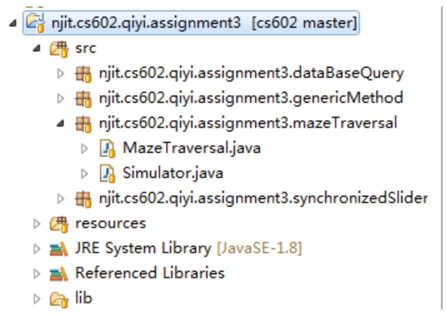
TEAM C
JAVA ASSIGNMENT -3

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.

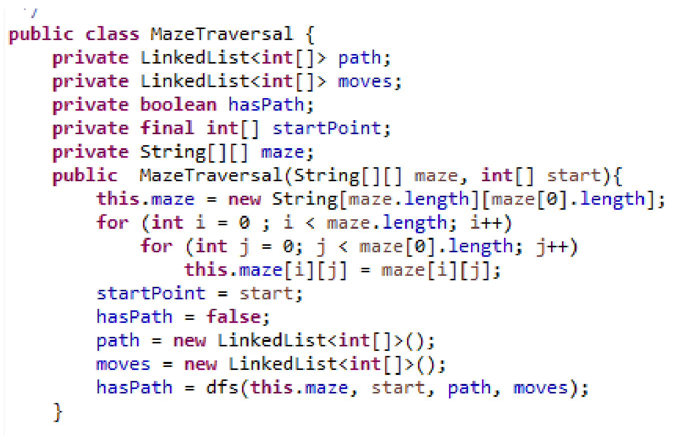
a. We write two classes, one for solving the maze problem and the other one for display.



```
njit.cs602.qiyi.assignment3  [cs602 master]
  src
    njit.cs602.qiyi.assignment3.dataBaseQuery
    njit.cs602.qiyi.assignment3.genericMethod
    njit.cs602.qiyi.assignment3.mazeTraversal
      MazeTraversal.java
      Simulator.java
    njit.cs602.qiyi.assignment3.synchronizedSlider
  resources
  JRE System Library [JavaSE-1.8]
  Referenced Libraries
  lib
```

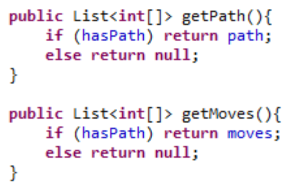
b. For the initiation part, we both store the current path and each move. For the maze Traversal itself, the current path is enough, but since we also want to display the whole process in the simulator, we store all the moves.

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



```
public class MazeTraversal {
    private LinkedList<int[]> path;
    private LinkedList<int[]> moves;
    private boolean hasPath;
    private final int[] startPoint;
    private String[][] maze;
    public  MazeTraversal(String[][] maze, int[] start){
        this.maze = new String[maze.length][maze[0].length];
        for (int i = 0 ; i < maze.length; i++)
            for (int j = 0; j < maze[0].length; j++)
                this.maze[i][j] = maze[i][j];
        startPoint = start;
        hasPath = false;
        path = new LinkedList<int[]>();
        moves = new LinkedList<int[]>();
        hasPath = dfs(this.maze, start, path, moves);
    }
```

c. Public methods for external use



```
public List<int[]> getPath(){
    if (hasPath) return path;
    else return null;
}

public List<int[]> getMoves(){
    if (hasPath) return moves;
    else return null;
}
```

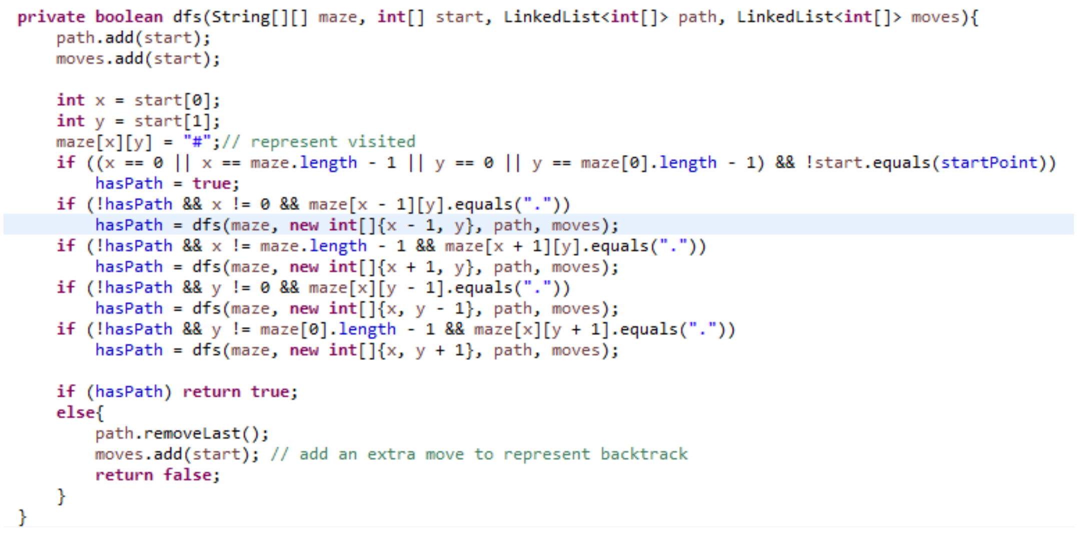
d. Use a recursive dfs method for backtracking

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.

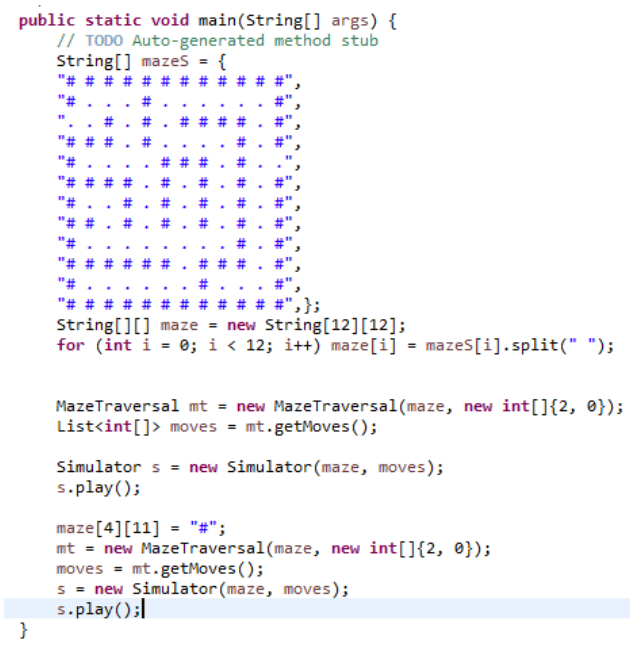
Second, try each possible valid direction and make a recursive call. Note that, if we already find a path, we don’t need to make extra recursive call.

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



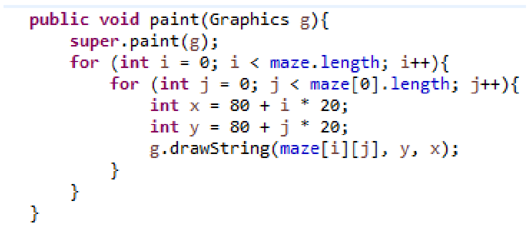
```
private boolean dfs(String[][] maze, int[] start, LinkedList<int[]> path, LinkedList<int[]> moves){
    path.add(start);
    moves.add(start);

    int x = start[0];
    int y = start[1];
    maze[x][y] = "#";// represent visited
    if ((x == 0 || x == maze.length - 1 || y == 0 || y == maze[0].length - 1) && !start.equals(startPoint))
        hasPath = true;
    if (!hasPath && x != 0 && maze[x - 1][y].equals("."))
        hasPath = dfs(maze, new int[]{x - 1, y}, path, moves);
    if (!hasPath && x != maze.length - 1 && maze[x + 1][y].equals("."))
        hasPath = dfs(maze, new int[]{x + 1, y}, path, moves);
    if (!hasPath && y != 0 && maze[x][y - 1].equals("."))
        hasPath = dfs(maze, new int[]{x, y - 1}, path, moves);
    if (!hasPath && y != maze[0].length - 1 && maze[x][y + 1].equals("."))
        hasPath = dfs(maze, new int[]{x, y + 1}, path, moves);

    if (hasPath) return true;
    else{
        path.removeLast();
        moves.add(start); // add an extra move to represent backtrack
        return false;
    }
}
```

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



```
public static void main(String[] args) {
    // TODO Auto-generated method stub
    String[] mazeS = {
    "# # # # # # # # # # # #",
    "# . . . # . . . . . . #",
    ". . # . # . # # # # . #",
    "# # # . # . . . . # . #",
    "# . . . . # # # . # . .",
    "# # # # . # . # . # . #",
    "# . . # . # . # . # . #",
    "# # . # . # . # . # . #",
    "# . . . . . . . . # . #",
    "# # # # # # . # # # . #",
    "# . . . . . . # . . . #",
    "# # # # # # # # # # # #",};
    String[][] maze = new String[12][12];
    for (int i = 0; i < 12; i++) maze[i] = mazeS[i].split(" ");


    MazeTraversal mt = new MazeTraversal(maze, new int[]{2, 0});
    List<int[]> moves = mt.getMoves();

    Simulator s = new Simulator(maze, moves);
    s.play();

    maze[4][11] = "#";
    mt = new MazeTraversal(maze, new int[]{2, 0});
    moves = mt.getMoves();
    s = new Simulator(maze, moves);
    s.play();
}
```

f. We override paint method to draw the maze.

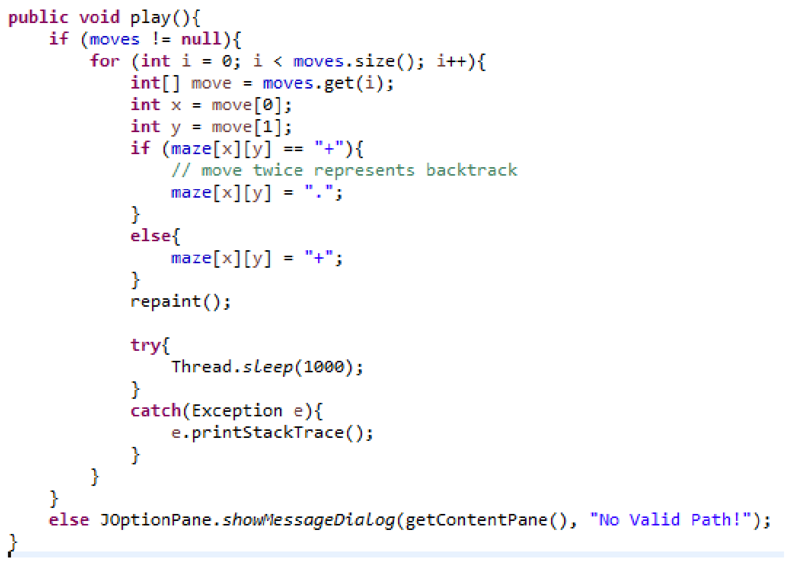


```
public void paint(Graphics g){
    super.paint(g);
    for (int i = 0; i < maze.length; i++){
        for (int j = 0; j < maze[0].length; j++){
            int x = 80 + i * 20;
            int y = 80 + j * 20;
            g.drawString(maze[i][j], y, x);
        }
    }
}
```

g. 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 “.”)

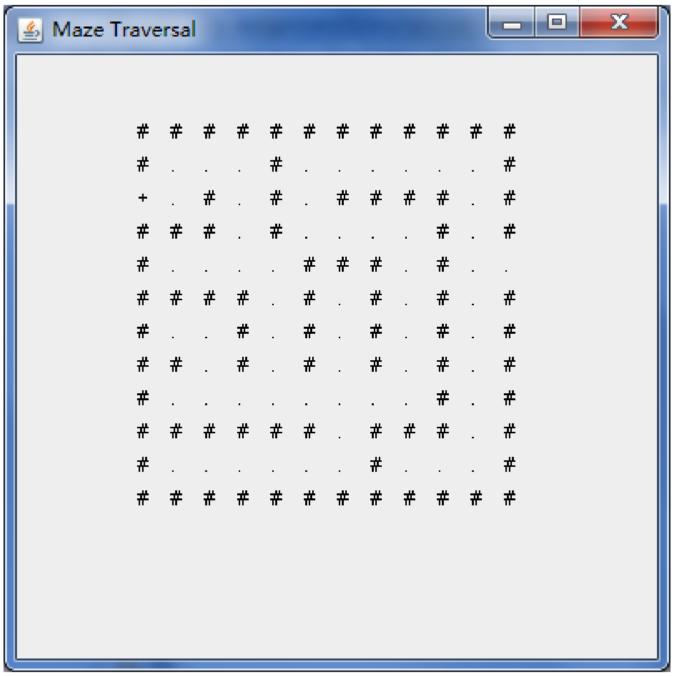
Also, set a “1 second” interval between each move.



```java
public void play(){
    if (moves != null){
        for (int i = 0; i < moves.size(); i++){
            int[] move = moves.get(i);
            int x = move[0];
            int y = move[1];
            if (maze[x][y] == "+"){
                // move twice represents backtrack
                maze[x][y] = ".";
            }
            else{
                maze[x][y] = "+";
            }
            repaint();

            try{
                Thread.sleep(1000);
            }
            catch(Exception e){
                e.printStackTrace();
            }
        }
    }
    else JOptionPane.showMessageDialog(getContentPane(), "No Valid Path!");
}
```

