TEAM C

JAVA ASSIGNMENT -3

Team Members:

Harshit Oberoi

Tanya Khullar

Yi Qi

zhenzhouyu

Yu Gong

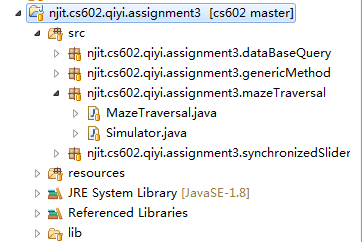
XiuruiHou

Xuan zhang

1. Maze Traversal

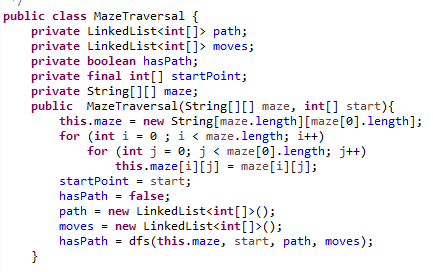
This problem is to ask us to solve a classic problem called maze problem. We can use backtracking to try all possible paths and find one valid path. To show the whole traversal process, we also write a simulator to display it.

1. We write two classes, one for solving the maze problem and the other one for display.

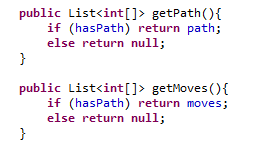


1. For the initiation part, we both store the current path and each move. For the maze Traversal itself, the current path is enough, but since we also want to display the whole process in the simulator, we store all the moves.

Note that, we don’t want to modify the original maze, so we make a copy first.



1. Public methods for external use



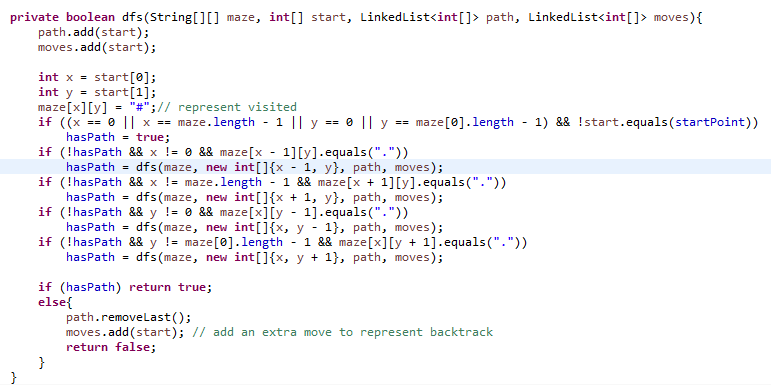
1. Use a recursive dfs method for backtracking

For each start point, we add the point to path and moves , and use “#” to represent “visited”

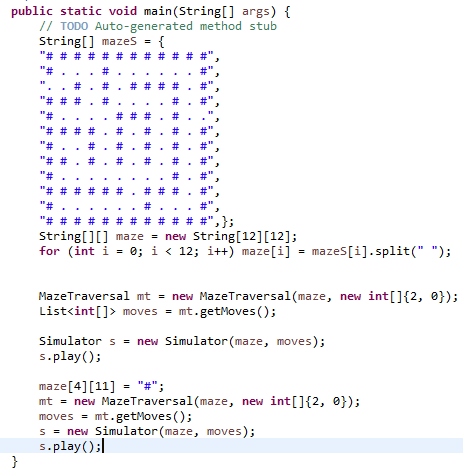
First we check whether we find an exit (boundaries of the maze), note that the exit must not be the same point with the start point.

Second, try each possible valid direction and make a recursive call. Note that, if we already find a path, we don’t need to make extra recursive call.

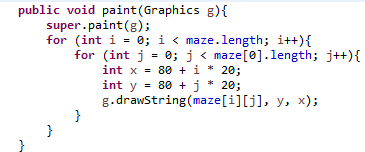
At last, if we cannot find path through this point, we do a backtracking, remove current point from “path”, and also add current point to the “moves” to represent backtracking



1. In the simulator, we first initiate a valid maze and play; then initiate an invalid maze and play.



1. We override paint method to draw the maze.



1. In the play method, if we can get a valid “move”, display the whole process; otherwise, show a dialog.

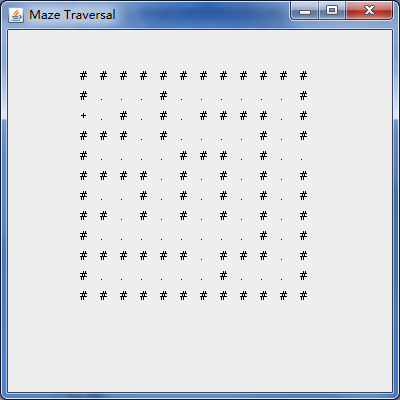
For each “move”, we update the maze, and if we meet a “move” twice, we remove that “move” (set it to its original value “.”)

Also, set a “1 second” interval between each move.

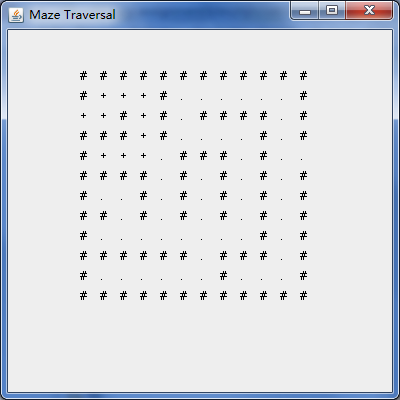


1. Final result

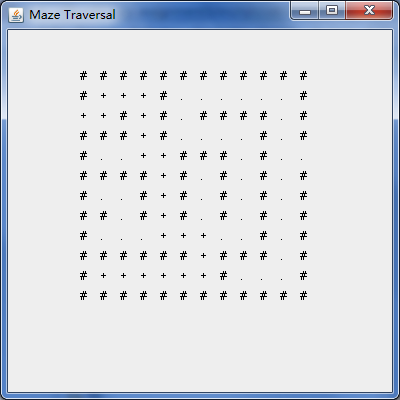
Start:



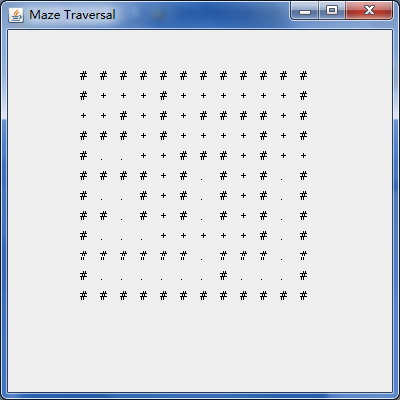
Go to the wrong way:



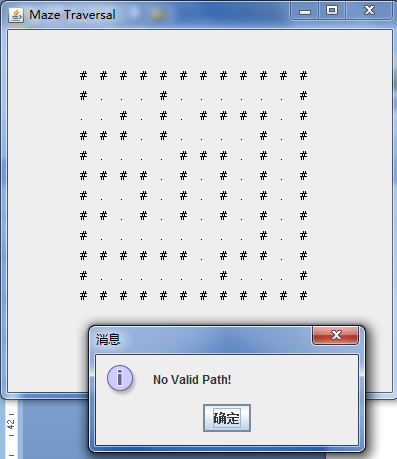
Go back and try another way:



Find an exit:



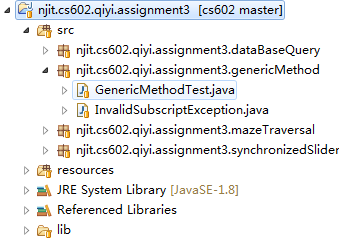
Invalid maze:



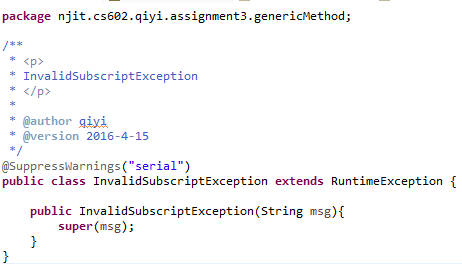
1. Print Array

This problem is to overload the original printArray method so that it can print a certain section of the input array.

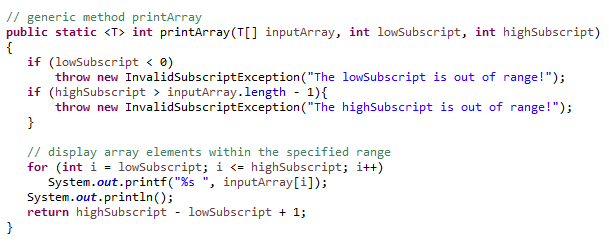
1. Rewrite the original genericMethodTest class and add an exception class for the certain type of exception.



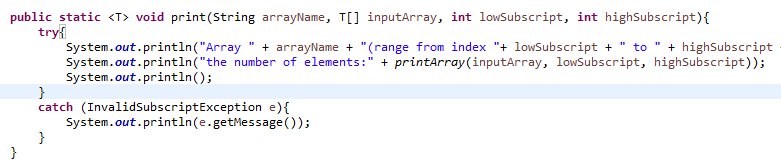
1. The exception class just extends the runtimeException and uses its parent’s method.



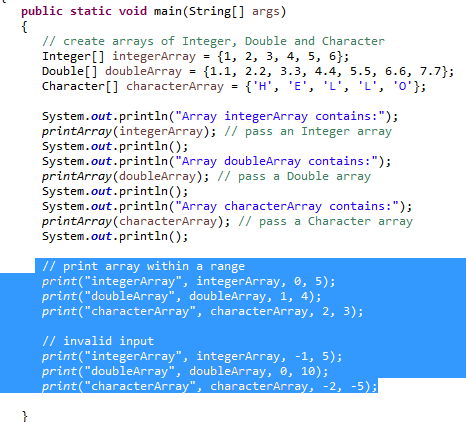
1. For the genericMethodTest, write an overloaded method to support low subscript and high subscript. Check the arguments first and throw corresponding exception.



1. Write a print method to print the results for the new printArray method

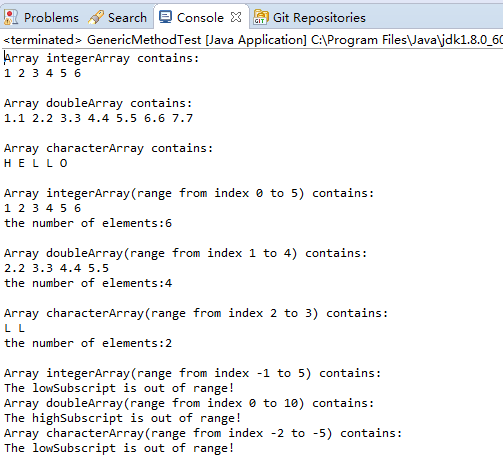


1. Add extra test cases for the new method



1. Final running result

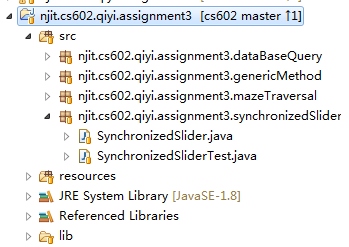
Note that, if we have more than one invalid argument, we just return the message for the first invalid argument (see the last test case)



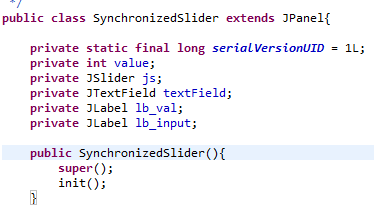
1. Synchronized Slider

This problem is to ask us to implement a synchronized slider, we need to maintain a value variable, and whenever it is updated, update the three components (displayed value, textfield, and slider).

