Matt Matuk

CSIT – 211

6 March 2015

Lab 4

**Programming Project PP 14.8**

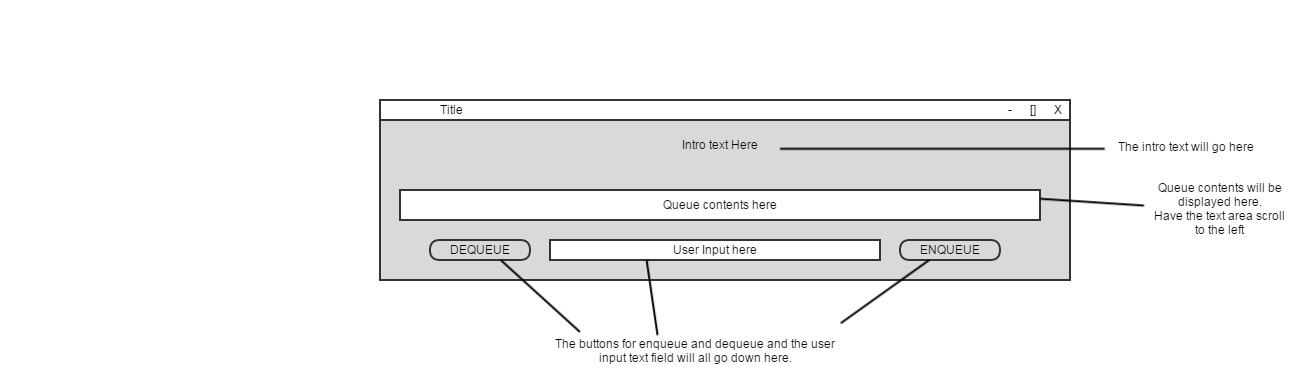
-----**SPEC**-----

Create a graphical application that provides buttons for enqueue and dequeue, a text field to accept a string as input for enqueue, and a text area to display the contents of the queue after each operation.

**SCRUM**

* Class QueueFrame
  + Method main
* Class QueuePanel
  + Variables
    - Button btnEnqueue
    - Button btnDequeue
    - Text Field txtFieldUserInput
    - Text Area txtAreaDisplayQueue
    - String Queue queueStrings
  + Method Constructor

**GUI Storyboard**



**Class: QueueFrame**

1. This class contains the main driver of the program.
2. This class will create a frame which will contain a panel from the QueuePanel class.
3. The application create will use a queue to store user input and display the contents after each operation.

**Import**

**Process**

* Method main
  + Create a new frame called frmQueue
  + Add a new panel from the QueuePanel class to the frame
  + Make sure the frame is set to visible

**Class: QueuePanel**

1. This class will create a panel that will contain two buttons, an intro text, a text area, and a text field.
2. The user will enter a string in the text field and press either button, which will either add or remove a string from the queue.
3. The text area will display the contents of the queue.
4. All methods are public and void unless otherwise stated.
5. All variables are protected unless otherwise stated.
6. The size and color of the panel can all be change to make the application look better. Keep it as close to the GUI as possible.

**Import**

* All swing and awt
* Queue

**Process**

* Variables
  + Button btnEnqueue
  + Button btnDequeue
  + Text Fied txtFieldUserInput
  + Text Area txtAreaQueueDisplay
  + String Queue queueStrings
  + String intro “Welcome to my application. Today you will be entering some strings into a Queue. The buttons Add and Remove will allow you to either add a string to the back of the Queue or remove the string from the front of the Queue.”
* Method Constructor
  + Create the two buttons, text field, text area, and a text label (Display the intro statements).
    - Make sure when creating the display too keep it looking as close as possible to the storyboard.
    - Changes can be made, but only minor.
  + Create a button listener for the two buttons
  + Add everything to the main panel
* Method addToQueue
  + If the txtFieldUserInput is not empty
    - Add the string entered to the queueStrings
    - Call emptyUserInput()
    - Call displayQueue()
* Method removeFromQueue
  + Remove the first item from queueStrings
  + Call displaytQueue()
* Method displayQueue
  + String temp is equal to queueString toString()
  + Remove any ([]) from the string
  + txtAreaQueueDisplay set text to temp
* Method emptyUserInput
  + Clear the txtFieldUserInput
* Private Class ButtonListener implements ActionListtener
  + Method actionPerformed (ActionEvent event)
    - This method accepts the action event, which will be the button pressed
    - If button pressed equals btnEnqueue
      * Call addToQueue()
    - If button pressed equals btnDequeue
      * Call removeFromQueue()

**CODE**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// QueueFrame.java Matt Matuk

// CSIT 211 Project 14.8 pg 578

// 1. This class contains the main driver of the program.