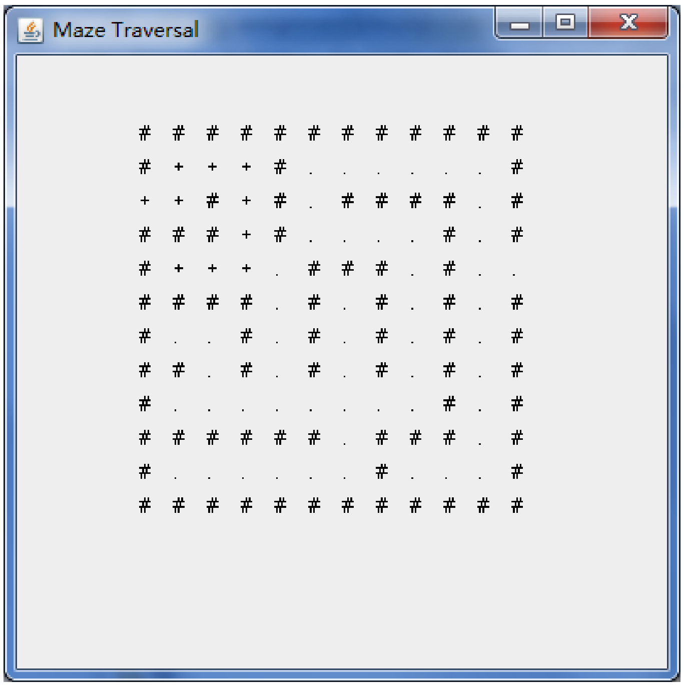
h. Final result

Start:



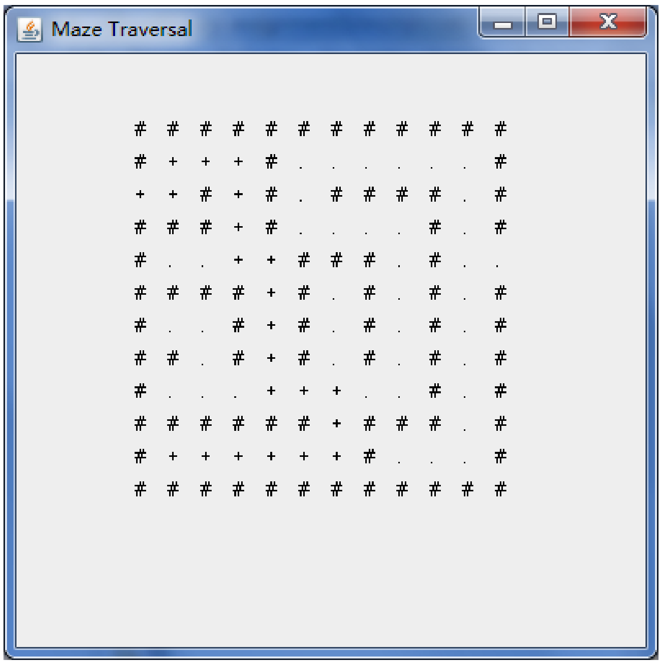
```
#  #  #  #  #  #  #  #  #  #  #  #
#  .  .  .  #  .  .  .  .  .  .  #
+  .  #  .  #  .  #  #  #  #  .  #
#  #  #  .  #  .  .  .  .  #  .  #
#  .  .  .  .  #  #  #  .  #  .  .
#  #  #  #  .  #  .  #  .  #  .  #
#  .  .  #  .  #  .  #  .  #  .  #
#  #  .  #  .  #  .  #  .  #  .  #
#  .  .  .  .  .  .  .  .  #  .  #
#  #  #  #  #  #  .  #  #  #  .  #
#  .  .  .  .  .  .  #  .  .  .  #
#  #  #  #  #  #  #  #  #  #  #  #
```

Go to the wrong way:



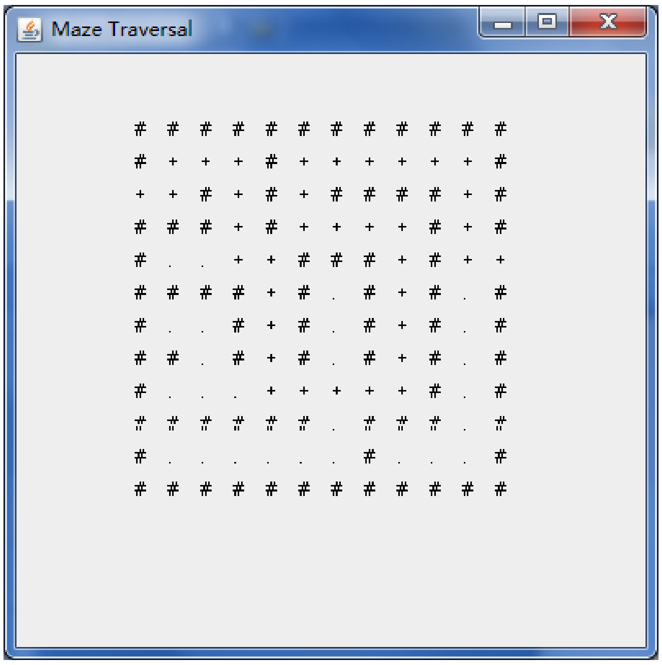
```
#  #  #  #  #  #  #  #  #  #  #  #
#  +  +  +  #  .  .  .  .  .  .  #
+  +  #  +  #  .  #  #  #  #  .  #
#  #  #  +  #  .  .  .  .  #  .  #
#  +  +  +  .  #  #  #  .  #  .  .
#  #  #  #  .  #  .  #  .  #  .  #
#  .  .  #  .  #  .  #  .  #  .  #
#  #  .  #  .  #  .  #  .  #  .  #
#  .  .  .  .  .  .  .  .  #  .  #
#  #  #  #  #  #  .  #  #  #  .  #
#  .  .  .  .  .  .  #  .  .  .  #
#  #  #  #  #  #  #  #  #  #  #  #
```

Go back and try another way:



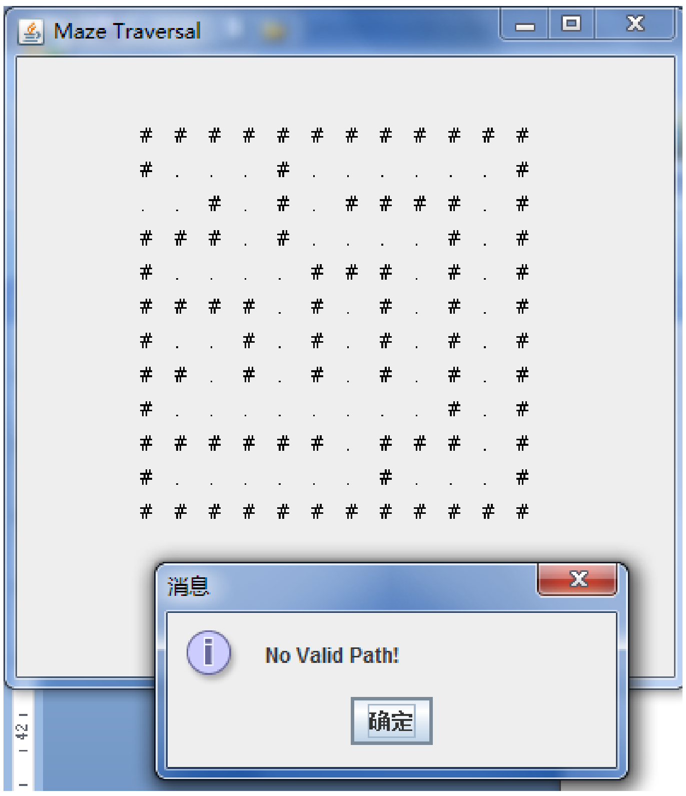
```
# # # # # # # # # # # #
# + + + # . . . . . . #
+ + # + # . # # # # . #
# # # + # . . . . # . #
# . . + + # # # . # . .
# # # # + # . # . # . #
# . . # + # . # . # . #
# # . # + # . # . # . #
# . . . + + + . . # . #
# # # # # # + # # # . #
# + + + + + + # . . . #
# # # # # # # # # # # #
```

Find an exit:



```
# # # # # # # # # # # #
# + + + # + + + + + + #
+ + # + # + # # # # + #
# # # + # + + + + # + #
# . . + + # # # + # + +
# # # # + # . # + # . #
# . . # + # . # + # . #
# # . # + # . # + # . #
# . . . + + + + + # . #
# # # # # # . # # # . #
# . . . . . . # . . . #
# # # # # # # # # # # #
```

Invalid maze:



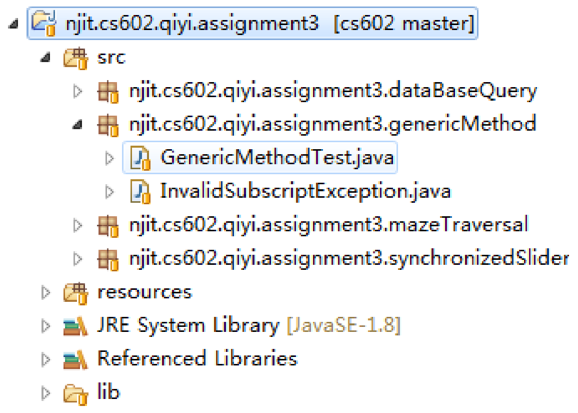
```
# # # # # # # # # # # #
# . . . # . . . . . . #
. . # . # . # # # # . #
# # # . # . . . . # . #
# . . . . # # # . # . #
# # # # . # . # . # . #
# . . # . # . # . # . #
# # . # . # . # . # . #
# . . . . . . . . # . #
# # # # # # . # # # . #
# . . . . . . # . . . #
# # # # # # # # # # # #
```

No Valid Path!

2. Print Array

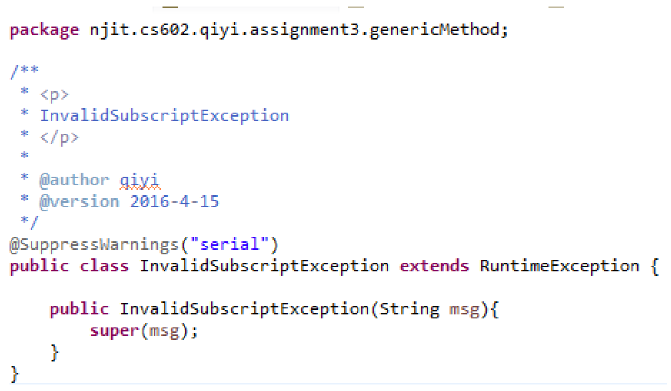
This problem is to overload the original printArray method so that it can print a certain section of the input array.

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



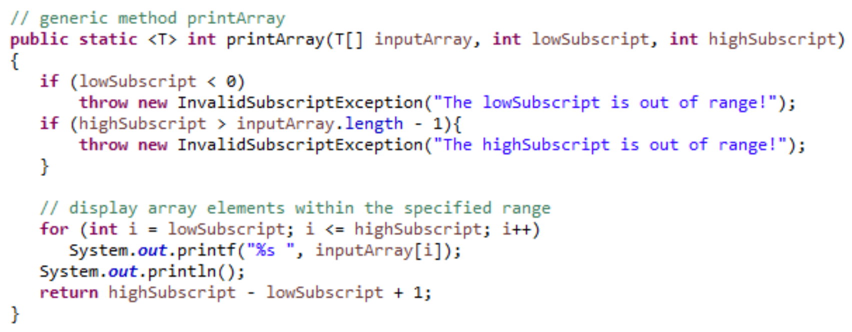
```
njit.cs602.qiyi.assignment3  [cs602 master]
  src
    njit.cs602.qiyi.assignment3.dataBaseQuery
    njit.cs602.qiyi.assignment3.genericMethod
      GenericMethodTest.java
      InvalidSubscriptException.java
    njit.cs602.qiyi.assignment3.mazeTraversal
    njit.cs602.qiyi.assignment3.synchronizedSlider
  resources
  JRE System Library [JavaSE-1.8]
  Referenced Libraries
  lib
```

b. The exception class just extends the runtimeException and uses its parent's method.