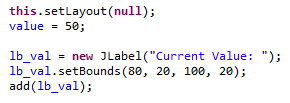
1. We write a class which extends JPanel to implement Synchronized Slider and write a test class which uses this JPanel to display.



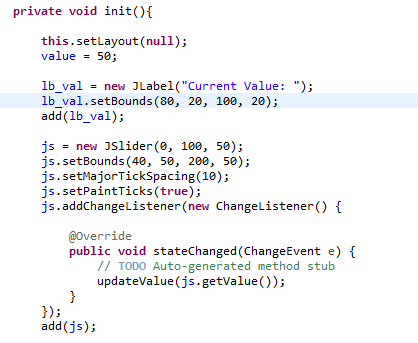
1. During the initiation, we create all the components we need



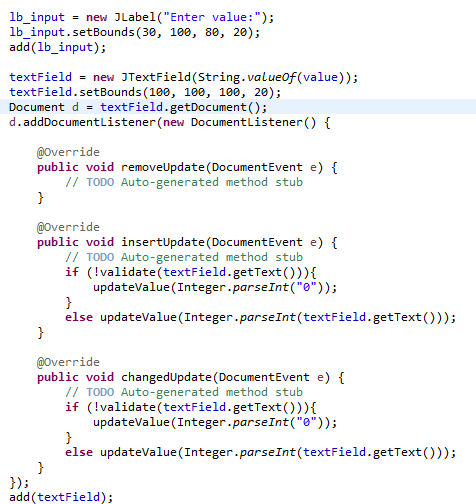
1. Value label: set the initiate value to 50, and create the value label.



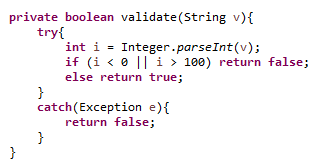
1. JSlider: create JSlider and whenever the state of the slider changes, call “update value” method (We will see this method later on)



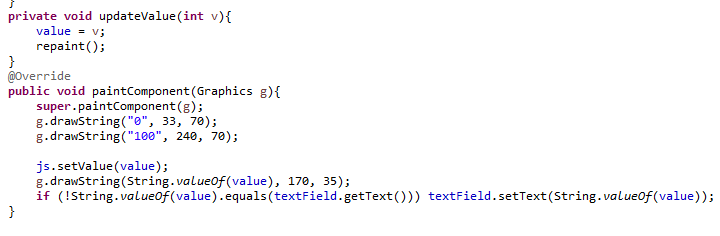
1. Text field: create textField, use document listener to listen the value changing and call “update value” method.



For the insert and update event above, if the input value is invalid, set it to 0(see previous screenshot).

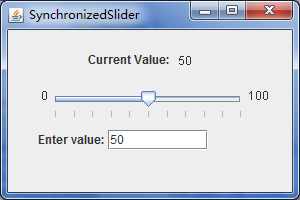


1. Update value: override the paintComponent method to draw the “value” on the panel and also set the value of JSlider and text field.

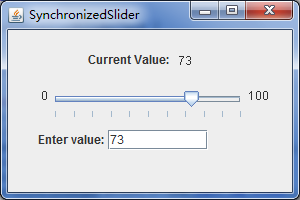


1. Final running result

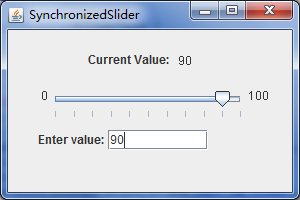
Start:



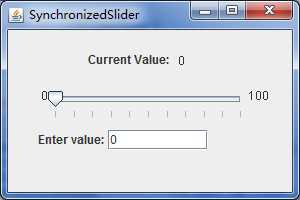
Drag the sliding block:



Input a valid value



Input an invalid value (automatically set to 0)



1. Book Query

This task is asking us to implement a database query application which supports predefined query and customized query.

1. In this program, we use mysql database and corresponding jdbc.

BookQuery.java: UI

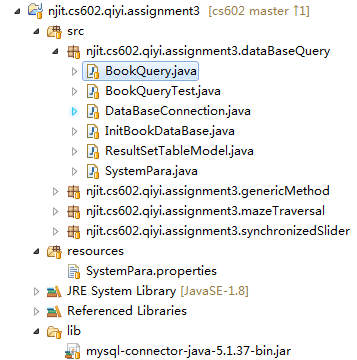
BookQueryTest.java: test program

DataBaseConnection.java: database connection util class from which we can get a database connection.

InitBookDataBase.java: initiate book database and corresponding test data.

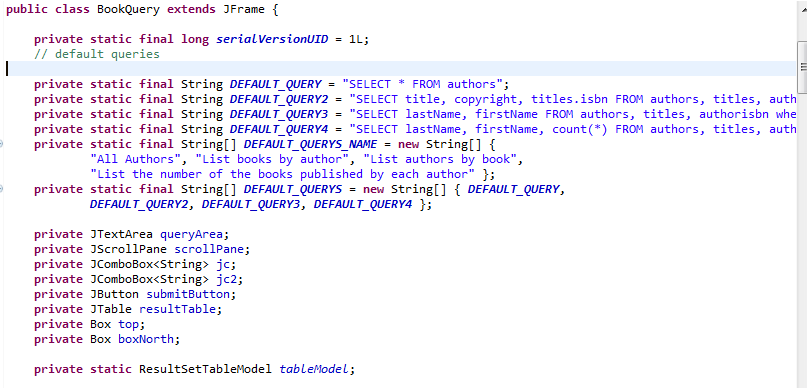
ResultSetTableModel.java: extends AbstractTableModel to display the query result.

SystemPara.java: read system parameter from SytemPara.properties which contains database connection information.

