Matt Matuk

CSIT – 211

28 February 2015

Lab 2

**Programming Project PP 12.9**

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

Create a graphical application that provides a button for push and pop from a stack, a text field to accept a string as input for push, and a text area to display the contents of the stack after each operation.

**SCRUM**

* Class PanelStack
  + GUI
    - Button btnPop
    - Button btnPush
    - Text Area txtAreaStack
    - Text Field txtFieldUserInput
  + Variable
    - Stack string userStack
  + Method constructor
  + Method popStack
  + Method pushStack
* Class FrameStack
  + Method main

**Class: PanelStack**

1. This class will contain the method that will create all the components of the panel for the frame
2. Make sure when creating the window, that it looks similar to the GUI storyboard. Color and exact size can change.
3. Multiple methods will be used to add and remove strings from the stack
4. All methods are public and void unless otherwise stated

**Import**

* Any imports need to create the window to make it look like the GUI storyboard

**Process**

* Variables
  + Stack<String> userStack
  + Text Field txtFieldUserInput
* Method main
* Method initialize()
  + Create the window in this method
  + Make sure there are two buttons (btnPush and btnPop) with each button adding or removing a string from the stack.
  + Make sure there is a text area (txtArea) to display the contents of the stack userStack
  + A text field (txtField) for the user to enter a string
  + Add Button listeners two each of the buttons
* Class ButtonListener implements ActionListener
  + Method actionPerformed(ActionEvent event)
    - If event equals btnPop then call the method popStack()
    - If event equals btnPush then call the method pushStack()
  + Method btnPush()
    - Add the value of the txtFieldUserInput to the stack userStack
    - Set the txtAreaStack text equal to the contents of the stack userStack
  + Method btnPop()
    - If userStack is not empty
      * Remove the top value of the stack userStack
    - If userStack is empty
      * Set the text value of txtAreaStack to “EMPTY:”
    - Else
      * Set the text value of txtAreaStack to the contents of userStack

**Class: FrameStack**

1. This class creates a new DisplayStack and frame for the application
2. This class is the main driver class

**Import**

**Process**

* Method main
  + Create a new DisplayStack and frame and set the frame to visable

**CODE**

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

// FrameStack.java Matt Matuk

// CSIT 211 Project 12.9 pg 516

// 1. This class creates a new DisplayStack and frame

// for the application

// 2. This class is the main driver class

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

**import** java.awt.Dimension;

**import** javax.swing.JFrame;

**public** **class** FrameStack

{

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

{

JFrame frmStack = **new** JFrame("Stack");

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

frmStack.setMinimumSize(**new** Dimension(450,300));

frmStack.setBounds(100, 100, 450, 300);

frmStack.getContentPane().add(**new** PanelStack());

frmStack.pack();

frmStack.setVisible(**true**);

}

}

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

// PanelStack.java Matt Matuk

// CSIT 211 Project 12.9 pg 516

// 1. This class will contain the method that will create all

// the components of the panel for the frame

// 2. Make sure when creating the window, that it looks similar

// to the GUI storyboard. Color and exact size can change.

// 3. Multiple methods will be used to add and remove strings

// from the stack

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

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

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

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

**import** javax.swing.text.BadLocationException;

**import** javax.swing.text.Utilities;

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

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

**import** java.util.Stack;

**public** **class** PanelStack **extends** JPanel {

**private** JTextField txtFieldUserInput;

**protected** JButton btnPop, btnPush;

**protected** Stack<String> userStack = **new** Stack<String>();

**protected** JTextArea txtAreaStack;

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

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

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

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

**public** PanelStack() {

setBorder(**new** TitledBorder(**null**, "Stack", TitledBorder.***LEADING***, TitledBorder.***TOP***, **null**, **null**));

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

setLayout(springLayout);

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

springLayout.putConstraint(SpringLayout.***NORTH***, panel, 0, SpringLayout.***NORTH***, **this**);

springLayout.putConstraint(SpringLayout.***WEST***, panel, 0, SpringLayout.***WEST***, **this**);

springLayout.putConstraint(SpringLayout.***SOUTH***, panel, -30, SpringLayout.***SOUTH***, **this**);

springLayout.putConstraint(SpringLayout.***EAST***, panel, 0, SpringLayout.***EAST***, **this**);

add(panel);

JPanel panel\_1 = **new** JPanel();

springLayout.putConstraint(SpringLayout.***NORTH***, panel\_1, 0, SpringLayout.***SOUTH***, panel);