// 2. This class will create a frame which will contain a panel

// from the QueuePanel class.

// 3. The application create will use a queue to store user

// input and display the contents after each operation.

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**import** java.awt.Dimension;

**import** javax.swing.JFrame;

**public** **class** QueueFrame

{

**public** **static** **void** main(String[] args)

{

JFrame frmQueue = **new** JFrame("Queue");

frmQueue.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

frmQueue.setMinimumSize(**new** Dimension(700,225));

frmQueue.getContentPane().add(**new** QueuePanel());

frmQueue.pack();

frmQueue.setLocationRelativeTo(**null**);

frmQueue.setVisible(**true**);

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// QueuePanel.java Matt Matuk

// CSIT 211 Project 14.8 pg 578

// 1. This class will create a panel that will contain two buttons,

// an intro text, a text area, and a text field.

// 2. The user will enter a string in the text field and press

// either button, which will either add or remove a string from

// the queue.

// 3. The text area will display the contents of the queue.

// 4. All methods are public and void unless otherwise stated.

// 5. All variables are protected unless otherwise stated.

// 6. The size and color of the panel can all be change to make

// the application look better. Keep it as close to the GUI as

// possible.

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**import** javax.swing.\*;

**import** java.awt.\*;

**import** java.awt.event.\*;

**import** java.util.LinkedList;

**import** java.util.Queue;

**import** javax.swing.border.TitledBorder;

**public** **class** QueuePanel **extends** JPanel

{

**protected** String intro = "Welcome to my application. Today you will "

+ "be entering some strings into a Queue. The buttons Add and "

+ "Remove will allow you to either add a string to the back of "

+ "the Queue or remove the string from the front of the Queue.";

**protected** JButton btnEnqueue, btnDequeue;

**protected** JTextArea txtAreaQueueDisplay;

**protected** JTextField txtFieldUserInput;

**protected** Queue<String> queueString = **new** LinkedList<String>();

//-------------------------------------------------------------------

// This method create all the panels and layouts for the application.

// This method also sets any listeners and creates the view.

//-------------------------------------------------------------------

**public** QueuePanel() {

setBorder(**new** TitledBorder(UIManager.*getBorder*("TitledBorder.border"),

"Users Queue", TitledBorder.***LEADING***, TitledBorder.***TOP***,

**null**, **new** Color(0, 0, 0)));

SpringLayout springLayout = **new** SpringLayout();

setLayout(springLayout);

JTextArea textArea = **new** JTextArea(intro);

springLayout.putConstraint(SpringLayout.***NORTH***, textArea, 10,

SpringLayout.***NORTH***, **this**);

springLayout.putConstraint(SpringLayout.***WEST***, textArea, 10,

SpringLayout.***WEST***, **this**);

springLayout.putConstraint(SpringLayout.***SOUTH***, textArea, 50,

SpringLayout.***NORTH***, **this**);

springLayout.putConstraint(SpringLayout.***EAST***, textArea, -10,

SpringLayout.***EAST***, **this**);

textArea.setEditable(**false**);

textArea.setBackground(SystemColor.***menu***);

textArea.setLineWrap(**true**);

textArea.setWrapStyleWord(**true**);

add(textArea);

JPanel panel = **new** JPanel();

springLayout.putConstraint(SpringLayout.***NORTH***, panel, -30,

SpringLayout.***SOUTH***, **this**);

springLayout.putConstraint(SpringLayout.***WEST***, panel, 10,

SpringLayout.***WEST***, **this**);

springLayout.putConstraint(SpringLayout.***SOUTH***, panel, -10,

SpringLayout.***SOUTH***, **this**);

springLayout.putConstraint(SpringLayout.***EAST***, panel, -10,

SpringLayout.***EAST***, **this**);

add(panel);

panel.setLayout(**new** BoxLayout(panel, BoxLayout.***X\_AXIS***));

