**Course Project**

**DeVry University**

**College of Engineering and Information Sciences**

**Course Number: CIS355A**

# Module 2: Start Your GUI Application

# Objectives

* Take a UML Class diagram to code.
* Create a GUI application with labels, textfields, and buttons.
* Use a layout manager to place components on the GUI.
* Add events to the command buttons.

Once we have designed our application, we need to take the design to code and create the application. The process will follow a very systematic approach because you have finished the hard work—the design.

As you look at your design, implement the components and make the application work the way that you envisioned. First, create the GUI structure. Notice that the GUI class inherits from JFrame. As a result, we automatically get the close, minimize, and maximize buttons at the top, right corner, the title bar, and the basic application frame. We also automatically get the events for the close, minimize, maximize buttons. When you press the close button (red X at the top, right corner), the application will exit.

Next, we will add the components. In the NetBeans environment, the components can be added to the GUI using the drag and drop interface. You are going to love it.

However, once we have built the GUI, the application will not respond to your clicks. What are you going to do? You need to complete the final step. Add events to your buttons and other controls. Once you add the actionPerformed event to your buttons and the appropriate code, your application will respond and complete the task that you want to do.

Software development is fun. Let’s get started.

# Steps

1. Create a new NetBeans project and save it in your CIS355A folder. Call the project “CIS355A - LASTNAME Landscaping”. Save all your CIS355A files in this folder. Make sure you create a GUI application. The main method can be in a separate launching class or, more typically, you can have the main method in the main JFrame class. If you do not create any classes when you create your NetBeans application, NetBeans will create the main method for you in your first JFrame GUI class. Remember to turn off the checkmark for create main class.
2. Open Visio class diagram that you created during the design phase. Take the Customer class diagram to code. As a reminder, here is the Customer class diagram from the design phase. Remember to code your constructors and get and set methods.  
     
   
3. The toString( ) behavior should look like the following.  
     
   @Override  
   public String toString( )  
   {  
    return name; // only show the Customer’s name  
   }
4. The getDetails( ) behavior should look like this.  
     
   public String getDetails( )  
   {  
    DecimalFormat fmt = new DecimalFormat( "$#,##0.00" );  
    String output = name + "\n";

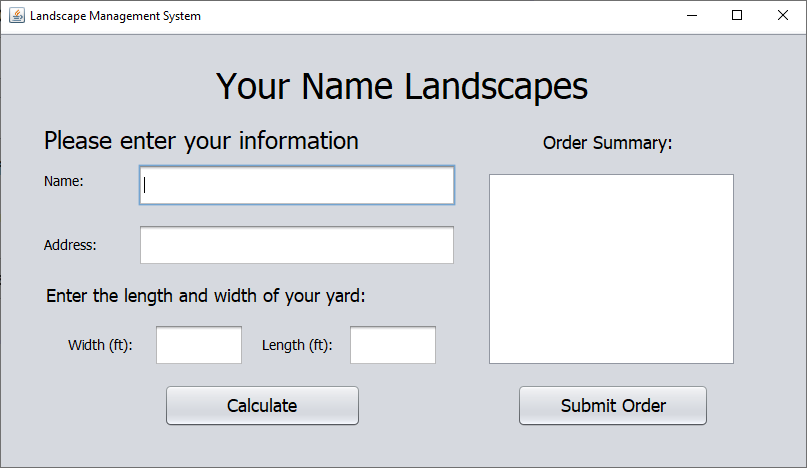
output += address + "\n";  
 output += "Type: " + yardType + "\n";

output += "Width: " + width + "\n";

output += "Length: " + length + "\n";

output += "Total Cost: " + fmt.format(totalCost); //currency

return output;  
}

1. Look at the SECOND page (Page-2) of your Visio wireframe diagram. To keep the process simple this week, we are only going to create the labels, textfields, and buttons from the SECOND panel (Page-2) on your wireframe diagram (not including the panel itself). Use the drag and drop interface on NetBeans to create this GUI. Note: a multiline textbox is called a JTextArea in Java. Look for the text area component in the swing controls on the right side of NetBeans.  
     
   
2. When you click on the calculate or the submit order buttons, nothing happens. We need to add events to these buttons. Right-click on the calculate button and choose Events 🡪 Action 🡪 actionPerformed. When you click on actionPerformed, NetBeans will create the actionPerformed event and method. NetBeans may take you to the method or, more likely, it will take you to the bottom of the code window. Scroll up and find the method for the event. If you called the button btnCalculate, then the event method will be called btnCalculateActionPerformed. Go inside the event method and add this code.