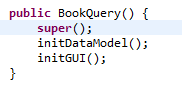


1. Define four predefined queries and components

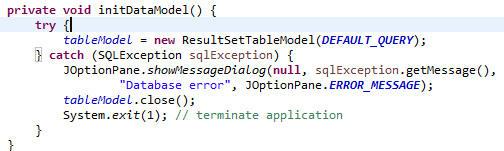
Besides the three queries provided by the book, we also add one: List the number of the books published by each author.



1. During the initiation, we need initiate UI and data Model

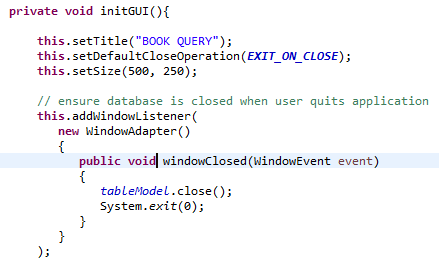


1. Initiate data model

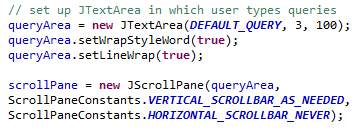


1. Initiate UI

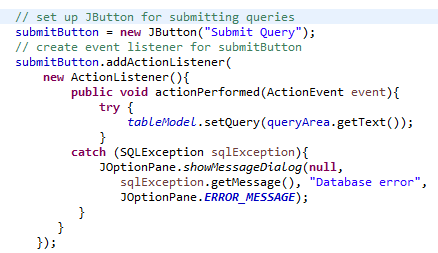
Add listener for closing database connection:



Set up text area and corresponding scroll pane for user to input sql



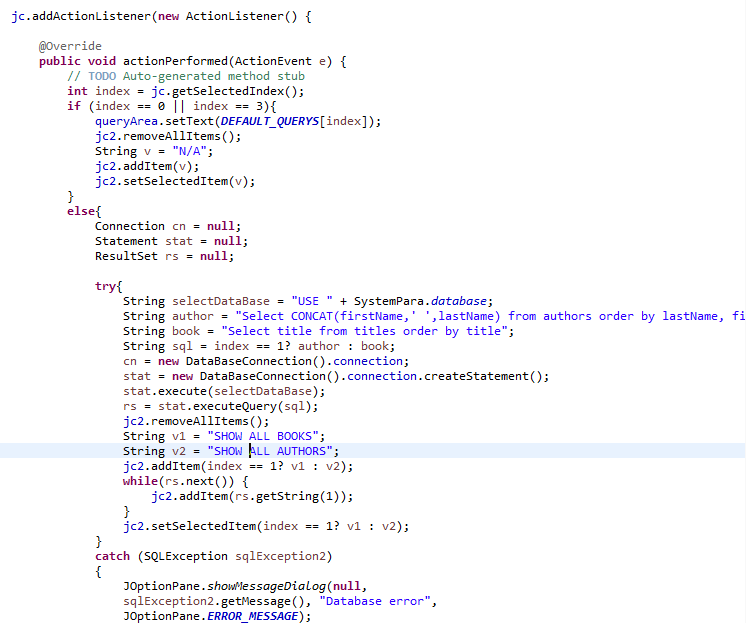
Set up submit button, if click the button, pass the query which comes from text area to the table model to display the result. If the query was failed, bring up a dialog



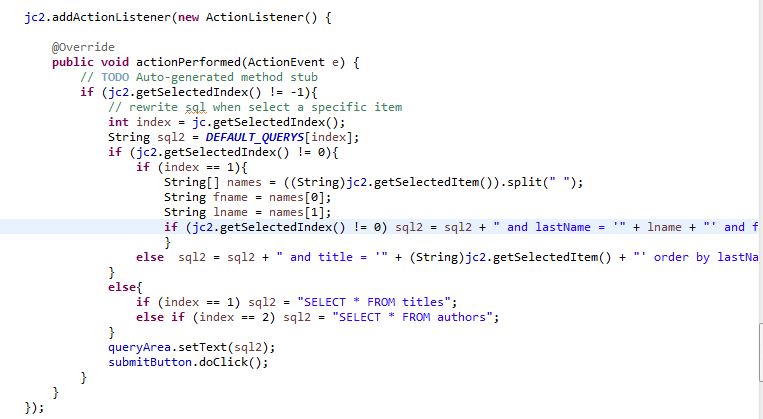
Create combo box for choose default sql and corresponding sub query



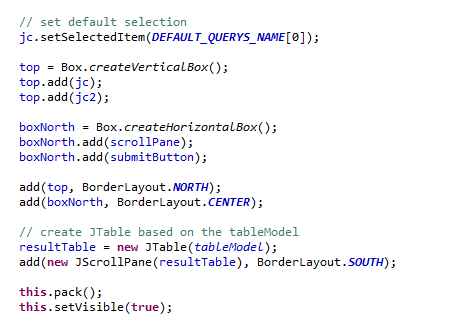
Add listener for the first combo box, if the user select the first and fourth predefined query, we don’t need a sub query, set the item of the second combo box to “N/A”; otherwise, search the items from database and add the searching result to the second combo box. After setting the items of the second combo box (for sub query), set the selected item, which will trigger the listener of the second combo box (will show it in the next screenshot)



Add listener for the second combo box, when a new item was selected, update the query string in the query area and click the button automatically, which will refresh the query result. Note that the query string will be the concatenation of predefined sql string and the value the user selected.