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

panel.setLayout(sl\_panel);

JPanel panel\_2 = **new** JPanel();

sl\_panel.putConstraint(SpringLayout.***NORTH***, panel\_2, 10, SpringLayout.***NORTH***, panel);

sl\_panel.putConstraint(SpringLayout.***WEST***, panel\_2, 10, SpringLayout.***WEST***, panel);

sl\_panel.putConstraint(SpringLayout.***SOUTH***, panel\_2, -22, SpringLayout.***SOUTH***, panel);

panel.add(panel\_2);

JLabel lblUserString = **new** JLabel("Please enter your string here.");

sl\_panel.putConstraint(SpringLayout.***NORTH***, lblUserString, 6, SpringLayout.***SOUTH***, panel\_2);

sl\_panel.putConstraint(SpringLayout.***WEST***, lblUserString, 106, SpringLayout.***WEST***, panel);

sl\_panel.putConstraint(SpringLayout.***EAST***, lblUserString, -95, SpringLayout.***EAST***, panel);

panel.add(lblUserString);

JTextArea txtAreaIntro = **new** JTextArea();

sl\_panel.putConstraint(SpringLayout.***NORTH***, txtAreaIntro, 10, SpringLayout.***NORTH***, panel);

sl\_panel.putConstraint(SpringLayout.***SOUTH***, txtAreaIntro, -6, SpringLayout.***NORTH***, lblUserString);

sl\_panel.putConstraint(SpringLayout.***EAST***, panel\_2, -6, SpringLayout.***WEST***, txtAreaIntro);

sl\_panel.putConstraint(SpringLayout.***WEST***, txtAreaIntro, 227, SpringLayout.***WEST***, panel);

txtAreaIntro.setLineWrap(**true**);

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

sl\_panel.putConstraint(SpringLayout.***EAST***, txtAreaIntro, -10, SpringLayout.***EAST***, panel);

txtAreaIntro.setWrapStyleWord(**true**);

txtAreaIntro.setEditable(**false**);

txtAreaIntro.setText("Welcome to my application. Today you will be "

+ "entering a string at that bottom of the page and then "

+ "press PUSH or POP to either add or remove the string from "

+ "your stack. The contents of your stack will be displayed to the left.");

panel.add(txtAreaIntro);

panel\_2.setLayout(**new** BoxLayout(panel\_2, BoxLayout.***Y\_AXIS***));

JLabel lblStack = **new** JLabel("Here is the contents of your stack:");

lblStack.setAlignmentX(Component.***CENTER\_ALIGNMENT***);

lblStack.setAlignmentY(Component.***TOP\_ALIGNMENT***);

panel\_2.add(lblStack);

sl\_panel.putConstraint(SpringLayout.***NORTH***, lblStack, 0, SpringLayout.***NORTH***, panel\_2);

sl\_panel.putConstraint(SpringLayout.***WEST***, lblStack, 30, SpringLayout.***EAST***, panel\_2);

Component verticalStrut = Box.*createVerticalStrut*(5);

panel\_2.add(verticalStrut);

txtAreaStack = **new** JTextArea();

JScrollPane scrollPane = **new** JScrollPane(txtAreaStack);

panel\_2.add(scrollPane);

sl\_panel.putConstraint(SpringLayout.***WEST***, txtAreaStack, 10, SpringLayout.***WEST***, panel);

sl\_panel.putConstraint(SpringLayout.***SOUTH***, txtAreaStack, -6, SpringLayout.***NORTH***, lblUserString);

txtAreaStack.setWrapStyleWord(**true**);

txtAreaStack.setEditable(**false**);

sl\_panel.putConstraint(SpringLayout.***EAST***, txtAreaStack, -6, SpringLayout.***WEST***, txtAreaIntro);

springLayout.putConstraint(SpringLayout.***WEST***, panel\_1, 0, SpringLayout.***WEST***, **this**);

springLayout.putConstraint(SpringLayout.***SOUTH***, panel\_1, -10, SpringLayout.***SOUTH***, **this**);

springLayout.putConstraint(SpringLayout.***EAST***, panel\_1, 0, SpringLayout.***EAST***, **this**);

add(panel\_1);

panel\_1.setLayout(**new** BoxLayout(panel\_1, BoxLayout.***X\_AXIS***));

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

panel\_1.add(horizontalStrut);

btnPush = **new** JButton("PUSH");

panel\_1.add(btnPush);