Component horizontalStrut = Box.*createHorizontalStrut*(50);

panel.add(horizontalStrut);

btnDequeue = **new** JButton("DEQUEUE");

panel.add(btnDequeue);

Component horizontalStrut\_1 = Box.*createHorizontalStrut*(20);

panel.add(horizontalStrut\_1);

txtFieldUserInput = **new** JTextField();

panel.add(txtFieldUserInput);

txtFieldUserInput.setColumns(10);

Component horizontalStrut\_2 = Box.*createHorizontalStrut*(20);

panel.add(horizontalStrut\_2);

btnEnqueue = **new** JButton("ENQUEUE");

panel.add(btnEnqueue);

Component horizontalStrut\_3 = Box.*createHorizontalStrut*(50);

panel.add(horizontalStrut\_3);

JLabel lblUserInput = **new** JLabel("Please enter a string to add "

+ "to the Queue");

springLayout.putConstraint(SpringLayout.***SOUTH***, lblUserInput, 0,

SpringLayout.***NORTH***, panel);

lblUserInput.setHorizontalAlignment(SwingConstants.***CENTER***);

springLayout.putConstraint(SpringLayout.***WEST***, lblUserInput, 0,

SpringLayout.***WEST***, textArea);

add(lblUserInput);

JLabel lblQueueDisplay = **new** JLabel("Queue Contents");

add(lblQueueDisplay);

JPanel panelTextArea = **new** JPanel();

springLayout.putConstraint(SpringLayout.***WEST***, lblQueueDisplay,

0, SpringLayout.***WEST***, panelTextArea);

springLayout.putConstraint(SpringLayout.***SOUTH***, lblQueueDisplay,

0, SpringLayout.***NORTH***, panelTextArea);

springLayout.putConstraint(SpringLayout.***NORTH***, panelTextArea,

20, SpringLayout.***SOUTH***, textArea);

springLayout.putConstraint(SpringLayout.***WEST***, panelTextArea,

10, SpringLayout.***WEST***, **this**);

springLayout.putConstraint(SpringLayout.***SOUTH***, panelTextArea,

-50, SpringLayout.***SOUTH***, **this**);

springLayout.putConstraint(SpringLayout.***EAST***, panelTextArea,

-10, SpringLayout.***EAST***, **this**);

add(panelTextArea);

SpringLayout sl\_panelTextArea = **new** SpringLayout();

panelTextArea.setLayout(sl\_panelTextArea);

txtAreaQueueDisplay = **new** JTextArea();

JScrollPane scroll = **new** JScrollPane(txtAreaQueueDisplay);

sl\_panelTextArea.putConstraint(SpringLayout.***NORTH***, scroll, 0,

SpringLayout.***NORTH***, panelTextArea);

sl\_panelTextArea.putConstraint(SpringLayout.***WEST***, scroll, 0,

SpringLayout.***WEST***, panelTextArea);

sl\_panelTextArea.putConstraint(SpringLayout.***SOUTH***, scroll, 0,

SpringLayout.***SOUTH***, panelTextArea);

sl\_panelTextArea.putConstraint(SpringLayout.***EAST***, scroll, 0,

SpringLayout.***EAST***, panelTextArea);

panelTextArea.add(scroll);

txtAreaQueueDisplay.setEditable(**false**);

ButtonListener listener = **new** ButtonListener();

btnEnqueue.addActionListener(listener);

btnDequeue.addActionListener(listener);

}

//-------------------------------------------------------------------

// This method adds the users input to the queue only if the

// user typed some text.

// Then the method clears the queue and displays the new queue

// contents.

//-------------------------------------------------------------------

**public** **void** addToQueue()

{

**if** (!txtFieldUserInput.getText().equalsIgnoreCase(""))

{

queueString.add(txtFieldUserInput.getText());

emptyUserInput();

displayQueue();

}

}

//-------------------------------------------------------------------

// This method removes the first item in the queue only if the

// queue is not empty.

// Then the method displays the new queue contents.

//-------------------------------------------------------------------

**public** **void** removeFromQueue()

{

**if** (queueString.isEmpty() == **false**)

{

queueString.remove();

displayQueue();

}

}

//-------------------------------------------------------------------

// This method removes the [ and ] from the front and back of the

// toString() method when calling it on the queue. This method also

// sets the text area contents to display the contents of the queue.