Set default selection and the layout of the components



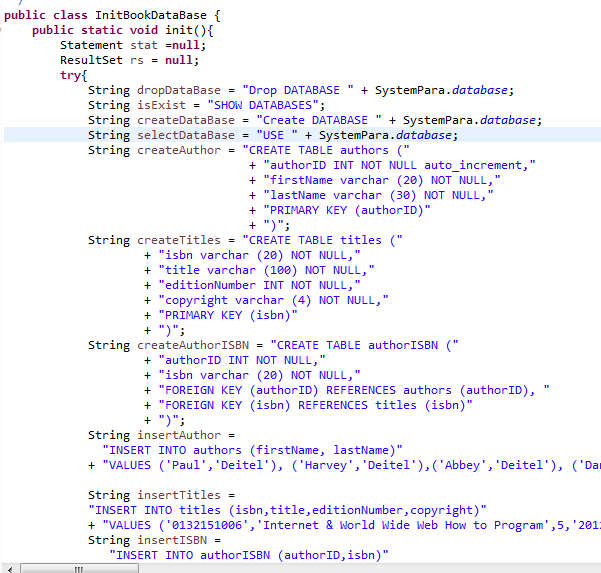
1. Database connection: create a database connection

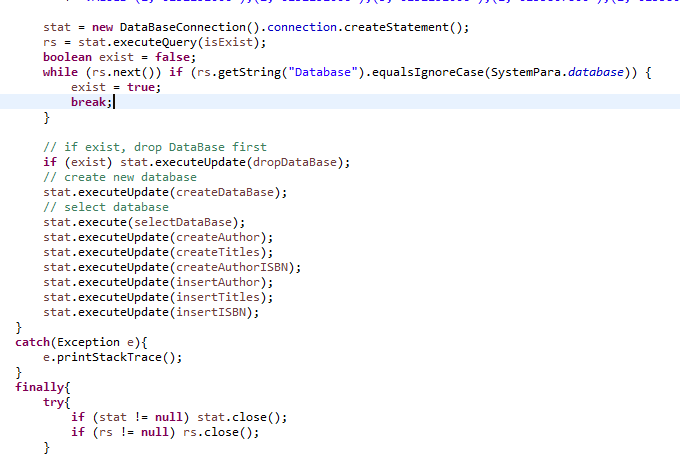


1. SystemPara: get system parameter from setting file.

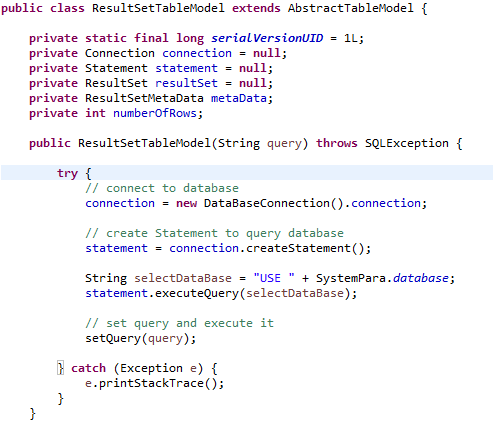


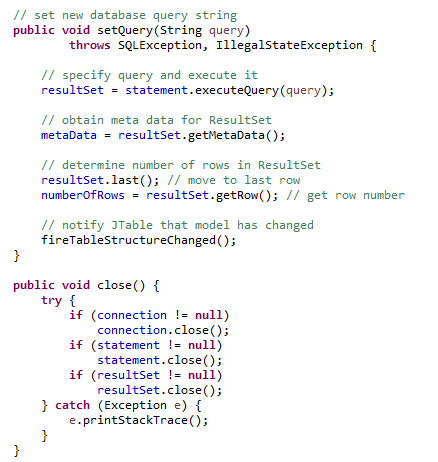
1. Initiate database and test data, if the database has already existed, drop the database first and then create the new one. Create table and corresponding test data.