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

panel\_1.add(horizontalStrut\_1);

txtFieldUserInput = **new** JTextField();

panel\_1.add(txtFieldUserInput);

txtFieldUserInput.setColumns(10);

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

panel\_1.add(horizontalStrut\_2);

btnPop = **new** JButton("POP");

panel\_1.add(btnPop);

Component horizontalStrut\_3 = Box.*createHorizontalStrut*(20);

panel\_1.add(horizontalStrut\_3);

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

btnPop.addActionListener(listener);

btnPush.addActionListener(listener);

}

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

{

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

// This method handles the button clickes

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

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

{

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

{

btnPop();

}

**else**

{

btnPush();

}

// returns teh text area back to the top after each action

txtAreaStack.setCaretPosition(0);

}

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

// When the user pushes the PUSH button, the text entered in the

// text field is placed at the top of the text area txtAreaStack

// using the top string in the stack.

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

**private** **void** btnPush()

{

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

{

**if** (txtAreaStack.getText().equalsIgnoreCase("empty:"))

{

txtAreaStack.setText("");

}

userStack.push(txtFieldUserInput.getText());

txtFieldUserInput.setText("");

**try**

{

// inserts the string entered at the top of all the text in

// txtAreaStack

txtAreaStack.getDocument().insertString(0, userStack.peek()+"\n", **null**);

}

**catch** (BadLocationException e)

{

e.printStackTrace();

}

}

}

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

// When the user presses the POP button, the the top string in the

// stack is removed and then the top line in the txtAreaStack is removed

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

**private** **void** btnPop()

{

**if** (userStack.empty() != **true**)

{

userStack.pop();

**try**

{

txtAreaStack.getDocument().remove(0,

Utilities.*getRowEnd*(txtAreaStack, 0)+1);

}

**catch** (BadLocationException e)

{

e.printStackTrace();

}

}

// sets Text area to empty if stack is emtpy

**if** (userStack.empty())

{

txtAreaStack.setText("EMPTY:");