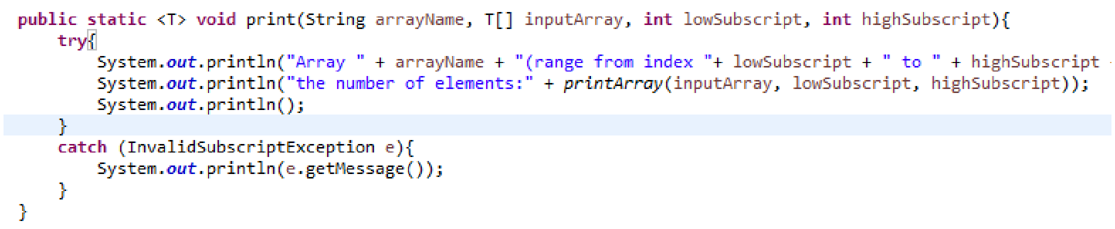
```java
package njit.cs602.qiyi.assignment3.genericMethod;

/**
 * <p>
 * InvalidSubscriptException
 * </p>
 *
 * @author qiyi
 * @version 2016-4-15
 */
@SuppressWarnings("serial")
public class InvalidSubscriptException extends RuntimeException {

    public InvalidSubscriptException(String msg){
        super(msg);
    }
}
```

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



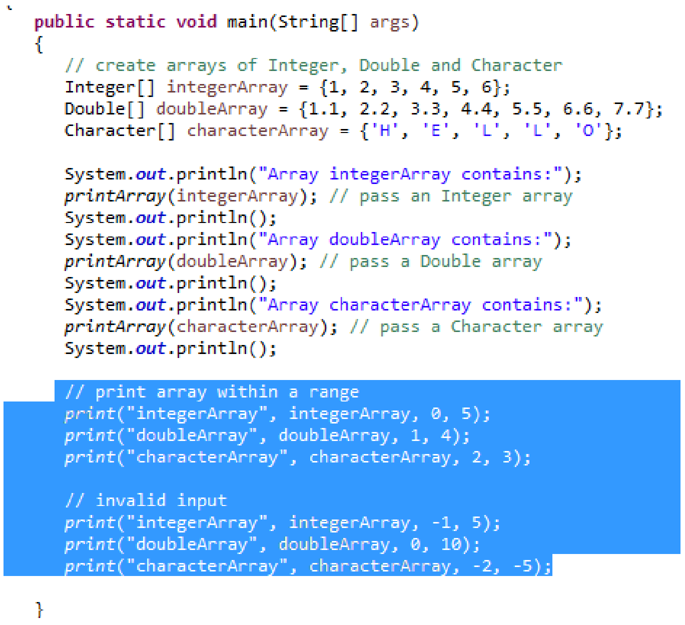
```
// generic method printArray
public static <T> int printArray(T[] inputArray, int lowSubscript, int highSubscript)
{
   if (lowSubscript < 0)
       throw new InvalidSubscriptException("The lowSubscript is out of range!");
   if (highSubscript > inputArray.length - 1){
       throw new InvalidSubscriptException("The highSubscript is out of range!");
   }

   // display array elements within the specified range
   for (int i = lowSubscript; i <= highSubscript; i++)
      System.out.printf("%s ", inputArray[i]);
   System.out.println();
   return highSubscript - lowSubscript + 1;
}
```

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



```
public static <T> void print(String arrayName, T[] inputArray, int lowSubscript, int highSubscript){
    try{
        System.out.println("Array " + arrayName + "(range from index "+ lowSubscript + " to " + highSubscript
        System.out.println("the number of elements:" + printArray(inputArray, lowSubscript, highSubscript));
        System.out.println();
    }
    catch (InvalidSubscriptException e){
        System.out.println(e.getMessage());
    }
}
```

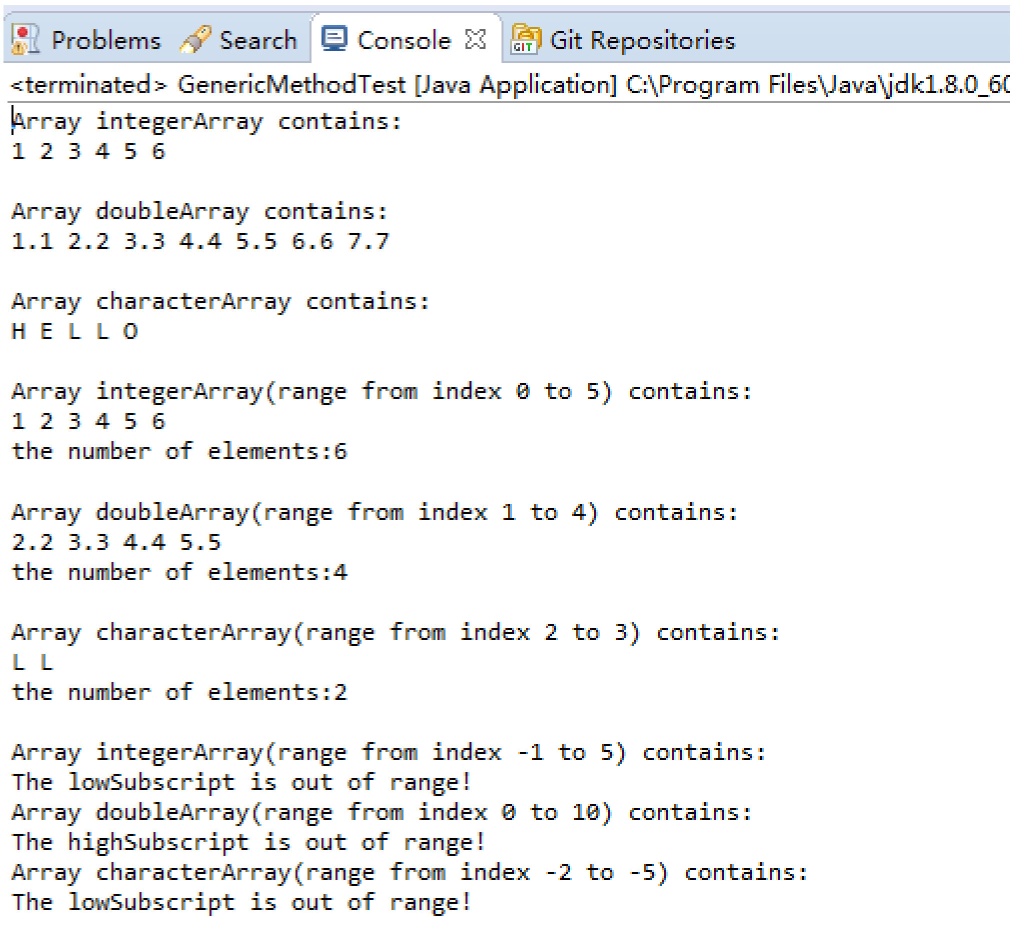
e. Add extra test cases for the new method



```
public static void main(String[] args)
{
    // create arrays of Integer, Double and Character
    Integer[] integerArray = {1, 2, 3, 4, 5, 6};
    Double[] doubleArray = {1.1, 2.2, 3.3, 4.4, 5.5, 6.6, 7.7};
    Character[] characterArray = {'H', 'E', 'L', 'L', 'O'};

    System.out.println("Array integerArray contains:");
    printArray(integerArray); // pass an Integer array
    System.out.println();
    System.out.println("Array doubleArray contains:");
    printArray(doubleArray); // pass a Double array
    System.out.println();
    System.out.println("Array characterArray contains:");
    printArray(characterArray); // pass a Character array
    System.out.println();

    // print array within a range
    print("integerArray", integerArray, 0, 5);
    print("doubleArray", doubleArray, 1, 4);
    print("characterArray", characterArray, 2, 3);

    // invalid input
    print("integerArray", integerArray, -1, 5);
    print("doubleArray", doubleArray, 0, 10);
    print("characterArray", characterArray, -2, -5);

}
```

f. 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)

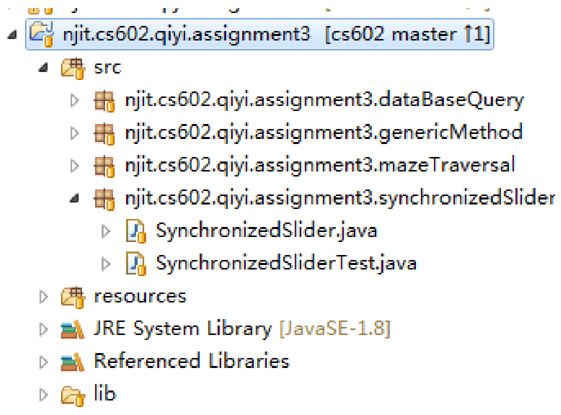


```
<terminated> GenericMethodTest [Java Application] C:\Program Files\Java\jdk1.8.0_6
Array integerArray contains:
1 2 3 4 5 6

Array doubleArray contains:
1.1 2.2 3.3 4.4 5.5 6.6 7.7

Array characterArray contains:
H E L L O

Array integerArray(range from index 0 to 5) contains:
1 2 3 4 5 6
the number of elements:6

Array doubleArray(range from index 1 to 4) contains:
2.2 3.3 4.4 5.5
the number of elements:4

Array characterArray(range from index 2 to 3) contains:
L L
the number of elements:2

Array integerArray(range from index -1 to 5) contains:
The lowSubscript is out of range!
Array doubleArray(range from index 0 to 10) contains:
The highSubscript is out of range!
Array characterArray(range from index -2 to -5) contains:
The lowSubscript is out of range!
```

3. Synchronized Slider

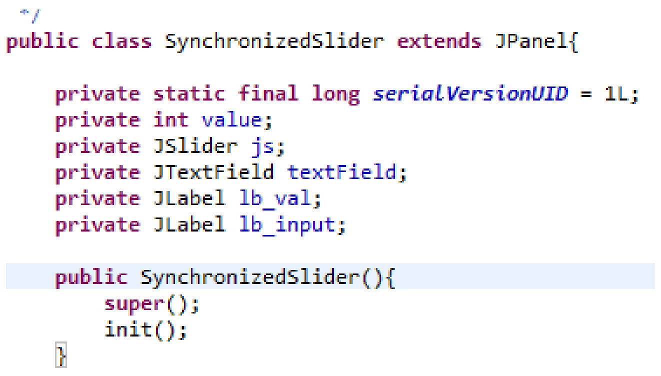
This problem is to ask us to implement a synchronized slider, we need to maintain a value variable, and whenever it is updated, update the three components (displayed value, textfield, and slider).

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



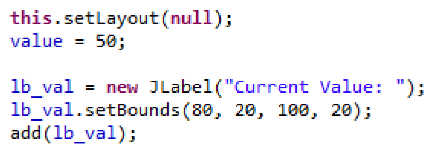
```
njit.cs602.qiyi.assignment3  [cs602 master ↑1]
  src
    njit.cs602.qiyi.assignment3.dataBaseQuery
    njit.cs602.qiyi.assignment3.genericMethod
    njit.cs602.qiyi.assignment3.mazeTraversal
    njit.cs602.qiyi.assignment3.synchronizedSlider
      SynchronizedSlider.java
      SynchronizedSliderTest.java
  resources
  JRE System Library [JavaSE-1.8]
  Referenced Libraries
  lib
```

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



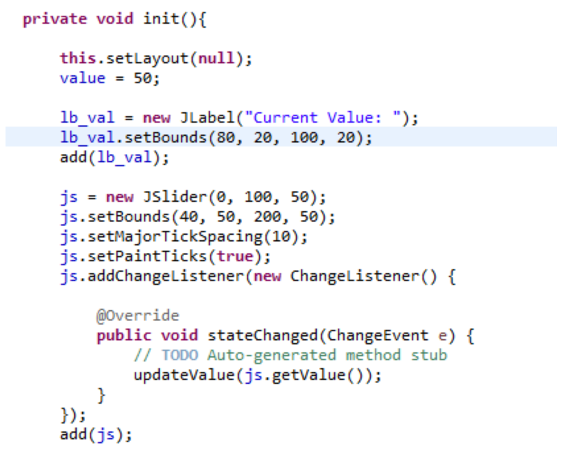
```
 */
public class SynchronizedSlider extends JPanel{

    private static final long serialVersionUID = 1L;
    private int value;
    private JSlider js;
    private JTextField textField;
    private JLabel lb_val;
    private JLabel lb_input;

    public SynchronizedSlider(){
        super();
        init();
    }
```

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



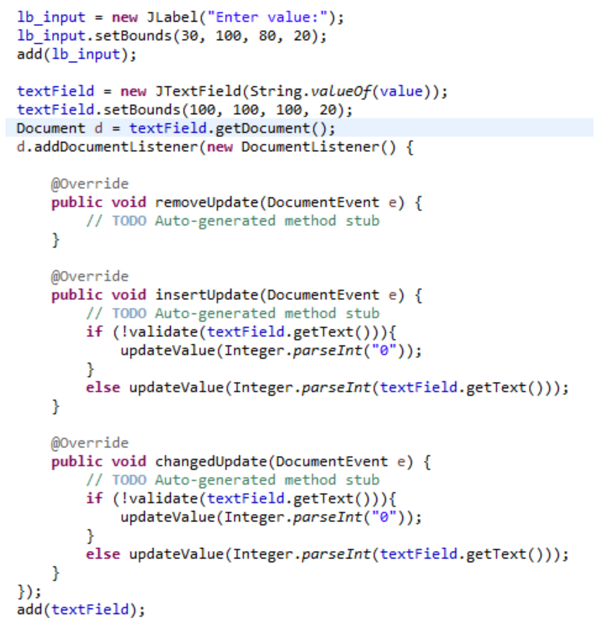
```
this.setLayout(null);
value = 50;

lb_val = new JLabel("Current Value: ");
lb_val.setBounds(80, 20, 100, 20);
add(lb_val);
```

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



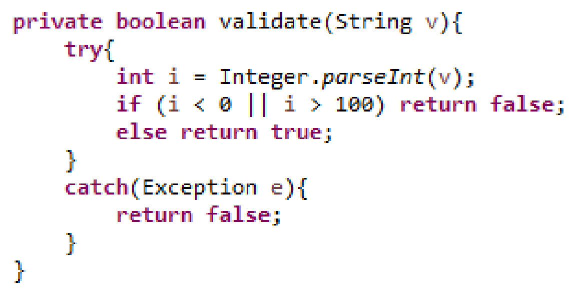
```
private void init(){

    this.setLayout(null);
    value = 50;

    lb_val = new JLabel("Current Value: ");
    lb_val.setBounds(80, 20, 100, 20);
    add(lb_val);

    js = new JSlider(0, 100, 50);
    js.setBounds(40, 50, 200, 50);
    js.setMajorTickSpacing(10);
    js.setPaintTicks(true);
    js.addChangeListener(new ChangeListener() {

        @Override
        public void stateChanged(ChangeEvent e) {
            // TODO Auto-generated method stub
            updateValue(js.getValue());
        }
    });
    add(js);
```

e. Text field: create textField, use document listener to listen the value changing and call "update value" method.