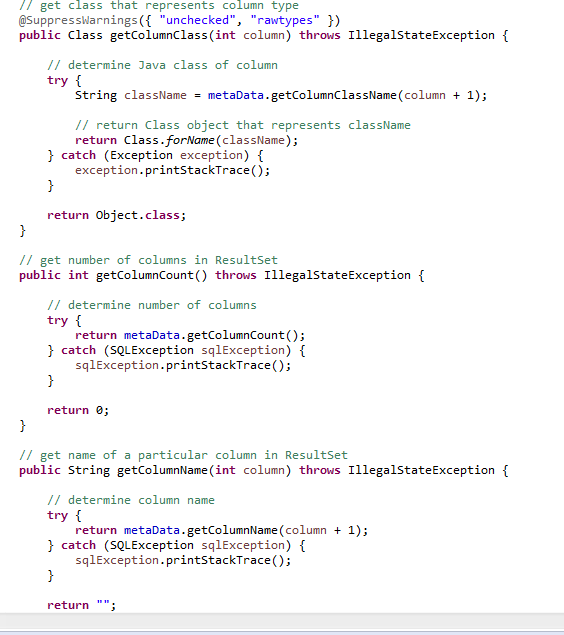


1. ResultSetTableModel: extends AbstractTableModel, get metadata from database query and update the table.



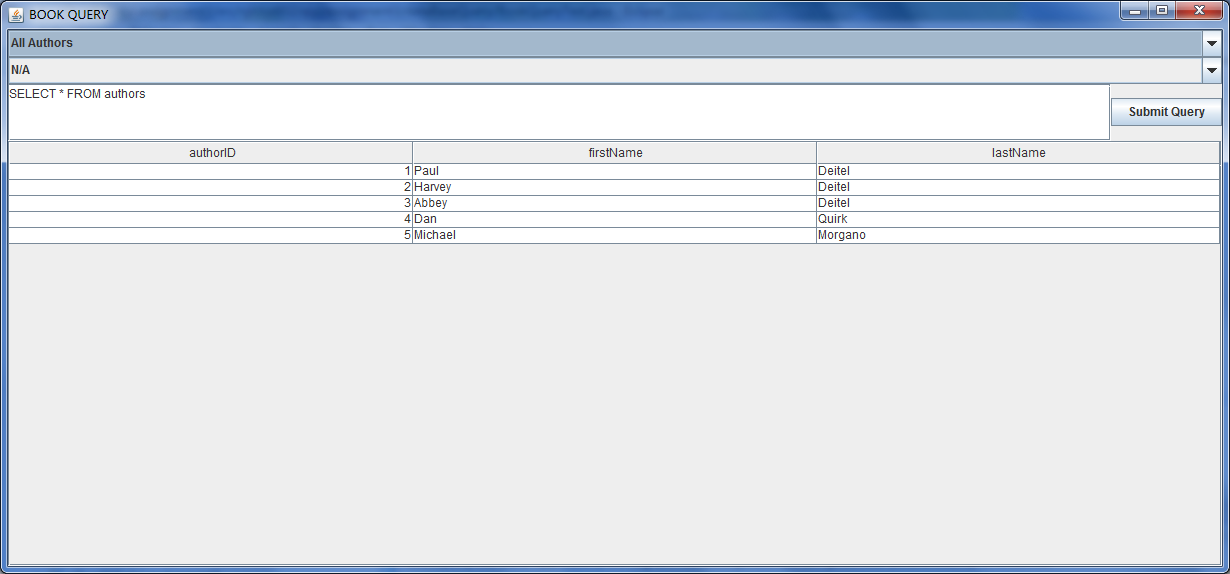


1. Corresponding overridden methods for AbstractTableModel

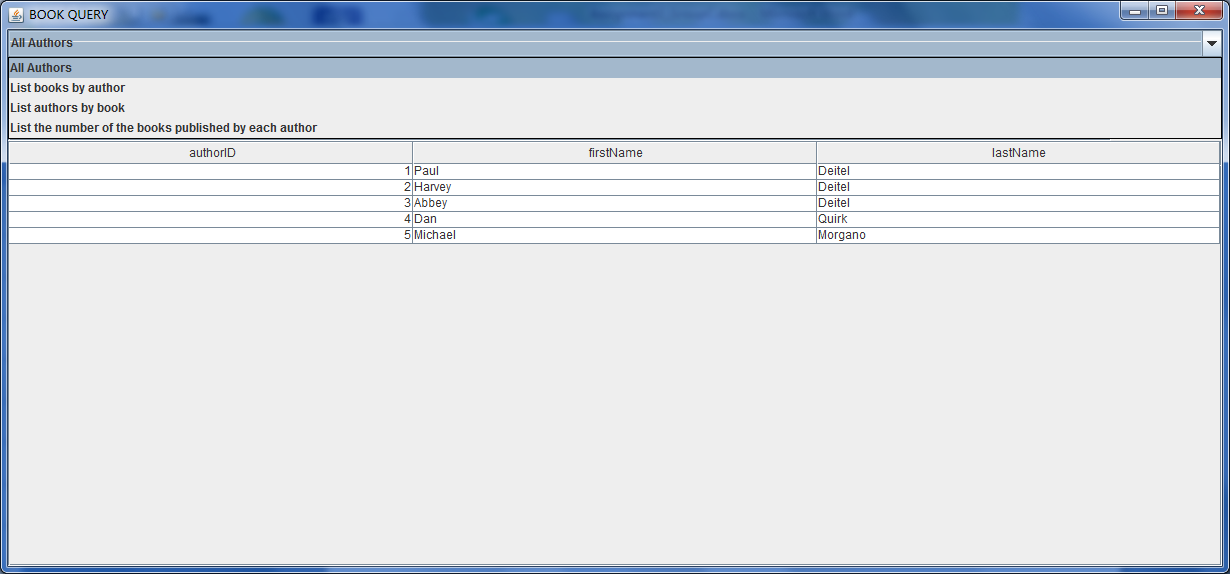


1. Final running result

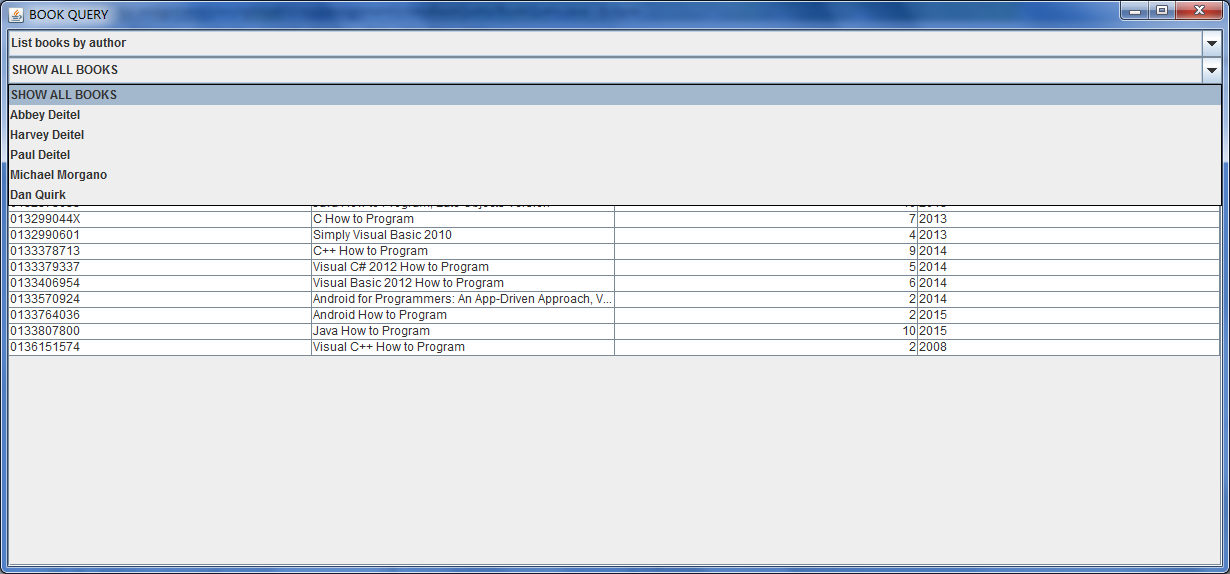
Start: select the first option by default, list all the authors