}

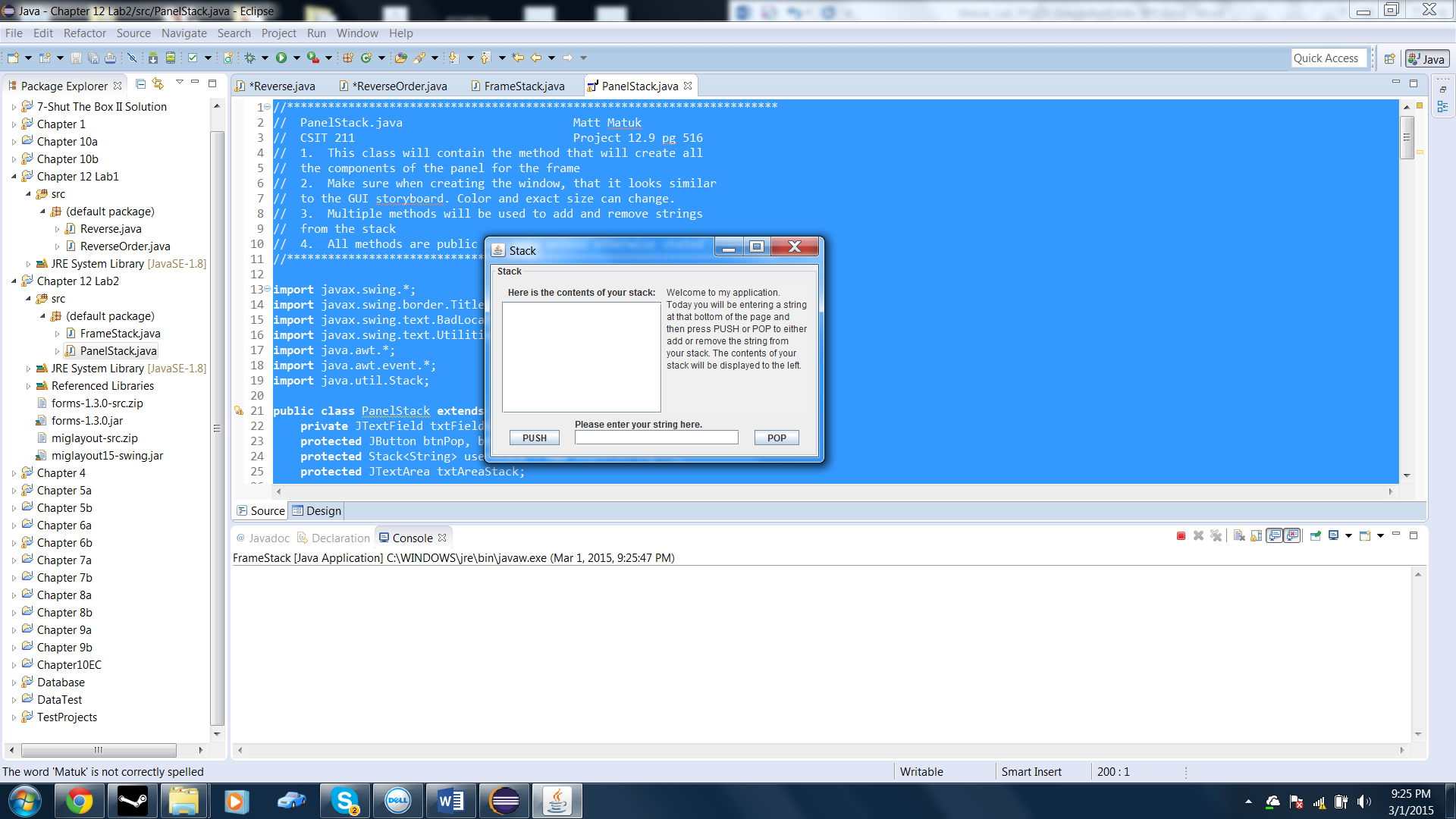
}

}

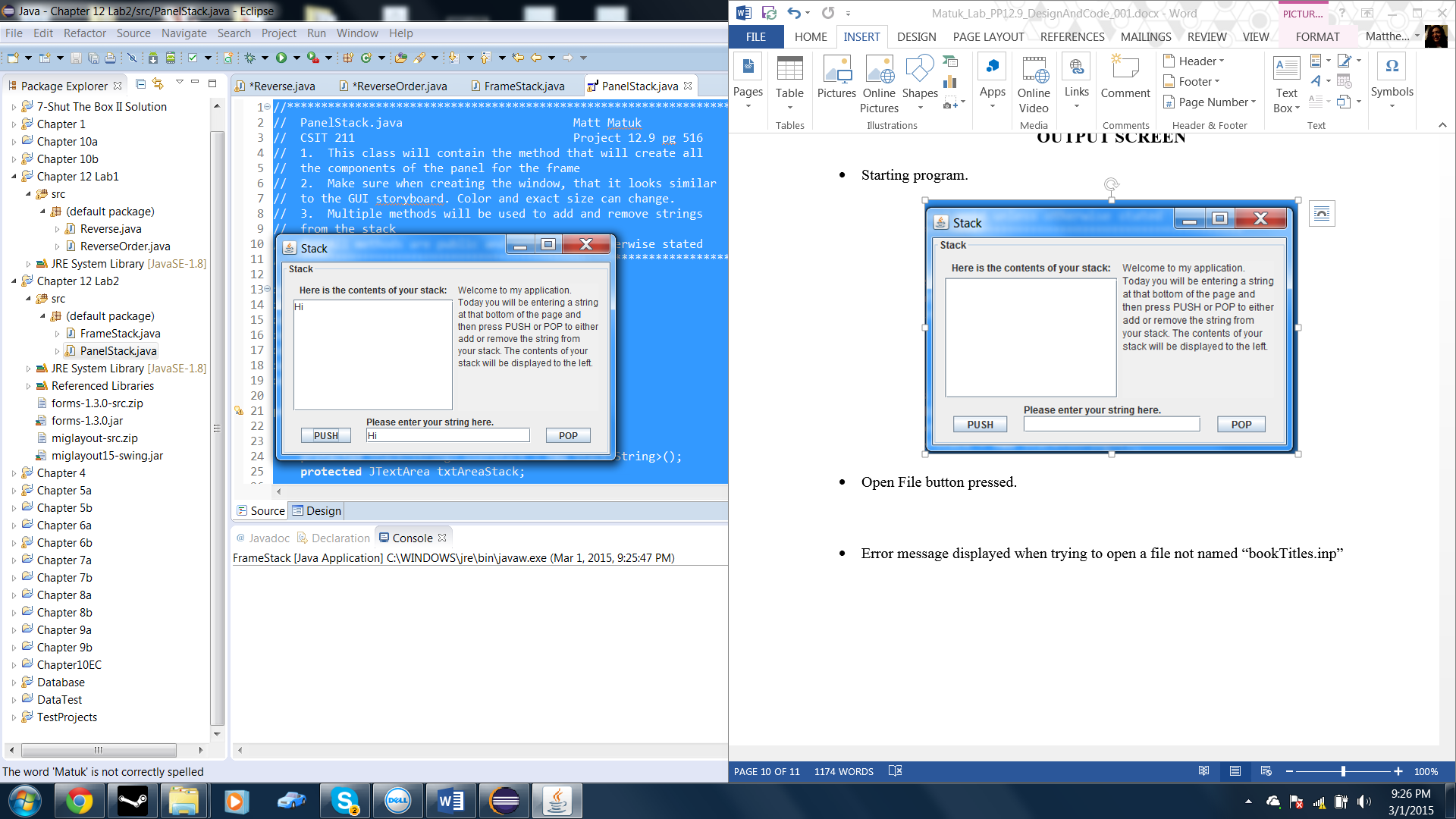