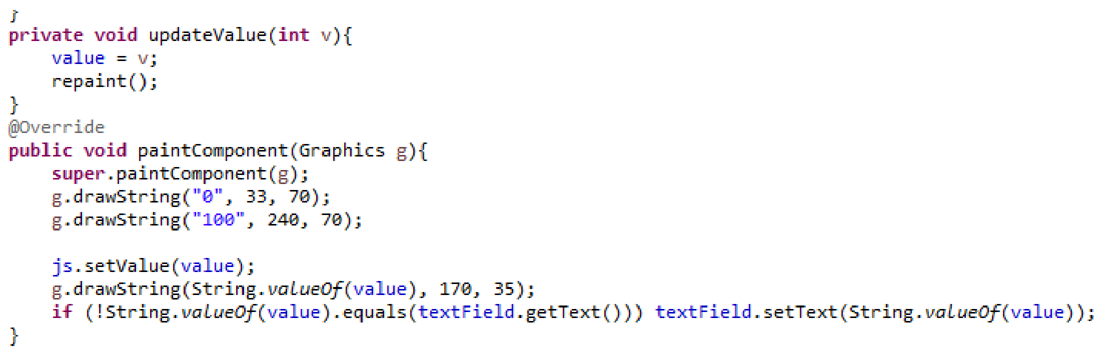
```
lb_input = new JLabel("Enter value:");
lb_input.setBounds(30, 100, 80, 20);
add(lb_input);

textField = new JTextField(String.valueOf(value));
textField.setBounds(100, 100, 100, 20);
Document d = textField.getDocument();
d.addDocumentListener(new DocumentListener() {

    @Override
    public void removeUpdate(DocumentEvent e) {
        // TODO Auto-generated method stub
    }

    @Override
    public void insertUpdate(DocumentEvent e) {
        // TODO Auto-generated method stub
        if (!validate(textField.getText())){
            updateValue(Integer.parseInt("0"));
        }
        else updateValue(Integer.parseInt(textField.getText()));
    }

    @Override
    public void changedUpdate(DocumentEvent e) {
        // TODO Auto-generated method stub
        if (!validate(textField.getText())){
            updateValue(Integer.parseInt("0"));
        }
        else updateValue(Integer.parseInt(textField.getText()));
    }
});
add(textField);
```

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



```
private boolean validate(String v){
    try{
        int i = Integer.parseInt(v);
        if (i < 0 || i > 100) return false;
        else return true;
    }
    catch(Exception e){
        return false;
    }
}
```

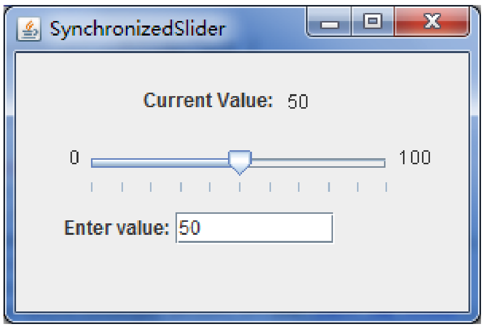
f. Update value: override the paintComponent method to draw the "value" on the panel and also set the value of JSlider and text field.



```
}
private void updateValue(int v){
    value = v;
    repaint();
}
@Override
public void paintComponent(Graphics g){
    super.paintComponent(g);
    g.drawString("0", 33, 70);
    g.drawString("100", 240, 70);

    js.setValue(value);
    g.drawString(String.valueOf(value), 170, 35);
    if (!String.valueOf(value).equals(textField.getText())) textField.setText(String.valueOf(value));
}
```

g. Final running result

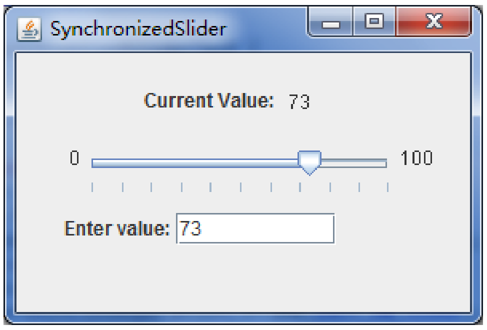
Start:



Current Value: 50

Enter value: 50

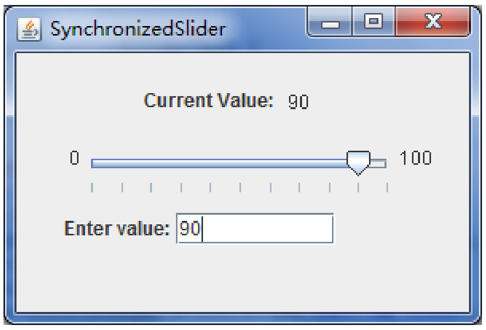
Drag the sliding block:



Current Value: 73

Enter value: 73

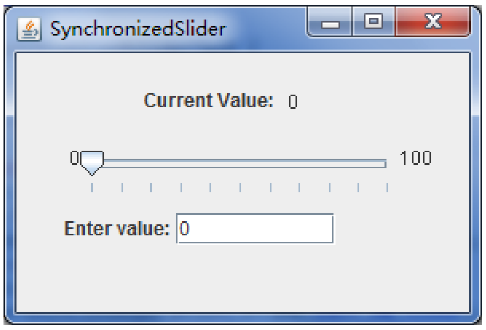
Input a valid value



Current Value: 90

Enter value: 90

Input an invalid value (automatically set to 0)



Current Value: 0

Enter value: 0

4. Book Query

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

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

BookQuery.java: UI

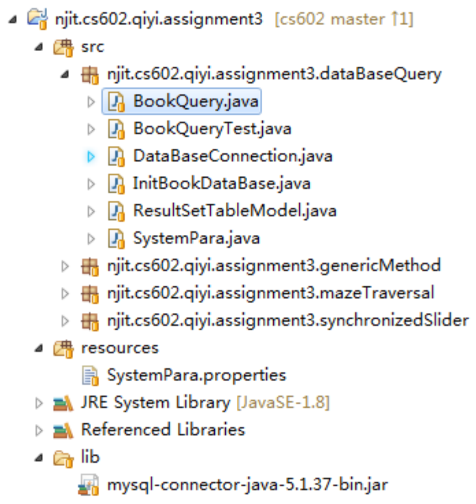
BookQueryTest.java: test program

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.

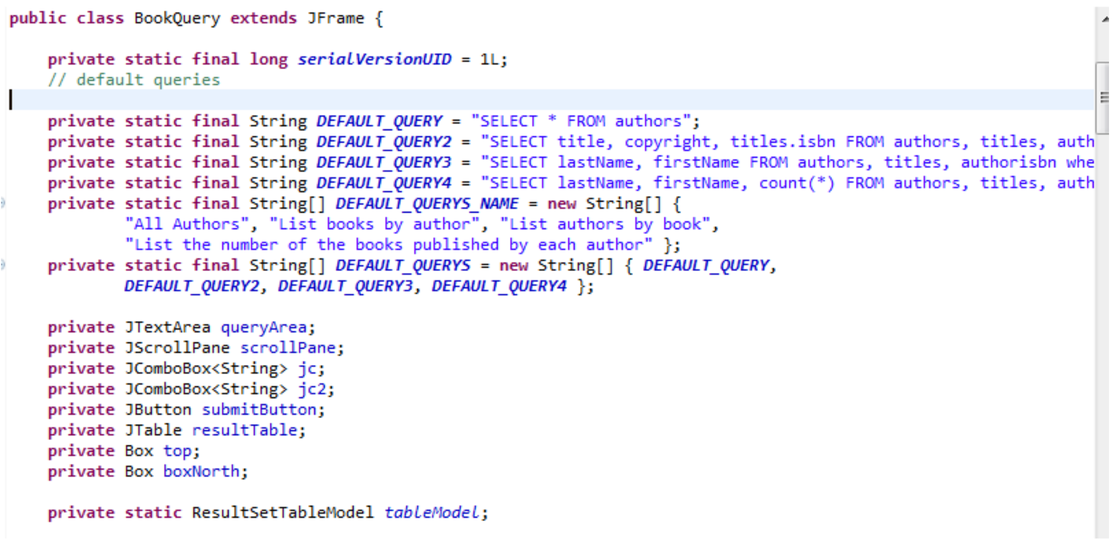
SystemPara.java: read system parameter from SytemPara.properties which contains database connection information.



```
njit.cs602.qiyi.assignment3 [cs602 master ↑1]
  src
    njit.cs602.qiyi.assignment3.dataBaseQuery
      BookQuery.java
      BookQueryTest.java
      DataBaseConnection.java
      InitBookDataBase.java
      ResultSetTableModel.java
      SystemPara.java
    njit.cs602.qiyi.assignment3.genericMethod
    njit.cs602.qiyi.assignment3.mazeTraversal
    njit.cs602.qiyi.assignment3.synchronizedSlider
  resources
    SystemPara.properties
  JRE System Library [JavaSE-1.8]
  Referenced Libraries
  lib
    mysql-connector-java-5.1.37-bin.jar
```

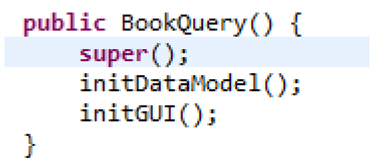
b. Define four predefined queries and components

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



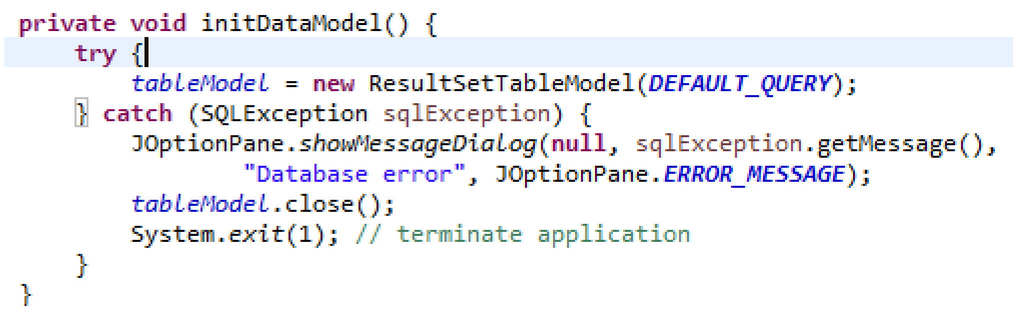
```
public class BookQuery extends JFrame {

    private static final long serialVersionUID = 1L;
    // default queries

    private static final String DEFAULT_QUERY = "SELECT * FROM authors";
    private static final String DEFAULT_QUERY2 = "SELECT title, copyright, titles.isbn FROM authors, titles, auth
    private static final String DEFAULT_QUERY3 = "SELECT lastName, firstName FROM authors, titles, authorisbn whe
    private static final String DEFAULT_QUERY4 = "SELECT lastName, firstName, count(*) FROM authors, titles, auth
    private static final String[] DEFAULT_QUERYS_NAME = new String[] {
            "All Authors", "List books by author", "List authors by book",
            "List the number of the books published by each author" };
    private static final String[] DEFAULT_QUERYS = new String[] { DEFAULT_QUERY,
            DEFAULT_QUERY2, DEFAULT_QUERY3, DEFAULT_QUERY4 };

    private JTextArea queryArea;
    private JScrollPane scrollPane;
    private JComboBox<String> jc;
    private JComboBox<String> jc2;
    private JButton submitButton;
    private JTable resultTable;
    private Box top;
    private Box boxNorth;

    private static ResultSetTableModel tableModel;
```

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



```
public BookQuery() {
    super();
    initDataModel();
    initGUI();
}
```

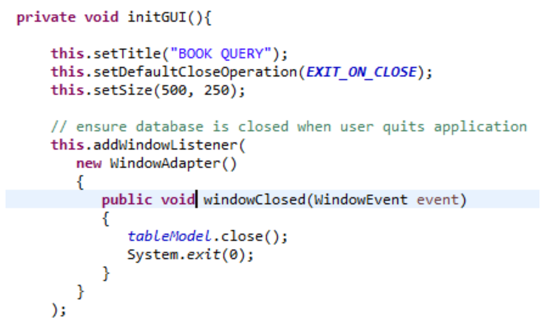
d. Initiate data model



```
private void initDataModel() {
    try {
        tableModel = new ResultSetTableModel(DEFAULT_QUERY);
    } catch (SQLException sqlException) {
        JOptionPane.showMessageDialog(null, sqlException.getMessage(),
                "Database error", JOptionPane.ERROR_MESSAGE);
        tableModel.close();
        System.exit(1); // terminate application
    }
}
```

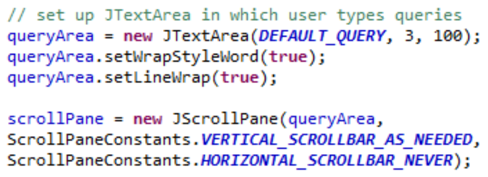
e. Initiate UI

Add listener for closing database connection:



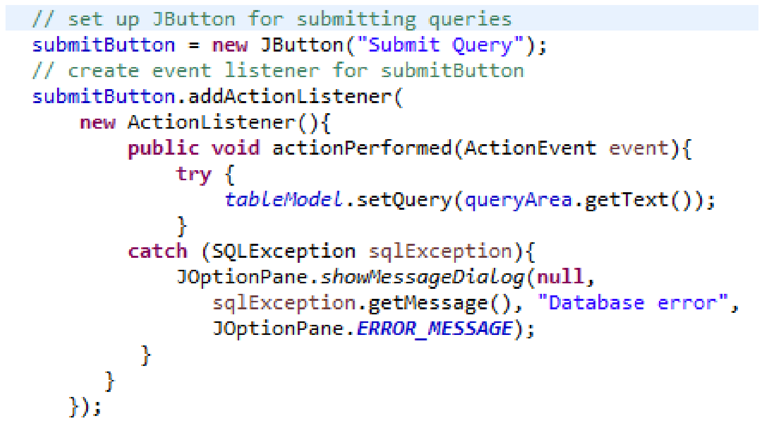
```
private void initGUI(){

    this.setTitle("BOOK QUERY");
    this.setDefaultCloseOperation(EXIT_ON_CLOSE);
    this.setSize(500, 250);

    // ensure database is closed when user quits application
    this.addWindowListener(
       new WindowAdapter()
       {
          public void windowClosed(WindowEvent event)
          {
             tableModel.close();
             System.exit(0);
          }
       }
    );
```