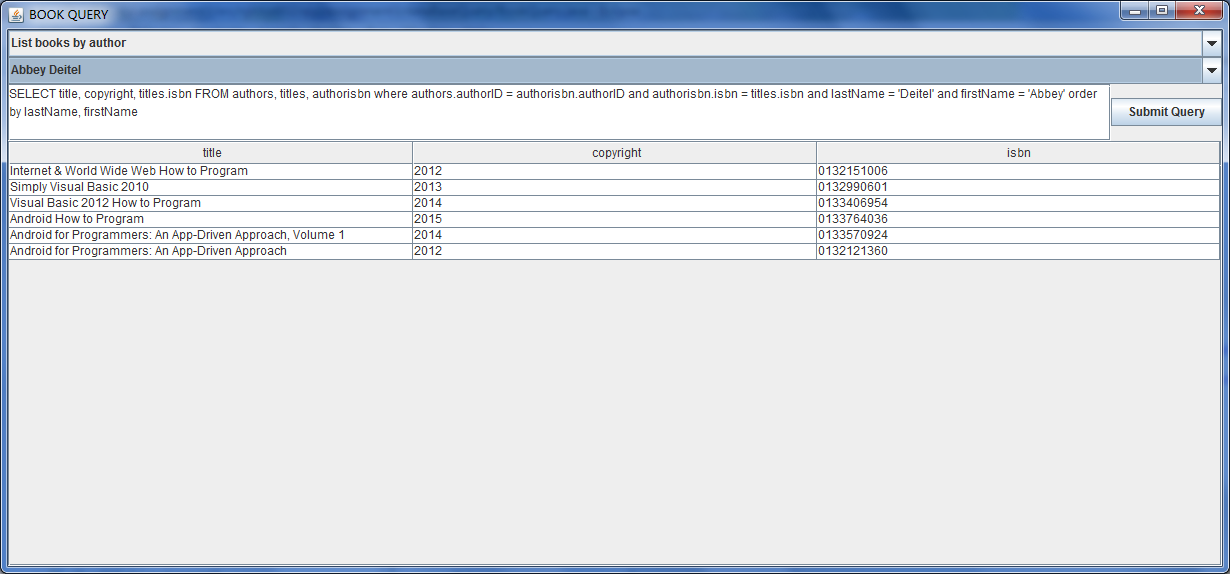
Four predefined queries,



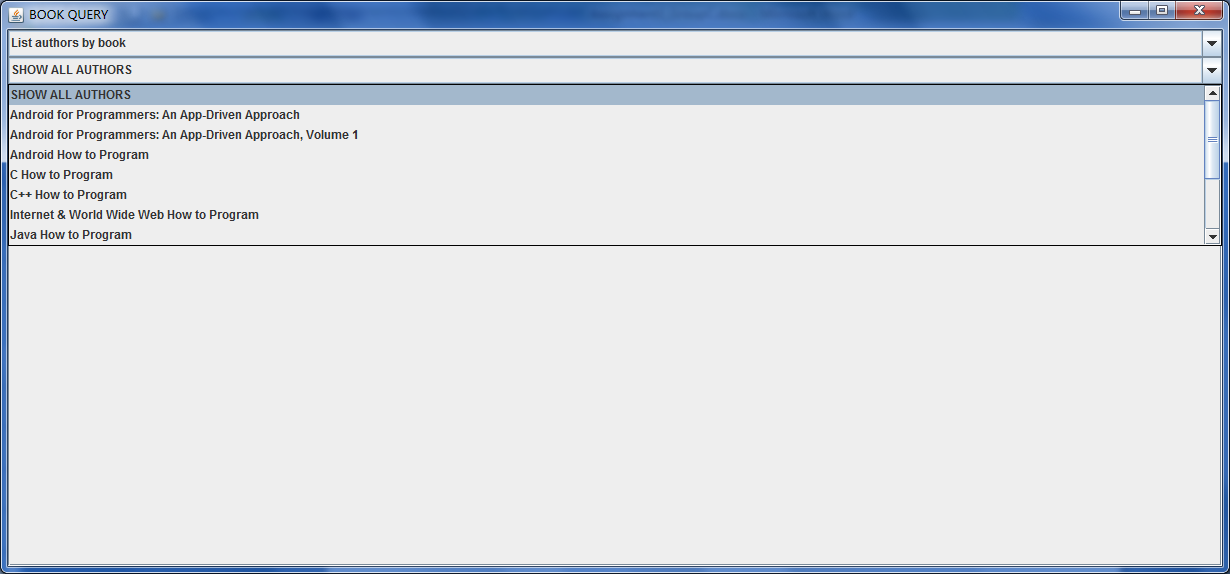
Select the second one: by default, it will show all books. We can also select a specific author, the authors will be ordered by last name, then by first name.



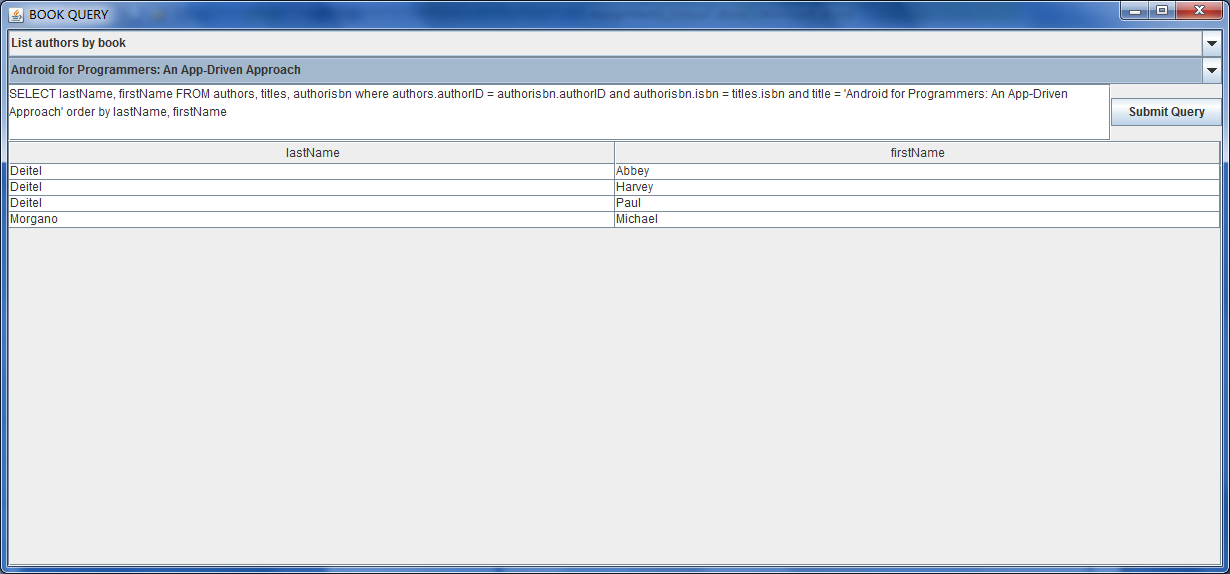
Select an author: list all books for that author. Include each book’s title, year and ISBN.



