INFO213-22SU1 - Object-Oriented Systems Development

INFO213 Tutorial 03 - Instructions

Information

In this tutorial, we will learn how to utilise the Jade Painter.

In comparison to prior tutorials, there will be fewer step-by-step and click-by-click instructions.

We encourage you to attempt anything before seeking assistance with any aspect of this lesson.

Tutorial Objectives

The goal of this tutorial is to give a practical understanding of event-driven interfaces and introduce the JADE painter. On completing the tutorial you will be able to do the following:

- Navigate and modify the properties of forms in JADE schemas
- Create new interface forms in JADE Painter
- Enable the behind-the-scenes functionality for interface forms
- Utilise If statements and the Boolean type

Exercise 0: Preparing Development Environment

The JADE IDE on the tutorial room computers is configured to store the database on P drive at the following location: P:\Jade2020\system.

Please make sure you know how to verify JADE IDE is running with the correct settings for the JADE database location. If you are not sure, ask your tutor before proceeding any further.

Downloads Needed

- Download the '**DemoEvents**' schema from the Tutorial Material.
- Download the '**SimpleBankModel**' solutions from last week.
- After the schema files are downloaded you can load the schemas with the default names '**DemoEvents**' and '**SimpleBankModel**' respectively.

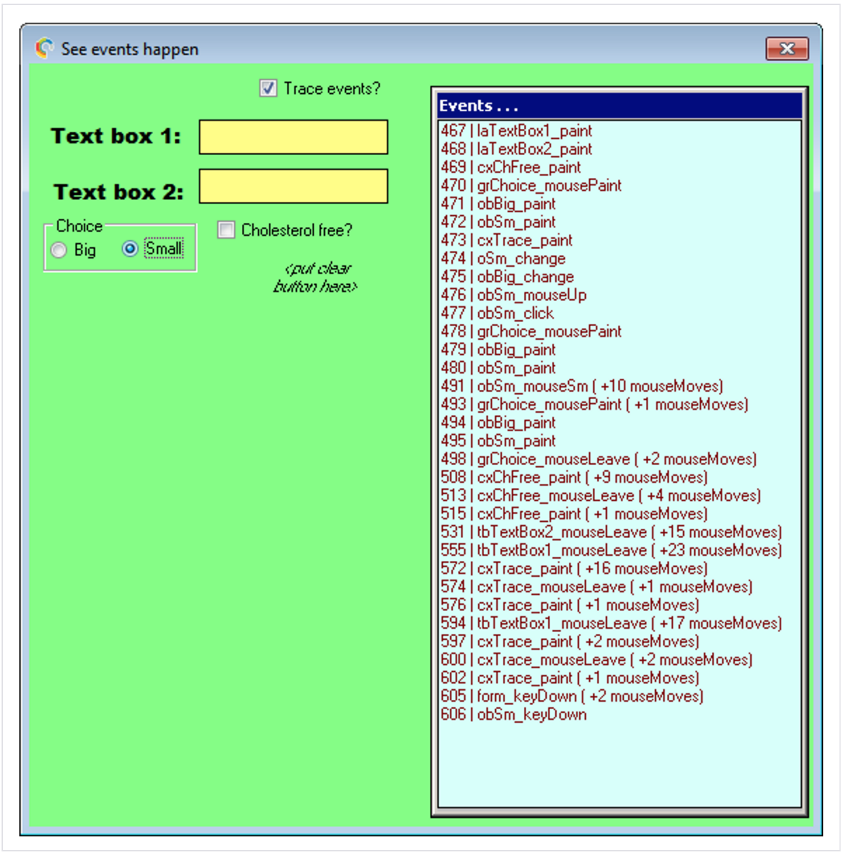
Please keep in mind that loading the downloaded SimpleBankModel schema may cause part of your code in the SimpleBankModel schema to be overwritten if it differs from the solution code.

Exercise 1: Explore And Modify DemoEvents Schema

In this exercise, you are invited to run and explore a new (and a little odd) schema in order to make some changes to the GUI of the application defined in the schema.

- Display the **Application** Browser dialog for the '**DemoEvents**' schema.
  - You can run the '**DemoEvents**' application GUI by selecting the '**DemoSeeEvents**' application from the list of applications for this schema and then selecting the **Run** context menu option.