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



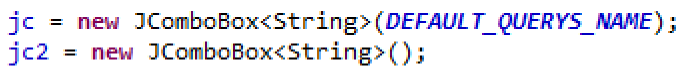
```
// set up JTextArea in which user types queries
queryArea = new JTextArea(DEFAULT_QUERY, 3, 100);
queryArea.setWrapStyleWord(true);
queryArea.setLineWrap(true);

scrollPane = new JScrollPane(queryArea,
ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED,
ScrollPaneConstants.HORIZONTAL_SCROLLBAR_NEVER);
```

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



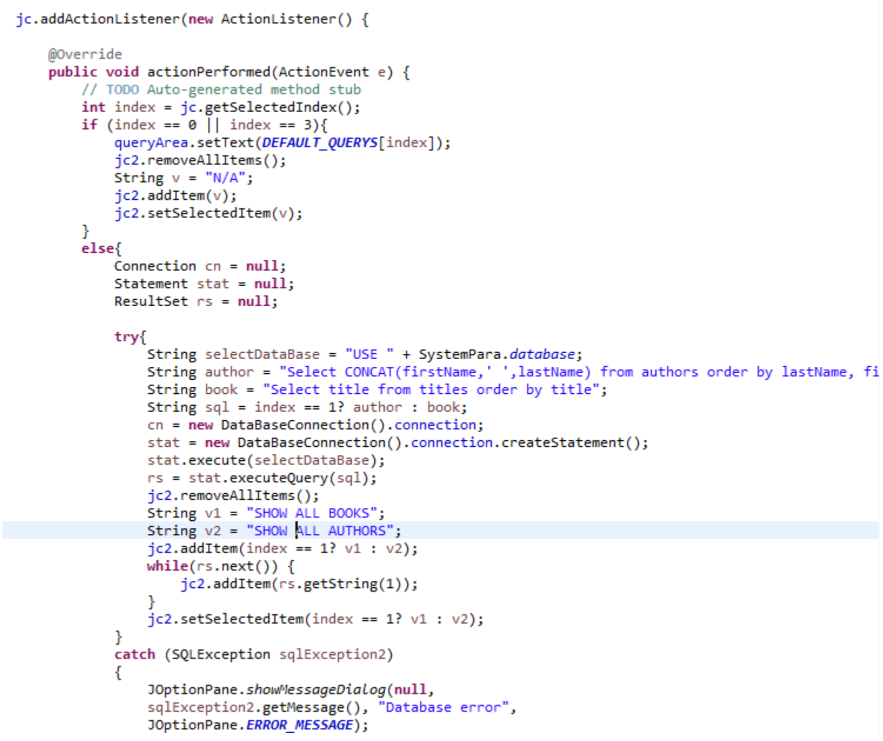
```
// set up JButton for submitting queries
submitButton = new JButton("Submit Query");
// create event listener for submitButton
submitButton.addActionListener(
    new ActionListener(){
        public void actionPerformed(ActionEvent event){
            try {
                tableModel.setQuery(queryArea.getText());
            }
        catch (SQLException sqlException){
            JOptionPane.showMessageDialog(null,
               sqlException.getMessage(), "Database error",
               JOptionPane.ERROR_MESSAGE);
         }
       }
   });
```

Create combo box for choose default sql and corresponding sub query



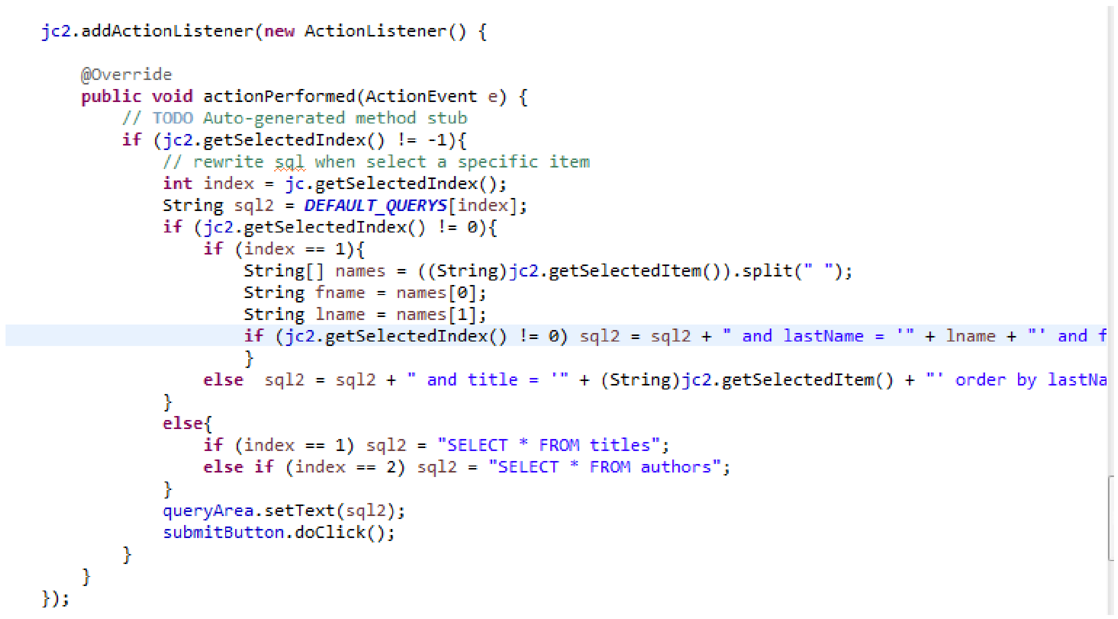
```
jc = new JComboBox<String>(DEFAULT_QUERYS_NAME);
jc2 = new JComboBox<String>();
```

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)



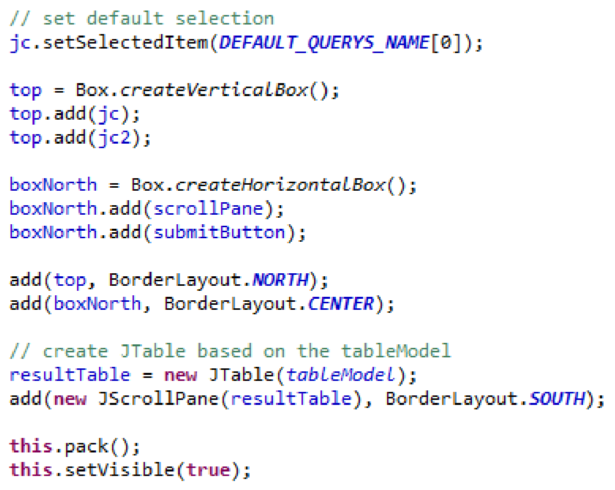
```
jc.addActionListener(new ActionListener() {

    @Override
    public void actionPerformed(ActionEvent e) {
        // TODO Auto-generated method stub
        int index = jc.getSelectedIndex();
        if (index == 0 || index == 3){
            queryArea.setText(DEFAULT_QUERYS[index]);
            jc2.removeAllItems();
            String v = "N/A";
            jc2.addItem(v);
            jc2.setSelectedItem(v);
        }
        else{
            Connection cn = null;
            Statement stat = null;
            ResultSet rs = null;

            try{
                String selectDataBase = "USE " + SystemPara.database;
                String author = "Select CONCAT(firstName,' ',lastName) from authors order by lastName, fi
                String book = "Select title from titles order by title";
                String sql = index == 1? author : book;
                cn = new DataBaseConnection().connection;
                stat = new DataBaseConnection().connection.createStatement();
                stat.execute(selectDataBase);
                rs = stat.executeQuery(sql);
                jc2.removeAllItems();
                String v1 = "SHOW ALL BOOKS";
                String v2 = "SHOW ALL AUTHORS";
                jc2.addItem(index == 1? v1 : v2);
                while(rs.next()) {
                    jc2.addItem(rs.getString(1));
                }
                jc2.setSelectedItem(index == 1? v1 : v2);
            }
            catch (SQLException sqlException2)
            {
                JOptionPane.showMessageDialog(null,
                sqlException2.getMessage(), "Database error",
                JOptionPane.ERROR_MESSAGE);
```

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.



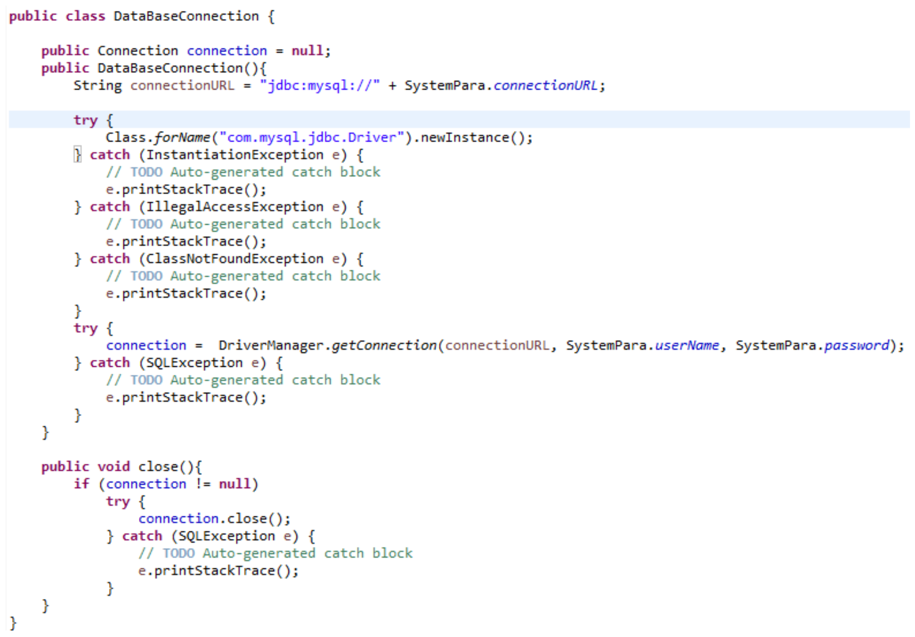
```
jc2.addActionListener(new ActionListener() {

    @Override
    public void actionPerformed(ActionEvent e) {
        // TODO Auto-generated method stub
        if (jc2.getSelectedIndex() != -1){
            // rewrite sql when select a specific item
            int index = jc.getSelectedIndex();
            String sql2 = DEFAULT_QUERYS[index];
            if (jc2.getSelectedIndex() != 0){
                if (index == 1){
                    String[] names = ((String)jc2.getSelectedItem()).split(" ");
                    String fname = names[0];
                    String lname = names[1];
                    if (jc2.getSelectedIndex() != 0) sql2 = sql2 + " and lastName = '" + lname + "' and f
                }
                else  sql2 = sql2 + " and title = '" + (String)jc2.getSelectedItem() + "' order by lastNa
            }
            else{
                if (index == 1) sql2 = "SELECT * FROM titles";
                else if (index == 2) sql2 = "SELECT * FROM authors";
            }
            queryArea.setText(sql2);
            submitButton.doClick();
        }
    }
});
```

Set default selection and the layout of the components



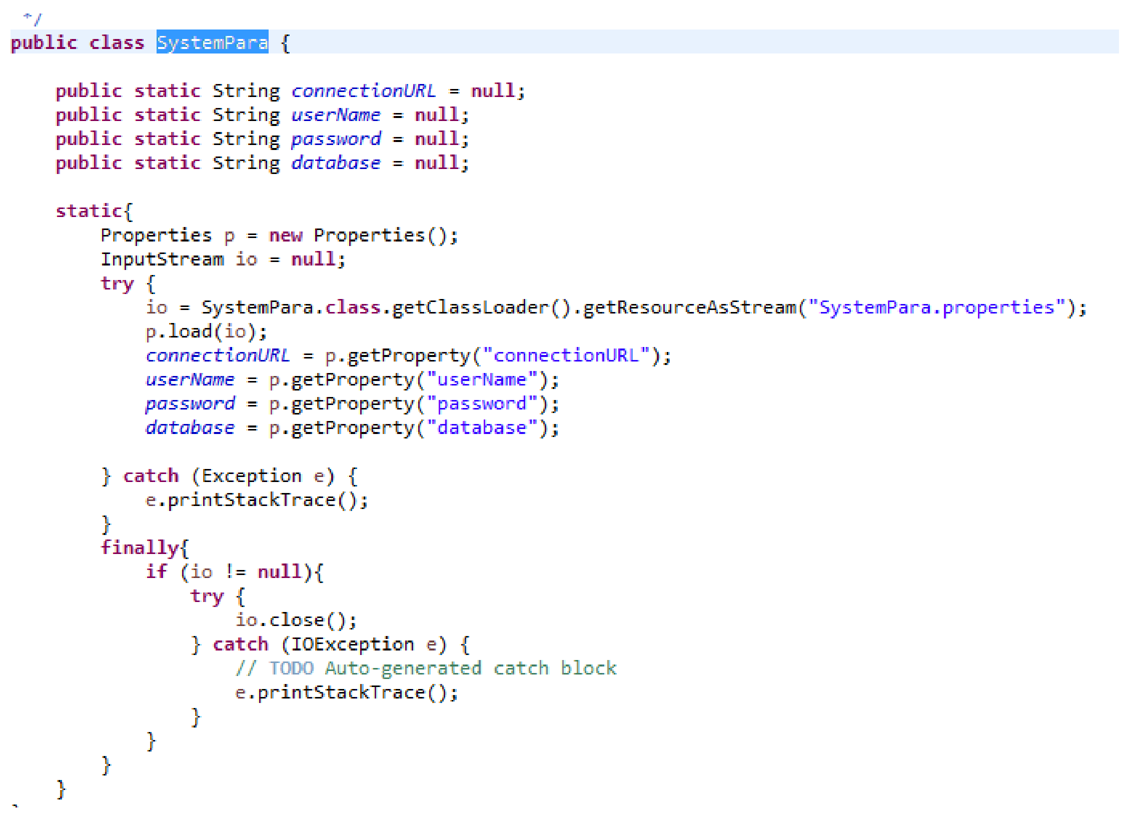
```
// set default selection
jc.setSelectedItem(DEFAULT_QUERYS_NAME[0]);

top = Box.createVerticalBox();
top.add(jc);
top.add(jc2);

boxNorth = Box.createHorizontalBox();
boxNorth.add(scrollPane);
boxNorth.add(submitButton);

add(top, BorderLayout.NORTH);
add(boxNorth, BorderLayout.CENTER);

// create JTable based on the tableModel
resultTable = new JTable(tableModel);
add(new JScrollPane(resultTable), BorderLayout.SOUTH);

this.pack();
this.setVisible(true);
```