}

**OUTPUT SCREEN**

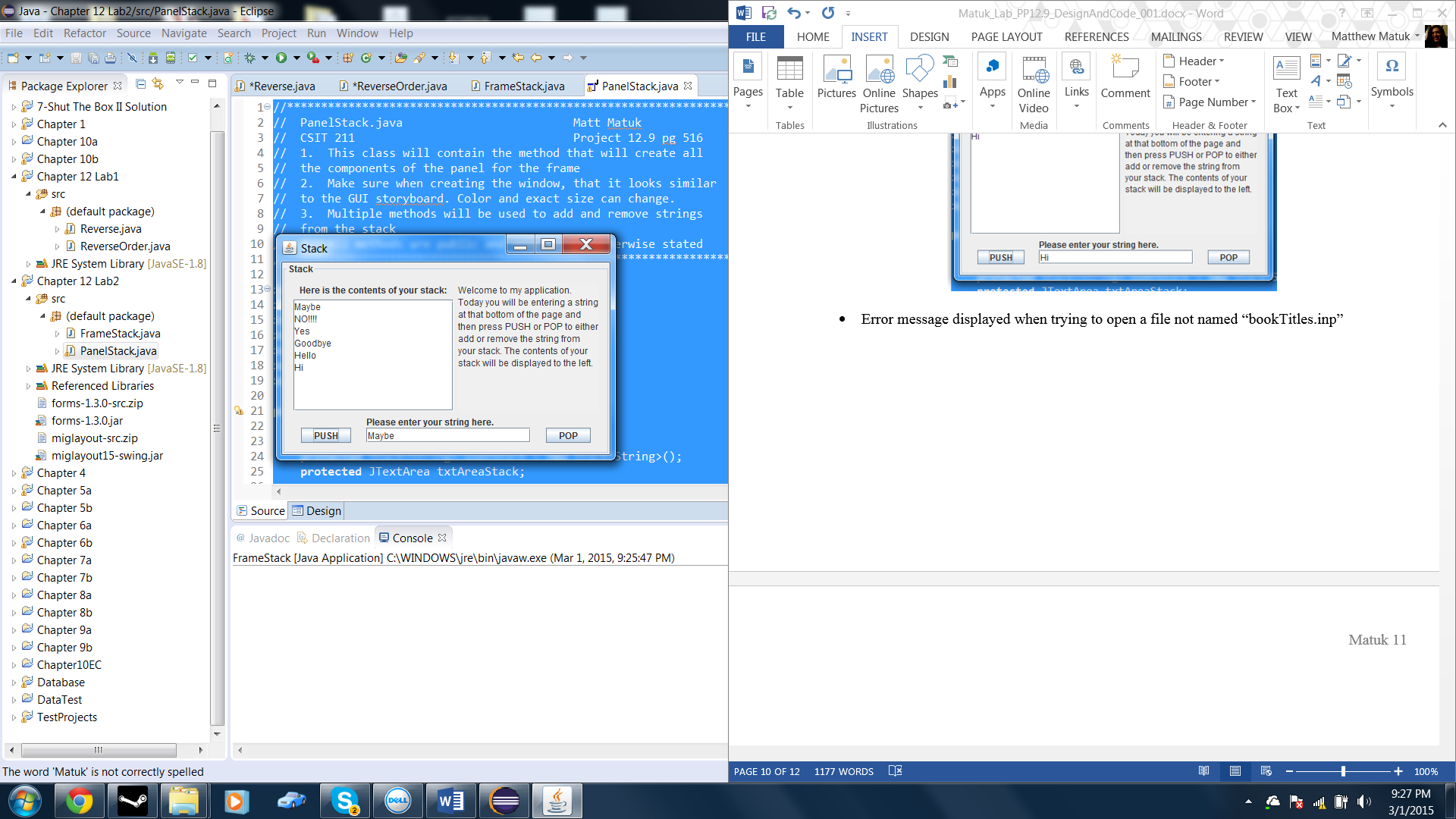
* Starting program.