- **Play** with this application to make sure you understand what you see on the screen.
  - Please make sure you examine the output as it is shown in the Events list:

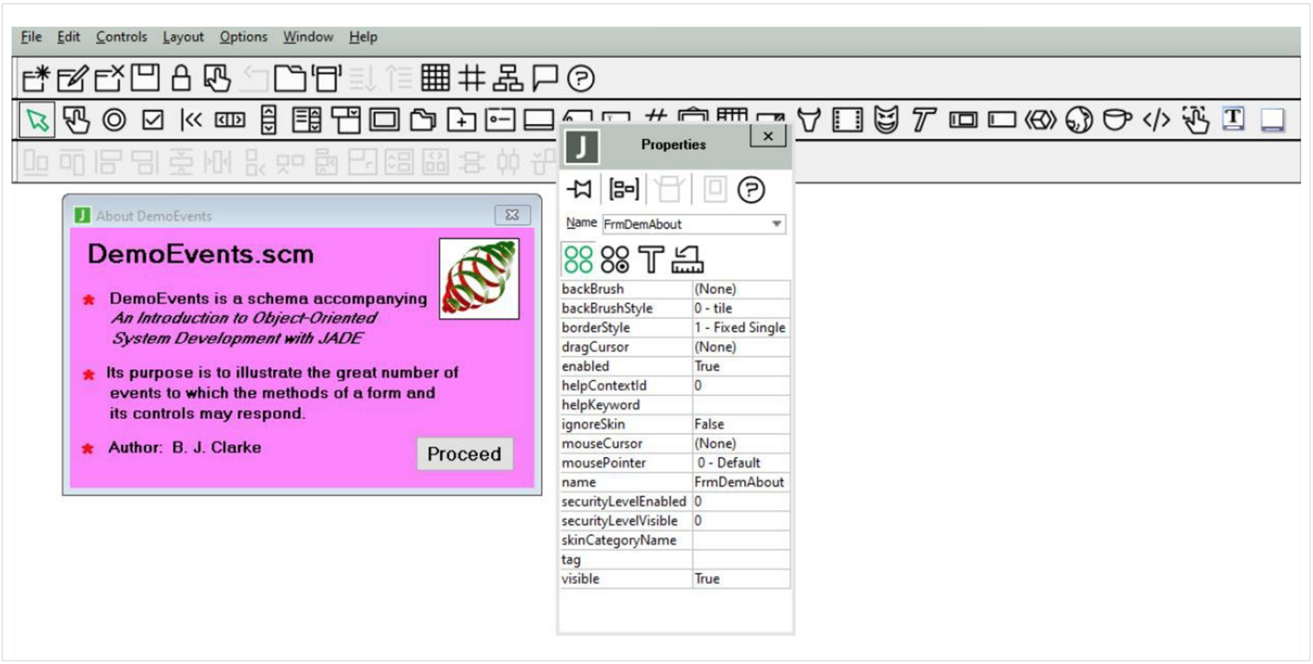


```
467 | laTextBox1_paint
468 | laTextBox2_paint
469 | cxChFree_paint
470 | grChoice_mousePaint
471 | obBig_paint
472 | obSm_paint
473 | cxTrace_paint
474 | oSm_change
475 | obBig_change
476 | obSm_mouseUp
477 | obSm_click
478 | grChoice_mousePaint
479 | obBig_paint
480 | obSm_paint
491 | obSm_mouseSm ( +10 mouseMoves)
493 | grChoice_mousePaint ( +1 mouseMoves)
494 | obBig_paint
495 | obSm_paint
498 | grChoice_mouseLeave ( +2 mouseMoves)
508 | cxChFree_paint ( +9 mouseMoves)
513 | cxChFree_mouseLeave ( +4 mouseMoves)
515 | cxChFree_paint ( +1 mouseMoves)
531 | tbTextBox2_mouseLeave ( +15 mouseMoves)
555 | tbTextBox1_mouseLeave ( +23 mouseMoves)
572 | cxTrace_paint ( +16 mouseMoves)
574 | cxTrace_mouseLeave ( +1 mouseMoves)
576 | cxTrace_paint ( +1 mouseMoves)
594 | tbTextBox1_mouseLeave ( +17 mouseMoves)
597 | cxTrace_paint ( +2 mouseMoves)
600 | cxTrace_mouseLeave ( +2 mouseMoves)
602 | cxTrace_paint ( +1 mouseMoves)
605 | form_keyDown ( +2 mouseMoves)
606 | obSm_keyDown
```

  - Q: Can you discover the method in the schema that is in charge of printing event information to the Events list? For example, if you examine the click method of the '**obBig** ' button in the '**FSeeEventsHappen**' form, what code does it call to report the click event?