f. Database connection: create a database connection



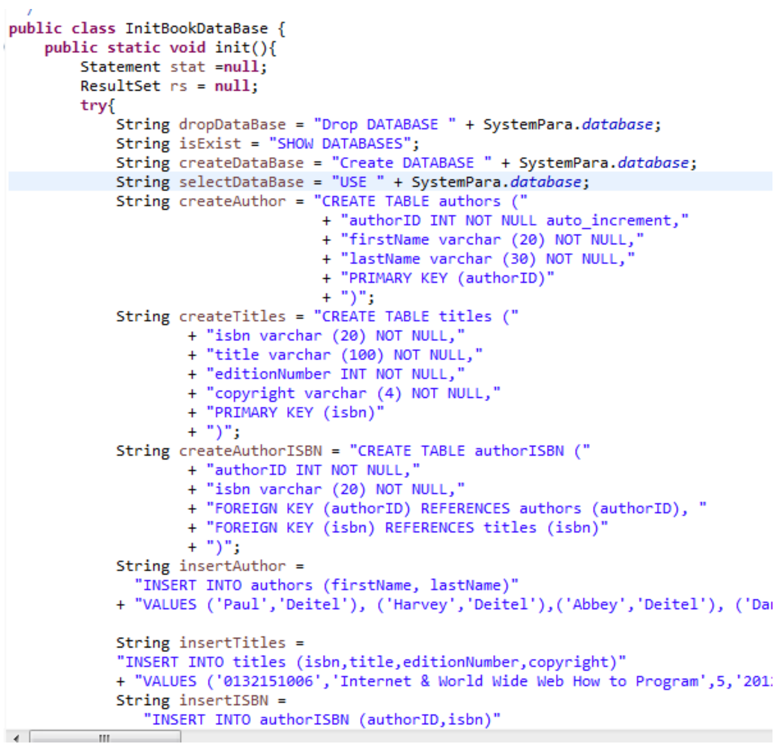
```java
public class DataBaseConnection {

    public Connection connection = null;
    public DataBaseConnection(){
        String connectionURL = "jdbc:mysql://" + SystemPara.connectionURL;

        try {
            Class.forName("com.mysql.jdbc.Driver").newInstance();
        } catch (InstantiationException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IllegalAccessException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (ClassNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        try {
            connection =  DriverManager.getConnection(connectionURL, SystemPara.userName, SystemPara.password);
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

    public void close(){
        if (connection != null)
            try {
                connection.close();
            } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
    }
}
```

g. SystemPara: get system parameter from setting file.

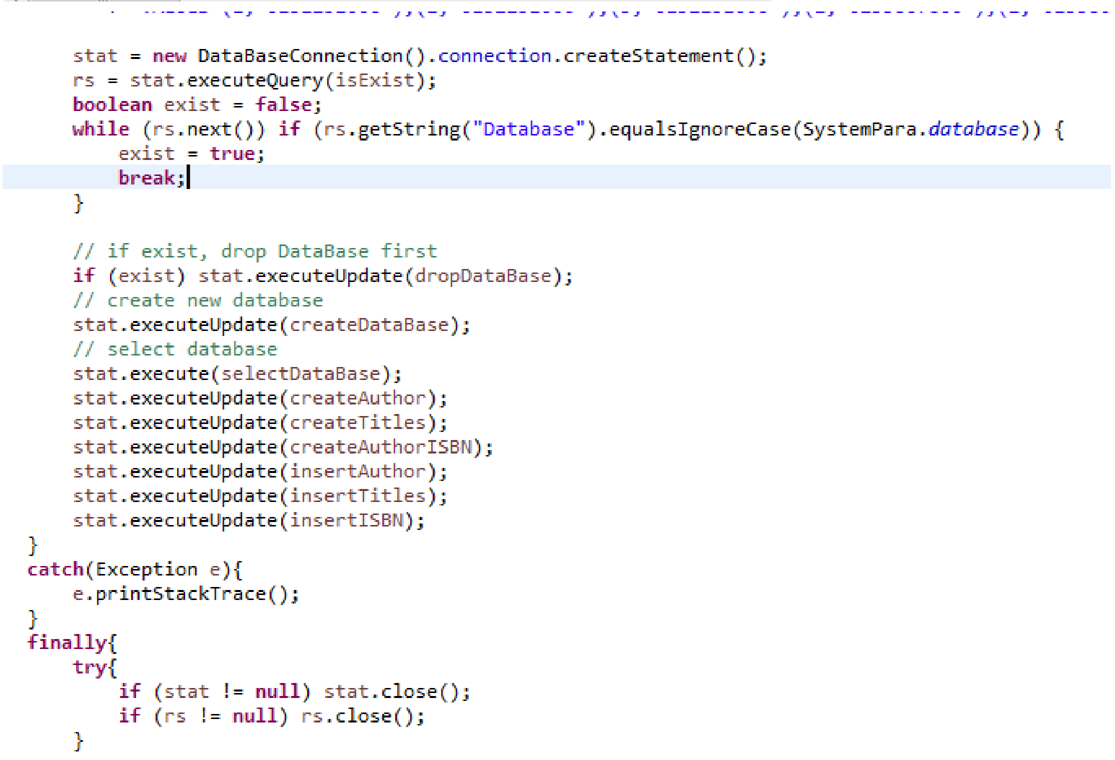


```java
 */
public class SystemPara {

    public static String connectionURL = null;
    public static String userName = null;
    public static String password = null;
    public static String database = null;

    static{
        Properties p = new Properties();
        InputStream io = null;
        try {
            io = SystemPara.class.getClassLoader().getResourceAsStream("SystemPara.properties");
            p.load(io);
            connectionURL = p.getProperty("connectionURL");
            userName = p.getProperty("userName");
            password = p.getProperty("password");
            database = p.getProperty("database");

        } catch (Exception e) {
            e.printStackTrace();
        }
        finally{
            if (io != null){
                try {
                    io.close();
                } catch (IOException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
            }
        }
    }
```

h. 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.



```
public class InitBookDataBase {
    public static void init(){
        Statement stat =null;
        ResultSet rs = null;
        try{
            String dropDataBase = "Drop DATABASE " + SystemPara.database;
            String isExist = "SHOW DATABASES";
            String createDataBase = "Create DATABASE " + SystemPara.database;
            String selectDataBase = "USE " + SystemPara.database;
            String createAuthor = "CREATE TABLE authors ("
                                + "authorID INT NOT NULL auto_increment,"
                                + "firstName varchar (20) NOT NULL,"
                                + "lastName varchar (30) NOT NULL,"
                                + "PRIMARY KEY (authorID)"
                                + ")";
            String createTitles = "CREATE TABLE titles ("
                    + "isbn varchar (20) NOT NULL,"
                    + "title varchar (100) NOT NULL,"
                    + "editionNumber INT NOT NULL,"
                    + "copyright varchar (4) NOT NULL,"
                    + "PRIMARY KEY (isbn)"
                    + ")";
            String createAuthorISBN = "CREATE TABLE authorISBN ("
                    + "authorID INT NOT NULL,"
                    + "isbn varchar (20) NOT NULL,"
                    + "FOREIGN KEY (authorID) REFERENCES authors (authorID), "
                    + "FOREIGN KEY (isbn) REFERENCES titles (isbn)"
                    + ")";
            String insertAuthor =
              "INSERT INTO authors (firstName, lastName)"
            + "VALUES ('Paul','Deitel'), ('Harvey','Deitel'),('Abbey','Deitel'), ('Da

            String insertTitles =
            "INSERT INTO titles (isbn,title,editionNumber,copyright)"
            + "VALUES ('0132151006','Internet & World Wide Web How to Program',5,'201
            String insertISBN =
               "INSERT INTO authorISBN (authorID,isbn)"
```



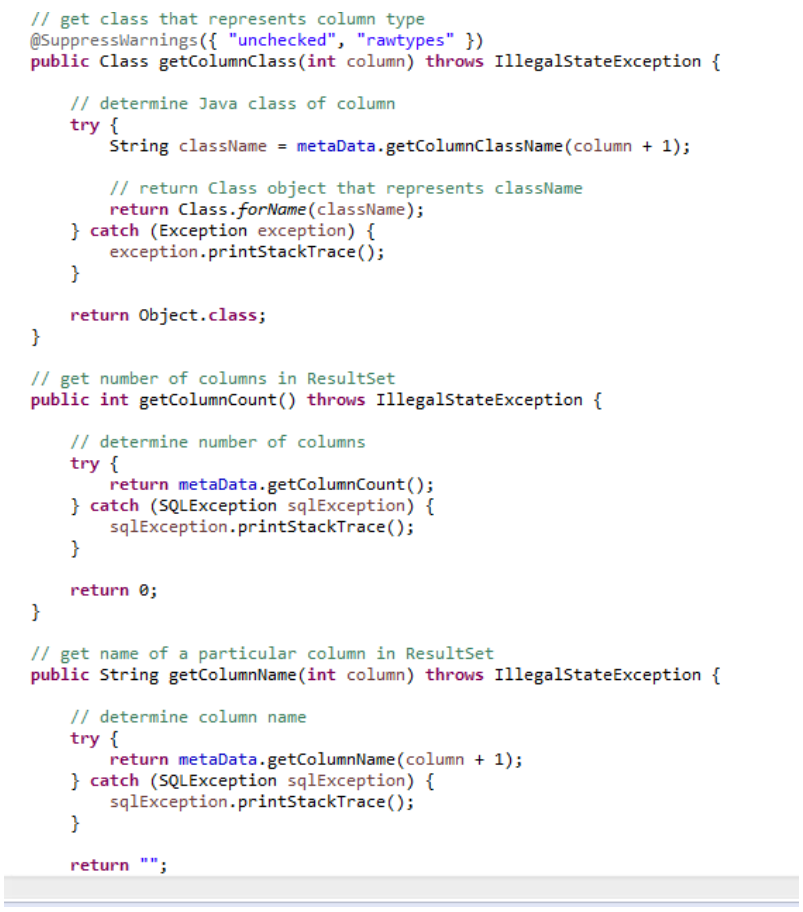
```
            stat = new DataBaseConnection().connection.createStatement();
            rs = stat.executeQuery(isExist);
            boolean exist = false;
            while (rs.next()) if (rs.getString("Database").equalsIgnoreCase(SystemPara.database)) {
                exist = true;
                break;
            }

            // if exist, drop DataBase first
            if (exist) stat.executeUpdate(dropDataBase);
            // create new database
            stat.executeUpdate(createDataBase);
            // select database
            stat.execute(selectDataBase);
            stat.executeUpdate(createAuthor);
            stat.executeUpdate(createTitles);
            stat.executeUpdate(createAuthorISBN);
            stat.executeUpdate(insertAuthor);
            stat.executeUpdate(insertTitles);
            stat.executeUpdate(insertISBN);
        }
        catch(Exception e){
            e.printStackTrace();
        }
        finally{
            try{
                if (stat != null) stat.close();
                if (rs != null) rs.close();
            }
```

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

```
public class ResultSetTableModel extends AbstractTableModel {

    private static final long serialVersionUID = 1L;
    private Connection connection = null;
    private Statement statement = null;
    private ResultSet resultSet = null;
    private ResultSetMetaData metaData;
    private int numberOfRows;

    public ResultSetTableModel(String query) throws SQLException {

        try {
            // connect to database
            connection = new DataBaseConnection().connection;

            // create Statement to query database
            statement = connection.createStatement();

            String selectDataBase = "USE " + SystemPara.database;
            statement.executeQuery(selectDataBase);

            // set query and execute it
            setQuery(query);

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

```
// set new database query string
public void setQuery(String query)
        throws SQLException, IllegalStateException {

    // specify query and execute it
    resultSet = statement.executeQuery(query);

    // obtain meta data for ResultSet
    metaData = resultSet.getMetaData();

    // determine number of rows in ResultSet
    resultSet.last(); // move to last row
    numberOfRows = resultSet.getRow(); // get row number

    // notify JTable that model has changed
    fireTableStructureChanged();
}

public void close() {
    try {
        if (connection != null)
            connection.close();
        if (statement != null)
            statement.close();
        if (resultSet != null)
            resultSet.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