// validate the inputs  
if (validateInputs( ) == false )  
 return; // end the method if validation failed  
  
// create the Customer object and show the information  
Customer cust = createCustomer( );  
txaOrderInfo.setText( cust.getDetails( ) );

1. The code above requires two methods: validateInputs( ) and createCustomer( ). The first method checks the inputs and makes certain that they contain good values. The second method gets the values from the textfields and creates a customer object with these values. Add the validateInputs( ) method to the very bottom of your code window. Notice that the validateInputs( ) method checks for empty fields and then uses try-catch error handling to validate numeric inputs.

private boolean validateInputs()

{

String sName = txtName.getText();

String sAddress = txtAddress.getText();

String sWidth = txtWidth.getText();

String sLength = txtLength.getText();

if (sName.equals(""))

{

JOptionPane.showMessageDialog(this, "Enter a Name",

"Name Error", JOptionPane.ERROR\_MESSAGE);

txtName.requestFocusInWindow();

return false;

}

if (sAddress.equals(""))

{

JOptionPane.showMessageDialog(this, "Enter a Address",

"Address Error", JOptionPane.ERROR\_MESSAGE);

txtAddress.requestFocusInWindow();

return false;

}

if (sAddress.length() <= 5)

{

JOptionPane.showMessageDialog(this,

"Address isn't long enough.",

"Address Error", JOptionPane.ERROR\_MESSAGE);

txtAddress.requestFocusInWindow();

return false;

}

if (sWidth.equals(""))

{

JOptionPane.showMessageDialog(this, "Enter a Width",

"Width Error", JOptionPane.ERROR\_MESSAGE);

txtWidth.requestFocusInWindow();

return false;

}

try

{

Double.parseDouble(sWidth);

}

catch (NumberFormatException e)

{

JOptionPane.showMessageDialog(this, "Width must be a number",

"Width Error", JOptionPane.ERROR\_MESSAGE);

txtWidth.setText("");

txtWidth.requestFocusInWindow();

return false;

}

if (Double.parseDouble(sWidth) <= 0)

{

JOptionPane.showMessageDialog(this,

"Width must be greater than 0",

"Width Error", JOptionPane.ERROR\_MESSAGE);

txtLength.setText("");

txtLength.requestFocusInWindow();

return false;

}

if (sLength.equals(""))

{

JOptionPane.showMessageDialog(this, "Enter a Length",

"Length Error", JOptionPane.ERROR\_MESSAGE);

txtLength.requestFocusInWindow();

return false;

}

try

{

Double.parseDouble(sLength);

}

catch (NumberFormatException e)

{

JOptionPane.showMessageDialog(this, "Length must be a number",

"Length Error", JOptionPane.ERROR\_MESSAGE);

txtLength.setText("");

txtLength.requestFocusInWindow();

return false;

}

if (Double.parseDouble(sLength) <= 0)

{

JOptionPane.showMessageDialog(this,

"Length must be must be greater than 0",

"Length Error", JOptionPane.ERROR\_MESSAGE);

txtLength.setText("");

txtLength.requestFocusInWindow();

return false;

}

else

{

// all is good so return true

return true;

}

}

1. Now, we need to add the second method, createCustomer( ). Add this method to the very bottom of your code window.

private Customer createCustomer()

{

String name = txtName.getText();

String address = txtAddress.getText();

double width = Double.parseDouble(txtWidth.getText());

double length = Double.parseDouble(txtLength.getText());

String yardType = "Grass"; // FIX THE TYPE

double totalCost = width \* length \* 5.0; // FIX THE PRICE

Customer cust = new Customer(0, name, address, yardType,

length, width, totalCost);

return cust;

}

1. We are not ready to read and write to files. As a result, let’s add a stub to the submit order button. Right-click on the submit order button and choose Events 🡪 Action 🡪 actionPerformed. When you click on actionPerformed, NetBeans will create the actionPerformed event and method. NetBeans may take you to the method or, more likely, it will take you to the bottom of the code window. Scroll up and find the method for the event. If you called the button btnSubmitOrder, then the event method will be called btnSubmitOrderActionPerformed. Go inside the event method and add this code.  
     
   submitOrder();
2. Notice that we are only calling the submitOrder( ) method. We want to put the code into a method so we can call the method from multiple locations and we will not have to rewrite the code. For example, we can call the method from our menu. The buzzwords that you will hear in industry is write once, use everywhere. How do you write it once and then reuse it? Create a method. Ad this method to the very bottom of your code window. Notice that the method only contains a stub at this point.

public void submitOrder()

{

JOptionPane.showMessageDialog(this, "Method is not complete.");

}

1. Run your application and test it.

# Course Project Deliverables for Week 2

* Close NetBeans so your zip file will have the latest code. Then, go to the project folder. Right-click on the project folder and choose Send to 🡪 Compressed (zipped) file. Then, rename the zipped file to include your name. Submit the zipped file on Canvas.
  + Week 2 GUI – Your Name.zip