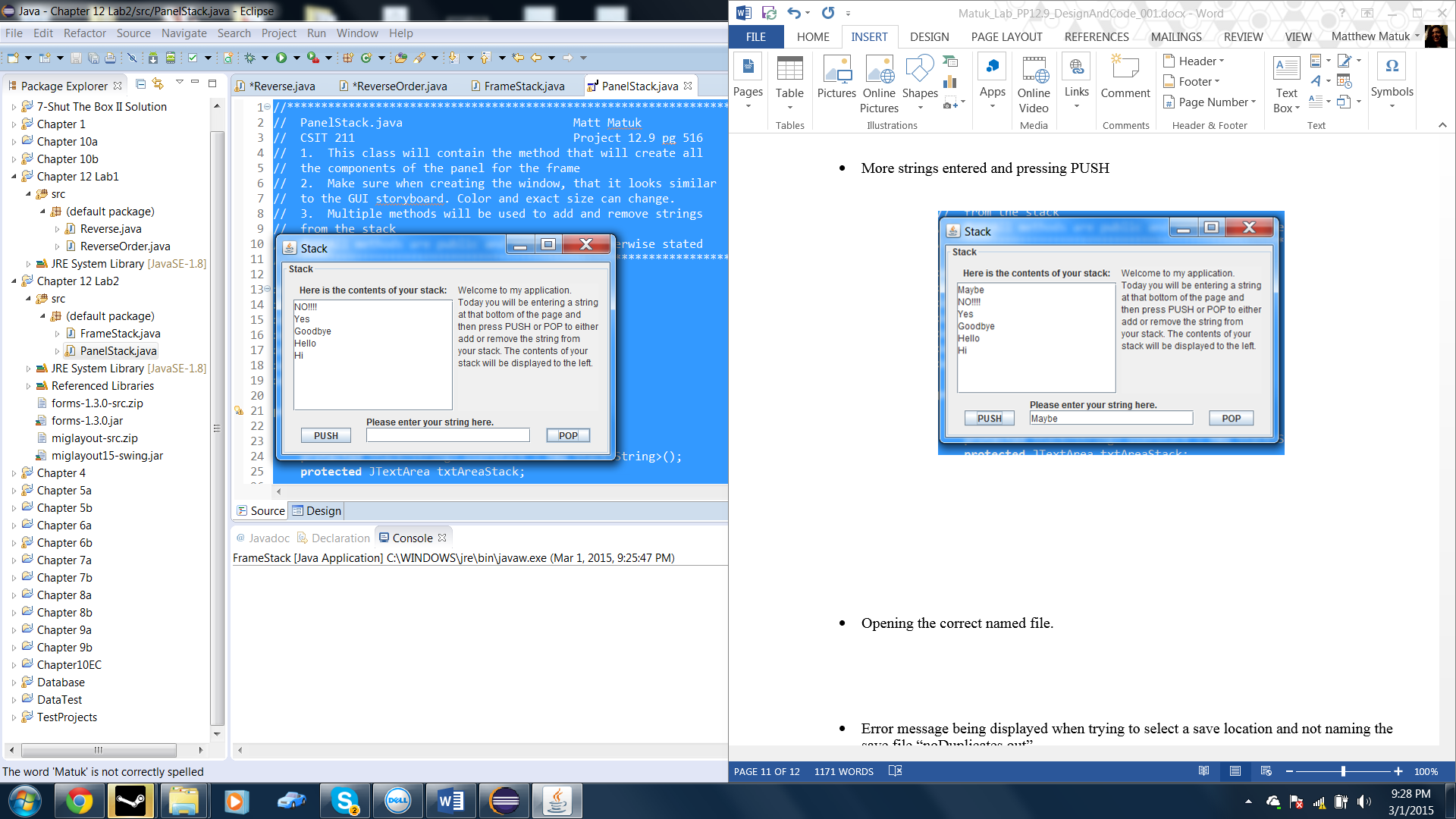
* String Hi entered an PUSH button pressed.



* More strings entered and pressing PUSH



* POP button pressed Once



* POP button pressed Five more times until the stack is empty