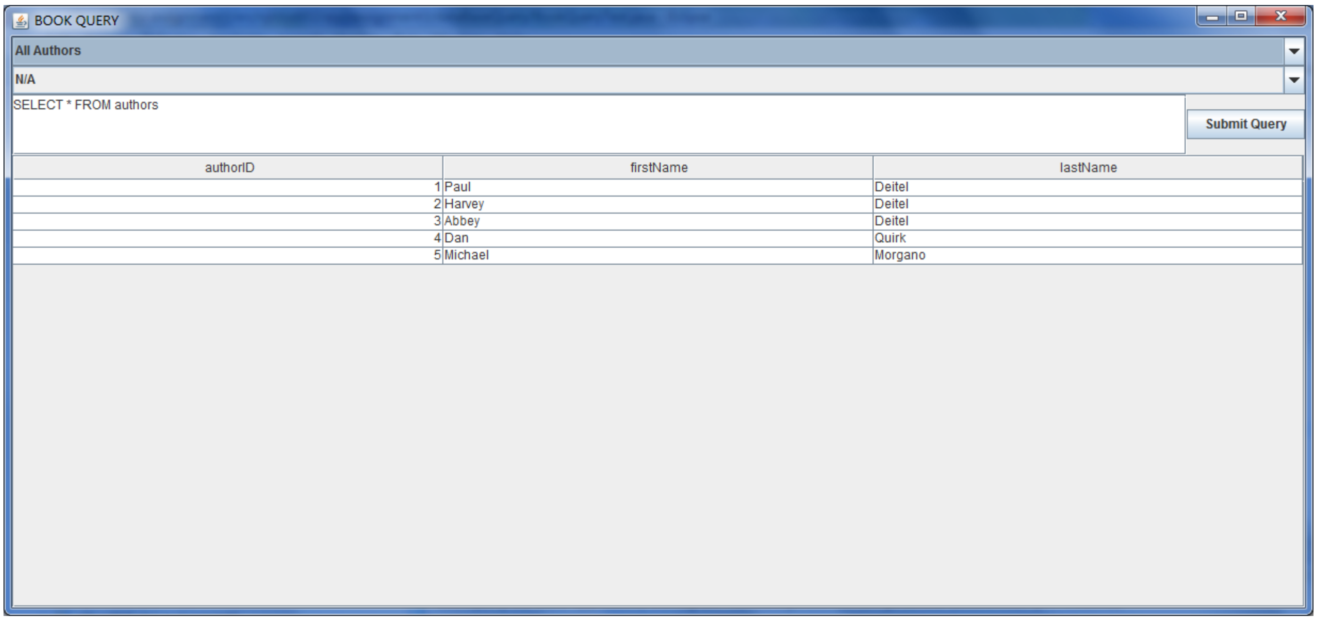
j. Corresponding overridden methods for AbstractTableModel



```java
// get class that represents column type
@SuppressWarnings({ "unchecked", "rawtypes" })
public Class getColumnClass(int column) throws IllegalStateException {

    // determine Java class of column
    try {
        String className = metaData.getColumnClassName(column + 1);

        // return Class object that represents className
        return Class.forName(className);
    } catch (Exception exception) {
        exception.printStackTrace();
    }

    return Object.class;
}

// get number of columns in ResultSet
public int getColumnCount() throws IllegalStateException {

    // determine number of columns
    try {
        return metaData.getColumnCount();
    } catch (SQLException sqlException) {
        sqlException.printStackTrace();
    }

    return 0;
}

// get name of a particular column in ResultSet
public String getColumnName(int column) throws IllegalStateException {

    // determine column name
    try {
        return metaData.getColumnName(column + 1);
    } catch (SQLException sqlException) {
        sqlException.printStackTrace();
    }

    return "";
```

k. Final running result

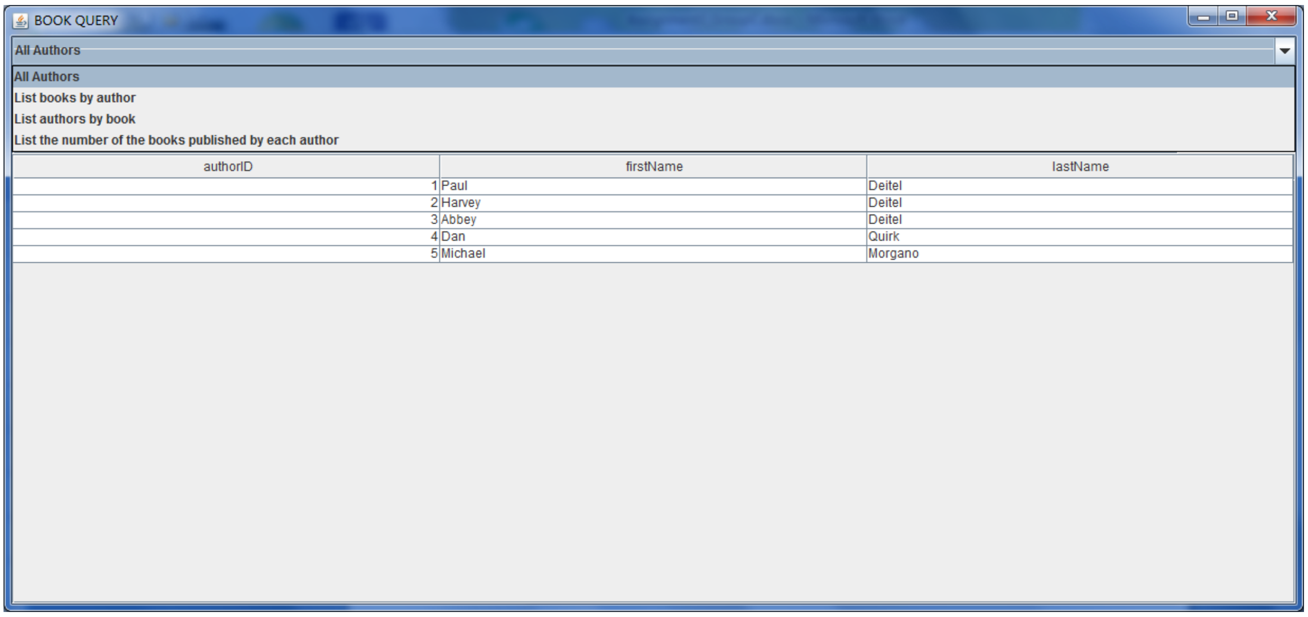
Start: select the first option by default, list all the authors



SELECT * FROM authors

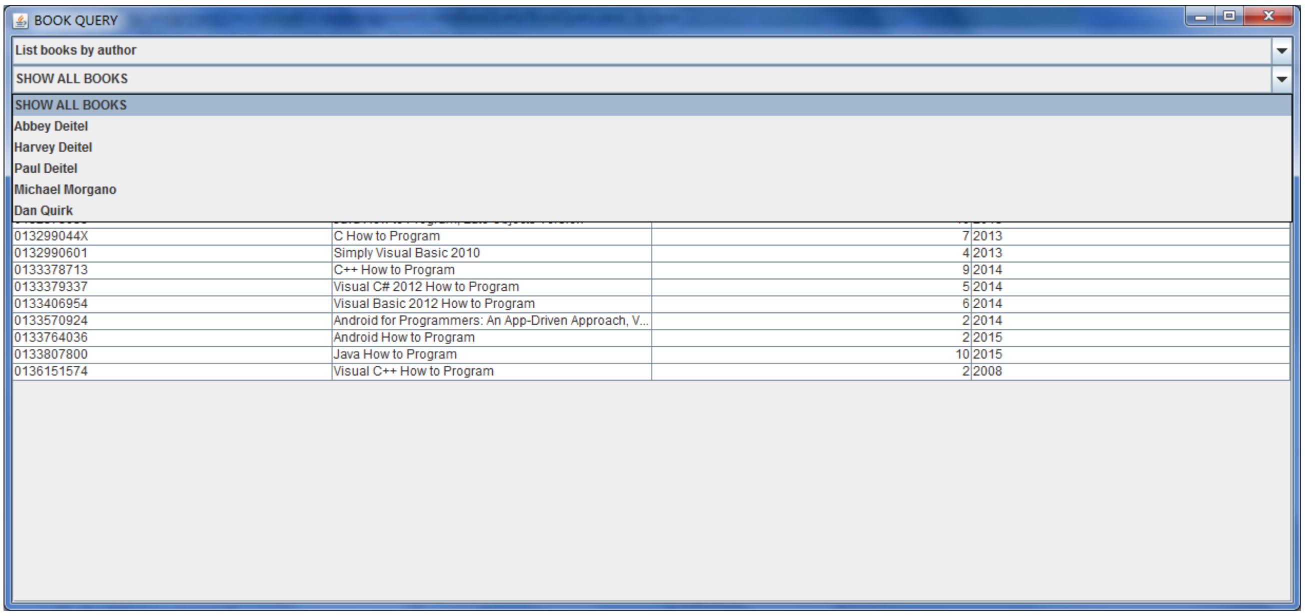
authorID	firstName	lastName
1	Paul	Deitel
2	Harvey	Deitel
3	Abbey	Deitel
4	Dan	Quirk
5	Michael	Morgano

Four predefined queries,



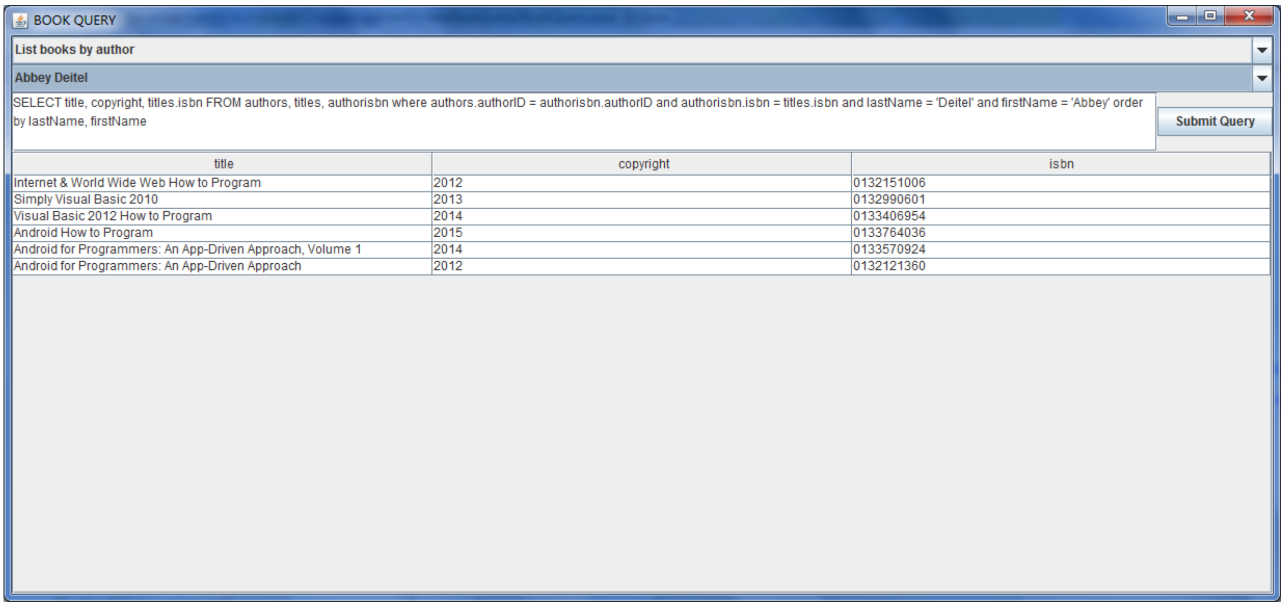
authorID	firstName	lastName
1	Paul	Deitel
2	Harvey	Deitel
3	Abbey	Deitel
4	Dan	Quirk
5	Michael	Morgano

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.



013299044X	C How to Program	7	2013
0132990601	Simply Visual Basic 2010	4	2013
0133378713	C++ How to Program	9	2014
0133379337	Visual C# 2012 How to Program	5	2014
0133406954	Visual Basic 2012 How to Program	6	2014
0133570924	Android for Programmers: An App-Driven Approach, V...	2	2014
0133764036	Android How to Program	2	2015
0133807800	Java How to Program	10	2015
0136151574	Visual C++ How to Program	2	2008

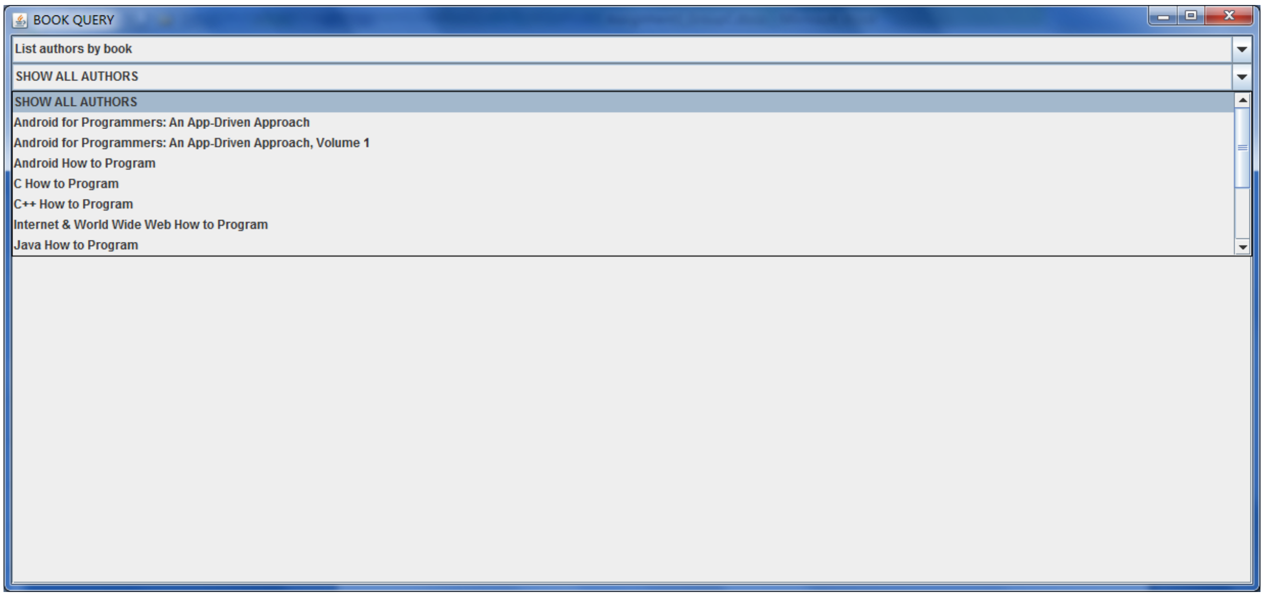
Select an author: list all books for that author. Include each book's title, year and ISBN.



```
SELECT title, copyright, titles.isbn FROM authors, titles, authorisbn where authors.authorID = authorisbn.authorID and authorisbn.isbn = titles.isbn and lastName = 'Deitel' and firstName = 'Abbey' order by lastName, firstName
```

title	copyright	isbn
Internet & World Wide Web How to Program	2012	0132151006
Simply Visual Basic 2010	2013	0132990601
Visual Basic 2012 How to Program	2014	0133406954
Android How to Program	2015	0133764036
Android for Programmers: An App-Driven Approach, Volume 1	2014	0133570924
Android for Programmers: An App-Driven Approach	2012	0132121360

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

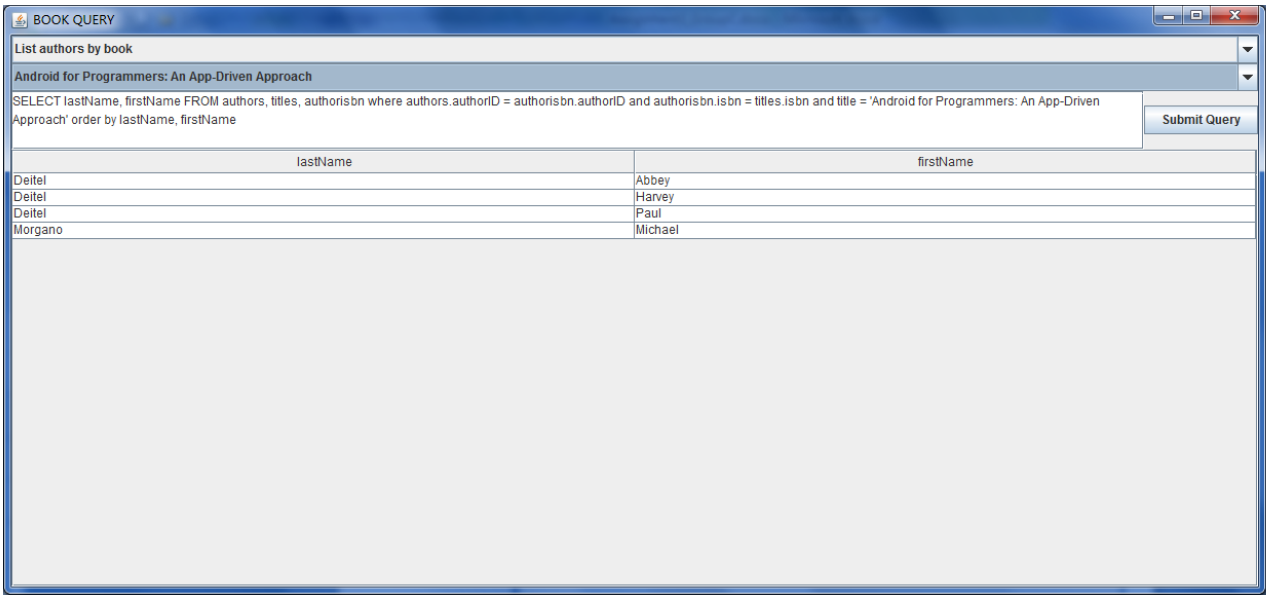


List authors by book

SHOW ALL AUTHORS

```
SHOW ALL AUTHORS
Android for Programmers: An App-Driven Approach
Android for Programmers: An App-Driven Approach, Volume 1
Android How to Program
C How to Program
C++ How to Program
Internet & World Wide Web How to Program
Java How to Program
```

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



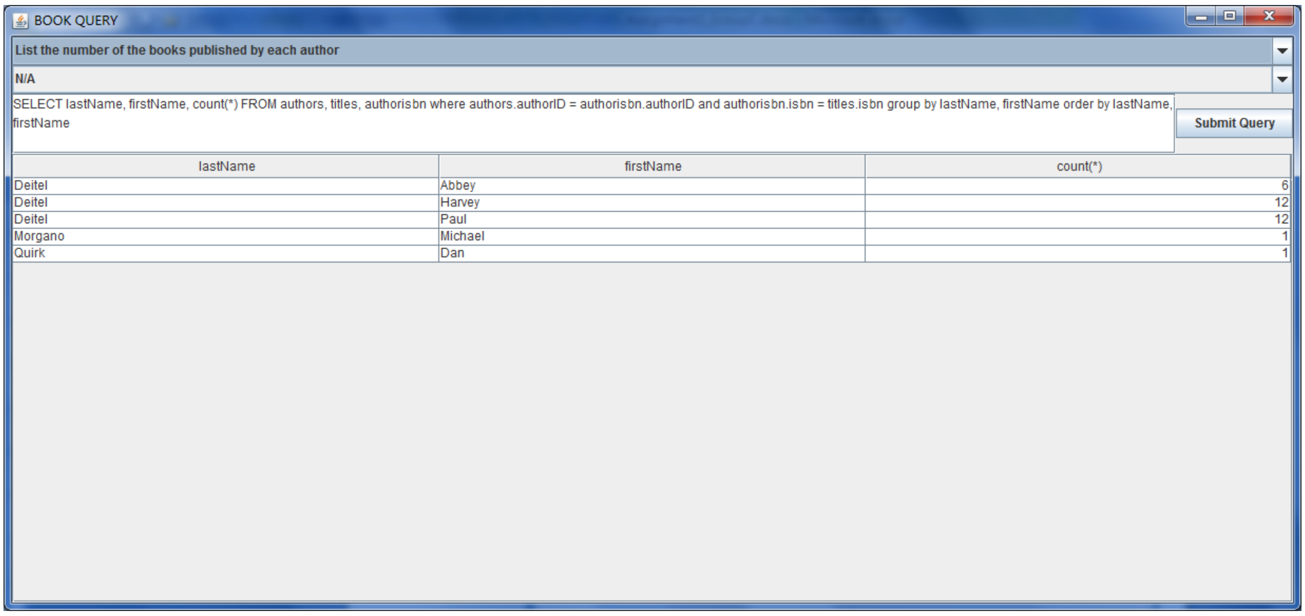
List authors by book

Android for Programmers: An App-Driven Approach

```
SELECT lastName, firstName FROM authors, titles, authorisbn where authors.authorID = authorisbn.authorID and authorisbn.isbn = titles.isbn and title = 'Android for Programmers: An App-Driven Approach' order by lastName, firstName
```

lastName	firstName
Deitel	Abbey
Deitel	Harvey
Deitel	Paul
Morgano	Michael

Select the fourth option:



BOOK QUERY

List the number of the books published by each author

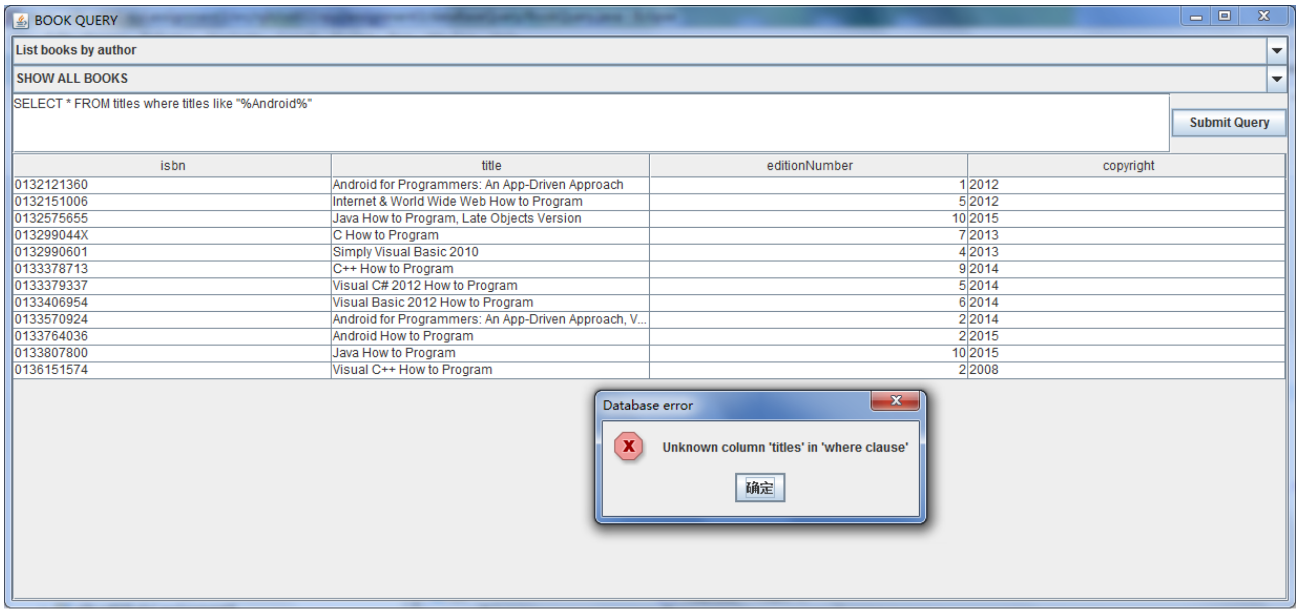
N/A

```
SELECT lastName, firstName, count(*) FROM authors, titles, authorisbn where authors.authorID = authorisbn.authorID and authorisbn.isbn = titles.isbn group by lastName, firstName order by lastName, firstName
```

Submit Query

lastName	firstName	count(*)
Deitel	Abbey	6
Deitel	Harvey	12
Deitel	Paul	12
Morgano	Michael	1
Quirk	Dan	1

Input an invalid customized query:



BOOK QUERY

List books by author

SHOW ALL BOOKS

```
SELECT * FROM titles where titles like "%Android%"
```

Submit Query

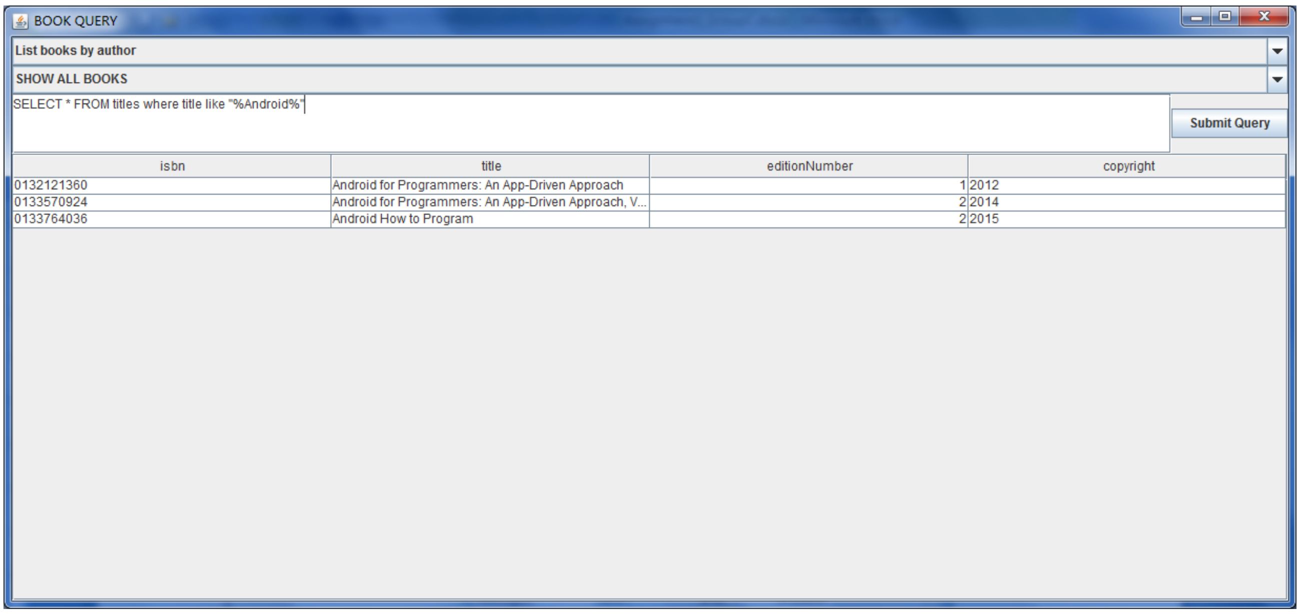
isbn	title	editionNumber	copyright
0132121360	Android for Programmers: An App-Driven Approach	1	2012
0132151006	Internet & World Wide Web How to Program	5	2012
0132575655	Java How to Program, Late Objects Version	10	2015
013299044X	C How to Program	7	2013
0132990601	Simply Visual Basic 2010	4	2013
0133378713	C++ How to Program	9	2014
0133379337	Visual C# 2012 How to Program	5	2014
0133406954	Visual Basic 2012 How to Program	6	2014
0133570924	Android for Programmers: An App-Driven Approach, V...	2	2014
0133764036	Android How to Program	2	2015
0133807800	Java How to Program	10	2015
0136151574	Visual C++ How to Program	2	2008

Database error

Unknown column 'titles' in 'where clause'

确定

Input a valid customized query:



```
SELECT * FROM titles where title like "%Android%"
```

isbn	title	editionNumber	copyright
0132121360	Android for Programmers: An App-Driven Approach	1	2012
0133570924	Android for Programmers: An App-Driven Approach, V...	2	2014
0133764036	Android How to Program	2	2015