//-------------------------------------------------------------------

**public** **void** displayQueue()

{

String temp = queueString.toString().substring(1,

queueString.toString().length() -1);

txtAreaQueueDisplay.setText(temp);

}

//-------------------------------------------------------------------

// Clears the contents of the user input text field

//-------------------------------------------------------------------

**public** **void** emptyUserInput()

{

txtFieldUserInput.setText("");

}

//-------------------------------------------------------------------

// Creates the button listener for each button, then call the

// propper method when the button is pressed.

//-------------------------------------------------------------------

**private** **class** ButtonListener **implements** ActionListener

{

**public** **void** actionPerformed(ActionEvent event)

{

**if** (event.getSource() == btnEnqueue)

{

addToQueue();

}

**if** (event.getSource() == btnDequeue)

{

removeFromQueue();

}

}

}

//-------------------------------------------------------------------

// Setter and Getter for teh intro text.

//-------------------------------------------------------------------

**public** **void** setIntro(String str)

{

intro = str;

}

**public** String getIntro()

{

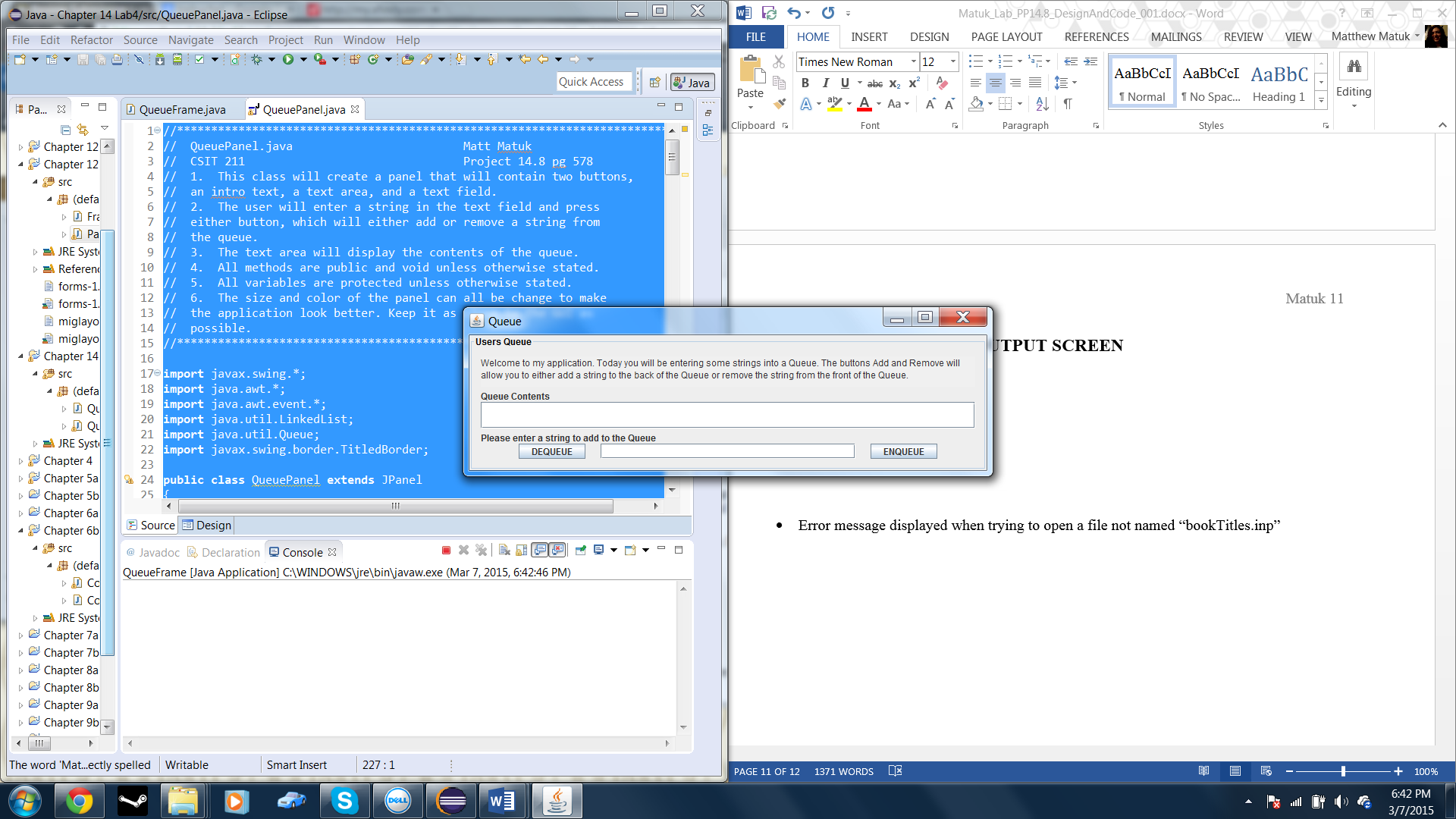
**return** intro;