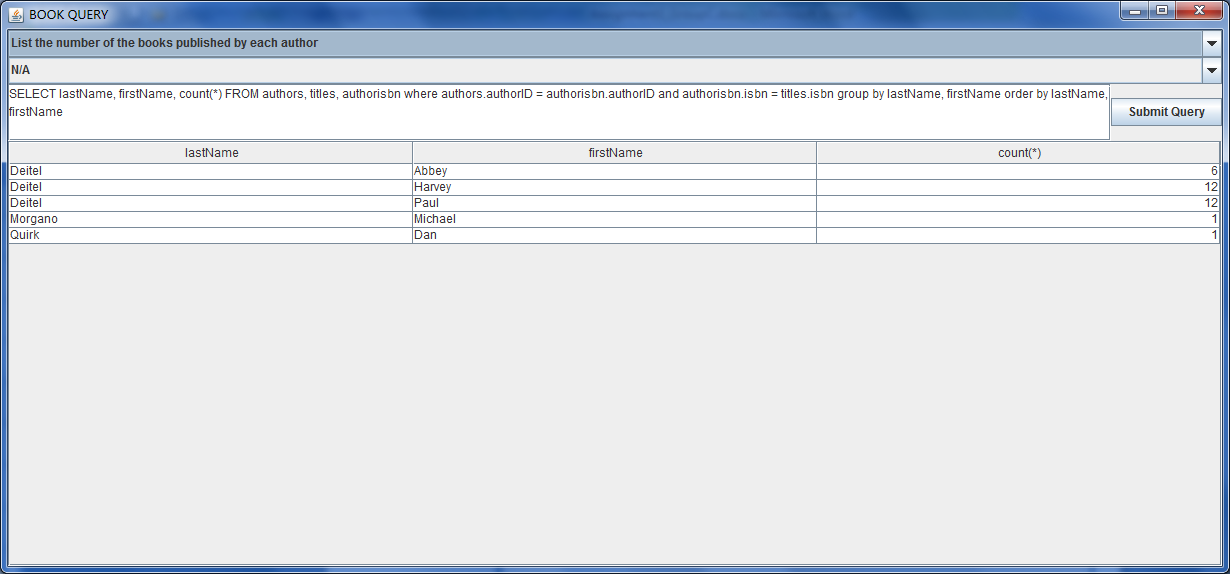
Select the third predefined query: by default, it will show all authors. We can also select a specific book



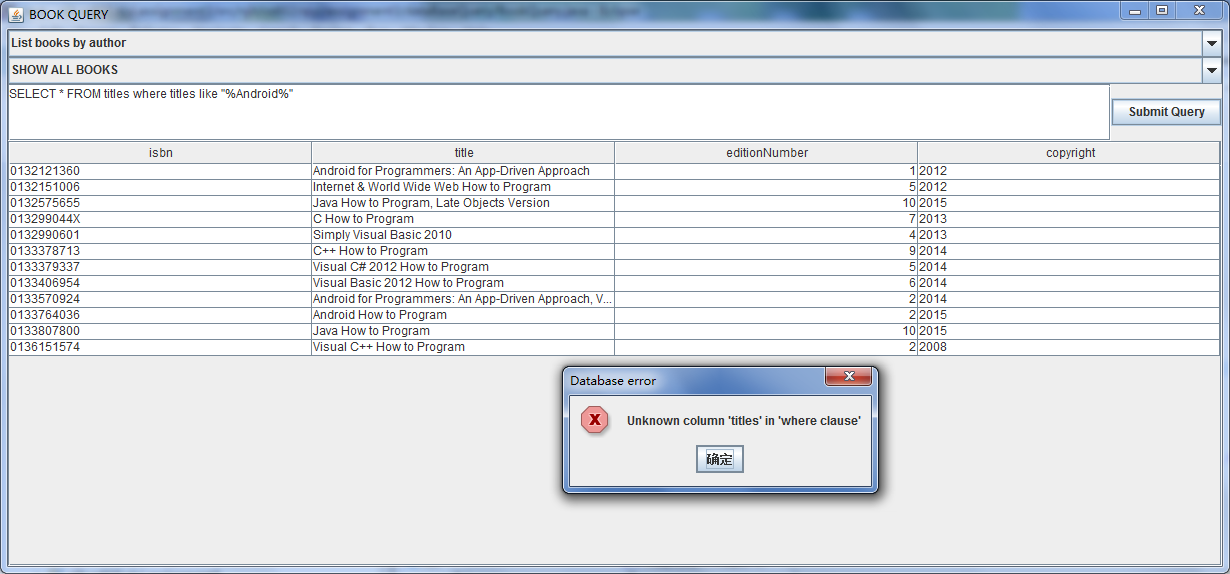
Select a book: list all authors for that book. Order the authors alphabetically by last name then by first name.



Select the fourth option:



Input an invalid customized query:



Input a valid customized query:

