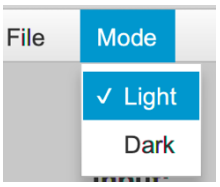
1.1.2 File Exit Menu

There will be a File Menu with a single Exit MenuItem that will allow the user to exit the program.



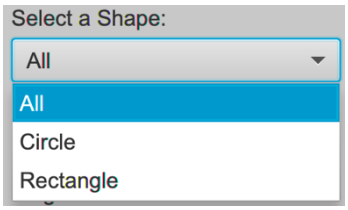
1.1.3 Dark Mode Operation

A Mode Menu will provide two RadioMenuItem objects for Light and Dark mode options. After initial startup, when the user clicks the Dark Mode RadioMenuItem, the page should immediately switch to the dark visual mode. When the Dark Mode RadioMenuItem is checked, the Dark Mode style should be used. When the Dark Mode RadioMenuItem is unchecked then the Light Mode style should be used.



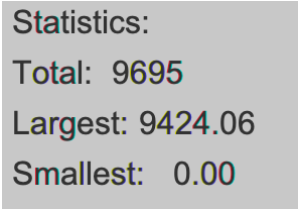
1.1.4 ComboBox Control

A ComboBox control with options for All, Circle, and Rectangle should be included and will be used to filter the Shape information displayed in the ListView to a single shape type.

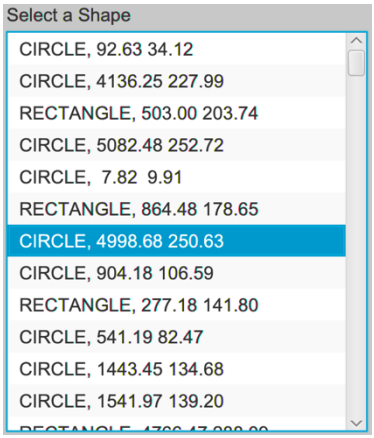


1.1.5 Filtered Shape Statistics Display

JavaFX Label controls will be used to display some simple statistics about the filtered set of shapes including the total number in the filtered set, the distance of the smallest shape in the filtered set and the radius (Jupiter radius or Jr) of the largest shape in the filtered set.

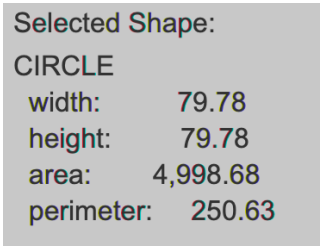


1.1.6 Select a Shape Display



A JavaFX ListView control will be used to display the complete list of shape records in the filtered set based on the user's shape type selection. Each row in the ListView is a String that contains the shape type (i.e., Circle or Rectangle) and the area and perimeter of the Shape. When the user selects a row, detailed information for the selected shape will be displayed in the right-hand control panel. If the user changes the Selected Shape type, then the ListView selection and selected shape information will be cleared.

1.1.7 Selected Shape Display



A set of JavaFX Label controls will be used to display specific information about the selected shape record. The information must include the Shape type, width, height, area, and perimeter similar to that shown in the screenshot. You are free to style differently as long as all the information is present.

1.2 Styling

You are free to style the program as you wish using two CSS files that will be required as part of the deliverable including one CSS file for the Light Mode and a separate CSS file for the Dark Mode. Feel free to change the background colors, text colors, fonts, or any other available style as you see fit. Here is a link to the complete JavaFX style reference in case you'd like to use it.

https://docs.oracle.com/javafx/2/api/javafx/scene/doc-files/cssref.html

1.3 Layout

You may optionally rearrange the layout of controls as along as all controls and labels shown in the figures above are present and the design is user friendly.

2 Java Implementation

This section corresponds to Item 4 of the "Program Rubric" and describes the details of how the program should be implemented to achieve the User Interface experience of Section 1.

You are free to implement the user interface described in Section 1 in any way you choose as long as the following requirements are met:

1. The program must be written using only the controls and classes we have learned in class. Do not utilize any libraries or classes we have not covered in our course.
2. The program must be written using the specific JavaFX Controls given in Section 1.
3. The program must implement all user-interface features described in Section 1.
4. The program must be user friendly and functionally correct.
5. The program must implement and utilize any specific classes or interfaces described in the following sections as part of the overall implementation.

6. A "throws" clause must not be written in main method header. All exceptions related to reading data from the database file should be handled internally by the program.

2.1 Program02 - Main Class File

A working partially completed main program template will be provided for your reference. You may use it as the starting point of your implementation, or you may use it as a model to follow for you own independent implementation. The template will contain numerous TODO comments to describe the tasks needed to complete the working program based on the template code. If you complete each TODO item, you will be able to produce a working program.

2.2 Program01 - Classes

Reuse all of the classes and interfaces you created in Program 01 for this project. Make any corrections and updates that are needed to get it to work with this program. Refer to the Program01 specifications for details on the specification for these classes, interfaces, and enumerations.

2.3 CSS Style Files

Write two CSS styling files, one for light mode and one for dark mode, based on the CSS styling specifications covered in the book. The book does not come close to covering all the options, but you are welcome to get more information from the link provided below.

https://docs.oracle.com/javafx/2/api/javafx/scene/doc-files/cssref.html

Two files should be included:

1. light.css - Contains CSS styling rules for the "light mode"
2. dark.css - Contains CSS styling rules for the "dark mode"

GRADING NOTE: There will be a lot of latitude given in the quality of your styling with respect to the grade. What I'm mainly looking for is evidence of a real effort to style the page. Your page may not look as good as my example, or you may be able to do something much better. Either way is fine so long as I see evidence that you made a sincere effort.

3 Deliverables

The deliverables for this project will consist of the following .java and .css files.

1. Program02.java – or whatever you name your main program.
2. light.css – A JavaFX CSS file that contains style classes for the Light Mode style.
3. dark.css - A JavaFX CSS file that contains style classes for the Dark Mode style.
4. Include the files from Program 1 with any updates or corrections you may have written.
5. Optionally, if you do not include the files from Program 1, I will reuse the ones you previously submitted. **Do not use this option if updates or corrections were needed.**

DO NOT SEND THE .class FILES. I CANNOT USE THEM.

The following Java interfaces and classes will be provided with this assignment to help you create your own implementation. Please do not alter these files in any way, except for Program02StarterTemplate.java which you may use as the starting point of our project.

1. UpdateHandler.java
   A functional interface with a method that accepts no arguments. This will be useful for handling updates to the filter selection.

2. BoundableSelector.java
   A functional interface to select between two Boundable objects based on some criteria. Will be useful for finding the largest and smallest shapes.

3. BoundableReducer.java
   A functional interface with a method accepting an ArrayList of Boundable instances and a BoundableSelector instance. Will be useful in finding the largest and smallest shapes.

4. BoundablePredicate.java
   A functional interface that helps determine if an Boundable meets some programmed condition. Will be useful for filtering the ArrayList of shapes based on a user-selected telescope.

5. BoundableFilter.java
   A functional interface with a method accepting an ArrayList of Boundable instances and a BoundablePredicate instance. Will be useful in filtering the ArrayList of Boundable objects for a specified telescope.

6. Program02StarterTemplate.java
   A partially implemented, working, template program that should be used as a starting point for your implementation. You should rename this file to something more appropriate for your Program 3 submission.

The Program02StarterTemplate.java file is provided to give you a starting point for the JavaFX program. You should do your work in a copy of this file with a more appropriate name.

You will also be provided with the "shapes_20230701.csv" file to use as the Shape database for the program.

The due date/time is:
Saturday, July 29, 2023, at 11:59PM

4 Sample Output

See the screenshots on pages 1 and 2 of this assignment for some examples of how the program user interface could look.

5 General Submission Policy

You may submit your code multiple times up until the due date/time. Any submission after the due date/time will be subject to the late assignment policy described in the course Syllabus. In all cases the last code you submit will be used for the grade.

6 General Requirements

All code you write for this course should meet the following general requirements:

1. All code should be your own work and not copied from another student, the internet, or any solution guide for the textbook.

2. Code from the following sources may be used or modified to complete the project:
    a. Any code provided by me in the class notes.
    b. Any code provided by the textbook itself (excluding any solution guides).
    c. Any code written by you for this or another course.

3. All Java classes should have a documentation comment including, author, date, course (i.e., CPS142-Summer 2023), and a purpose or brief description statement.

4. All methods must have properly formatted and descriptive documentation comments. The only exception is methods that override well documented interface or superclass methods. Private methods must also be commented.

5. Any method with more than just a few lines of code should have additional single-line comments describing important parts of the algorithm.

6. The use of break, continue, or exit statements inside loops is not acceptable in the implementation of any class. If found points will be subtracted from Item 5 of the Programming Rubric.

NOTE: The above requirements should be followed only where applicable.

7 Program Rubric

Your grade for this assignment will be computed using the following rubric.

Item	Description	Percent of Grade
1	Compilation: Compiles with no syntax errors.	10
2	Execution: Runs with no crashes or runtime errors.	10
3	External Operation: Written to meet all requirements for input and output as specified or implied by the problem.	30
4	Internal Operation: Written to meet all requirements for internal operation as specified as specified or implied by the problem statement.	30
5	Coding Style: Written and formatted in accordance with best practices presented in class (indentation, class names, variable names, method names, etc.).	5
6	Project Comment: Contains a **documentation comment** with the assignment name, author, date, class (i.e., CPS142-Summer 2023), and purpose/description at the top of each file.	5
7	Documentation: All methods, classes, and interfaces have **documentation comments**. Methods that override well-documented interface method and the main method are exempt.	5
8	Comments: Includes sufficient other regular comments to describe important parts of the code.	5
	Total	**100**