}

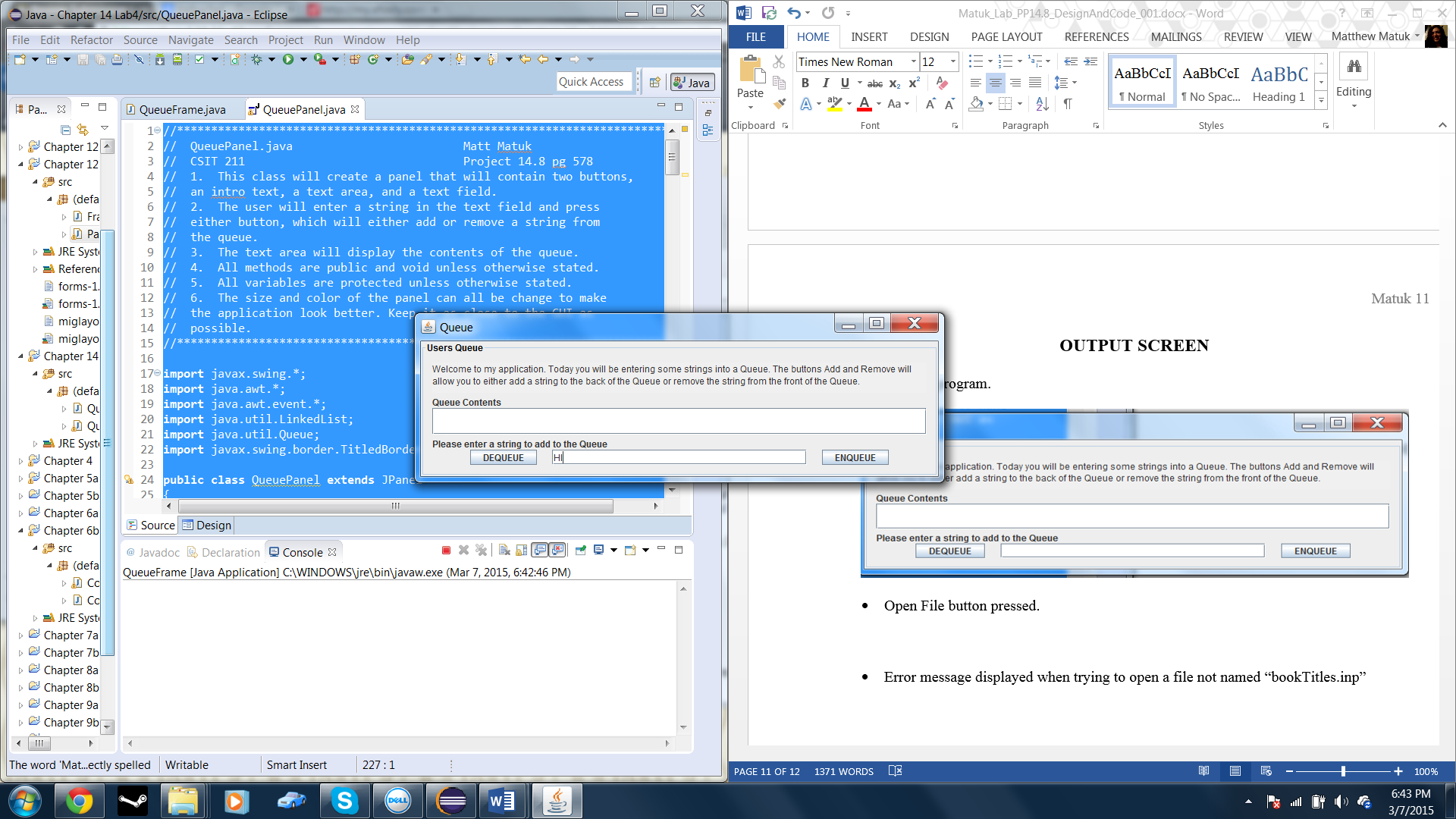
}

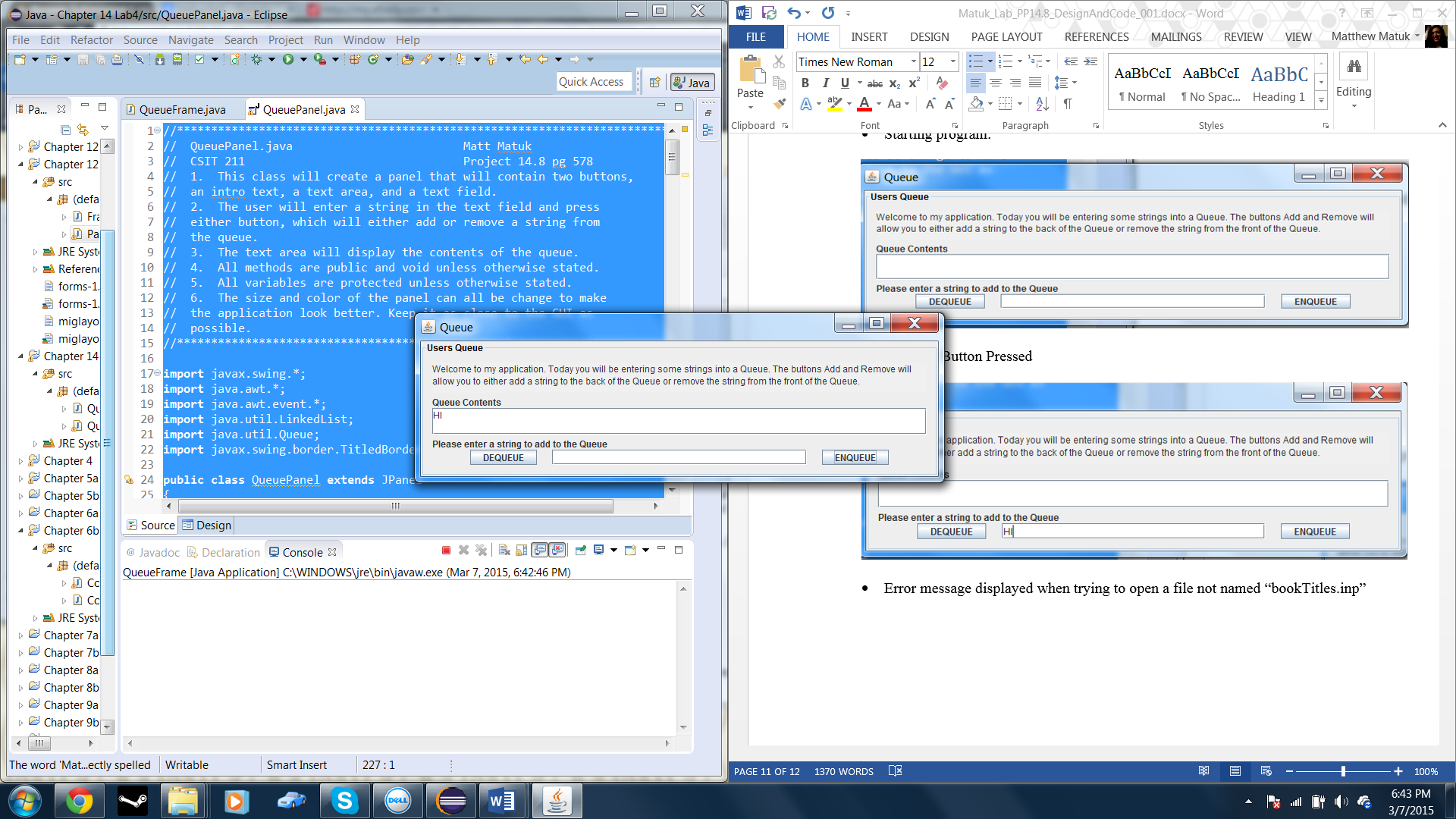
**OUTPUT SCREEN**

* Starting program.

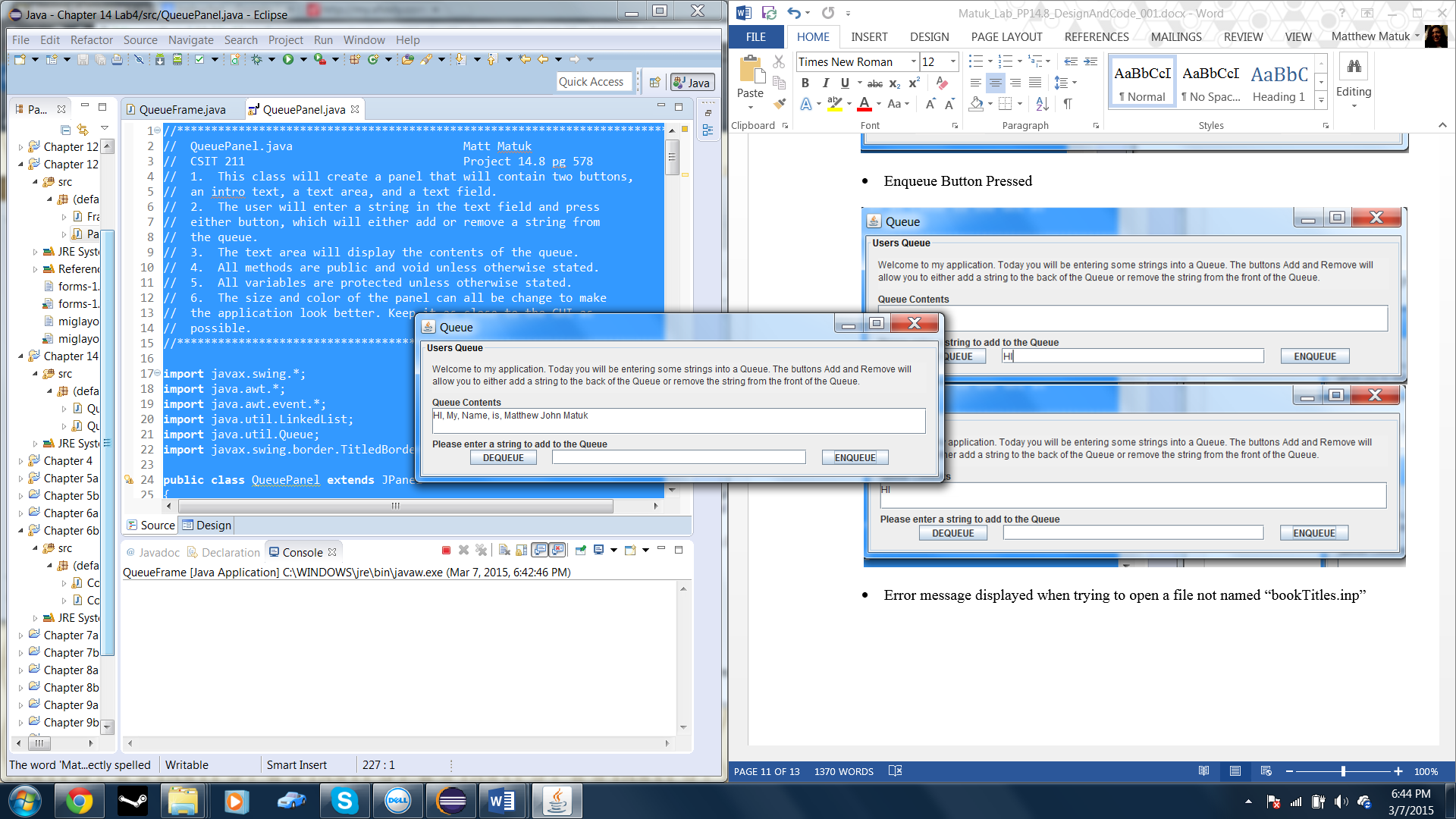


* Enqueue Button Pressed





* A couple more Enqueue buttons pressed



* Dequeue pressed four times.