If you want to open a form in **JADE Painter**, you can use the context menu that appears when you click on the form entry in the **Class Browser** dialogue, or you can open **JADE Painter** (from the main JADE IDE toolbar or from the **File | Painter** menu in the JADE IDE menu) and use the **File** menu to navigate and open the desired form.

- Open the '**FrmDemAbout**' form in Painter and bring up the Properties dialog:



DemoEvents.scm

- DemoEvents is a schema accompanying *An Introduction to Object-Oriented System Development with JADE*
- Its purpose is to illustrate the great number of events to which the methods of a form and its controls may respond.
- Author: B. J. Clarke

Name FrmDemAbout

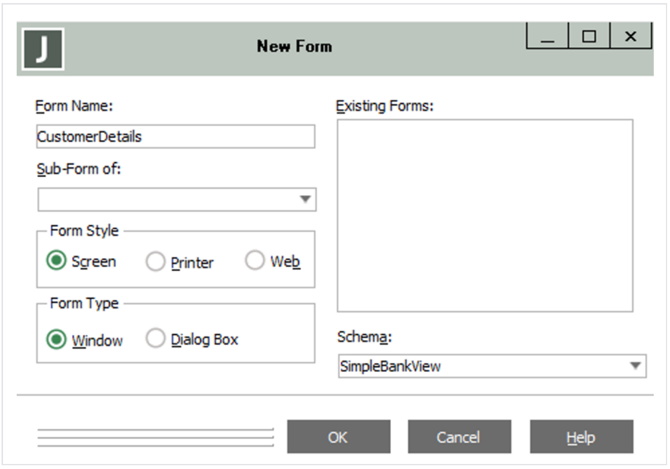
backBrush	(None)
backBrushStyle	0 - tile
borderStyle	1 - Fixed Single
dragCursor	(None)
enabled	True
helpContextId	0
helpKeyword	
ignoreSkin	False
mouseCursor	(None)
mousePointer	0 - Default
name	FrmDemAbout
securityLevelEnabled	0
securityLevelVisible	0
skinCategoryName	
tag	
visible	True

- Explore the components of this form, which include a bunch of text labels, a picture and a button.
- Find a way to change the disturbing pink background of the text labels to a more soothing colour.

Exercise 2: Working With Painter

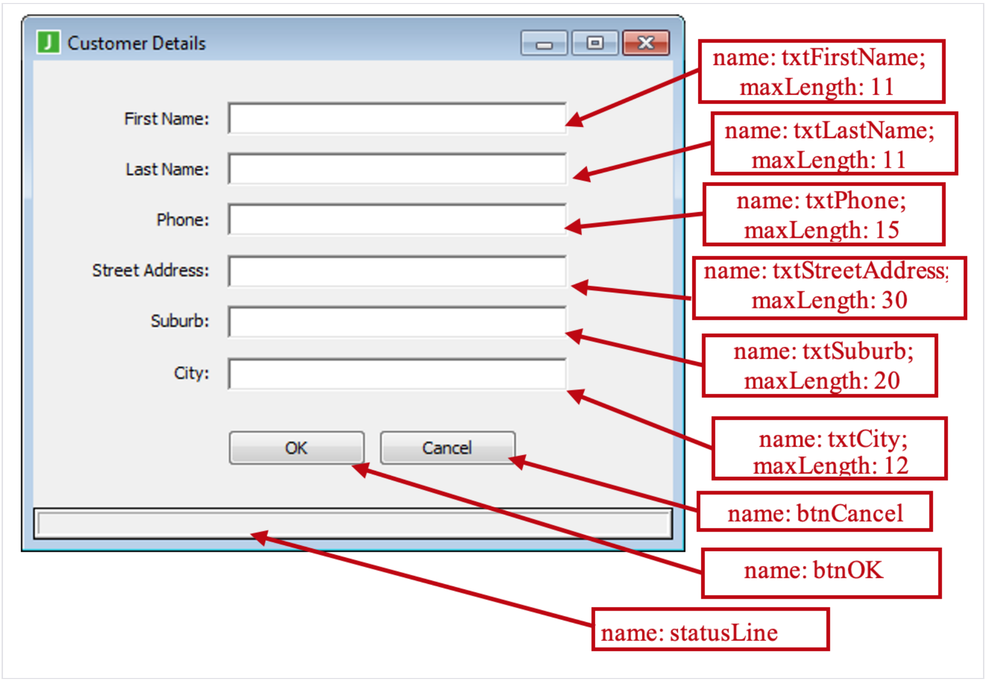
In this exercise, we will use Painter to create our first form. It is considered a good practice to separate the '**core business logic**' of a given project from its '**presentation**'.

- Create a new schema '**SimpleBankView**' as a sub-schema of our '**SimpleBankModel**' schema.
- Open **Painter** and create a new blank form in the new '**SimpleBankView**' sub-schema as follows called '**CustomerDetails**':



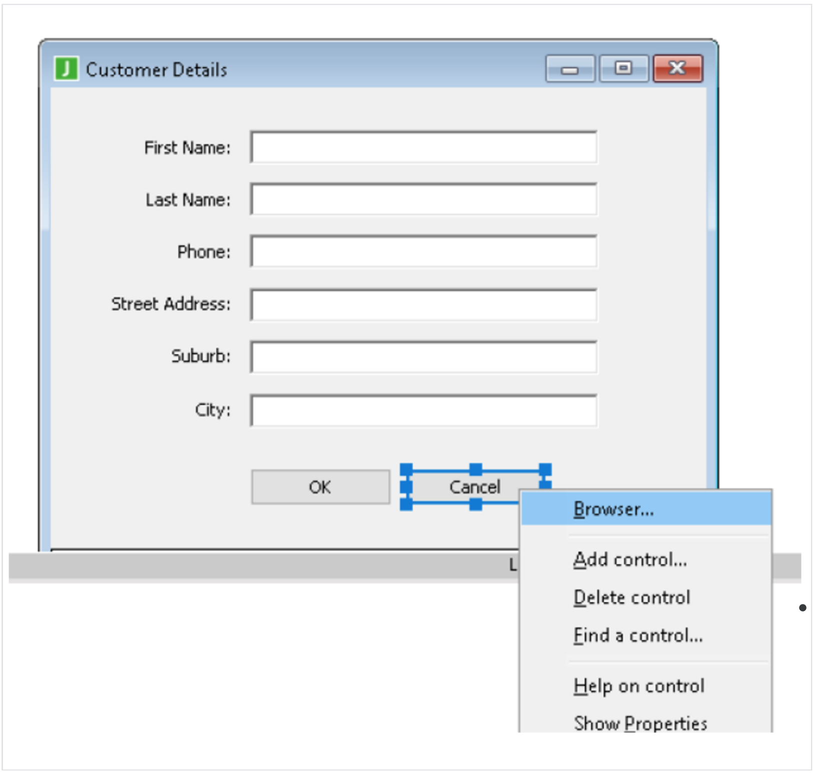
Form Name:	CustomerDetails
Sub-Form of:	
Form Style	[✓ Screen] Printer Web
Form Type	[✓ Window] Dialog Box
Existing Forms:	
Schema:	SimpleBankView

- Explore the **Painter** toolbar and other features.
- Add the following components into the '**CustomerDetails**' form (**TextBox**, **Button**, and **StatusLine**):

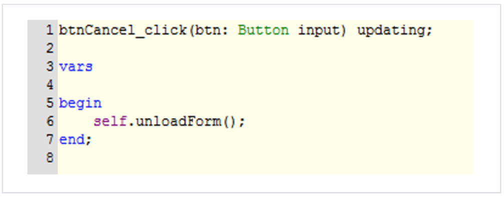


name: txtFirstName; maxLength: 11
name: txtLastName; maxLength: 11
name: txtPhone; maxLength: 15
name: txtStreetAddress; maxLength: 30
name: txtSuburb; maxLength: 20
name: txtCity; maxLength: 12
name: btnCancel
name: btnOK
name: statusLine

- Change the way the words appear on your screen with the **Captions** functionality.
- Use the **Label** components for placing text in the form and **TextBox** components for the input fields.
  - The (variable) names for labels are not essential, but the names of the input fields and buttons should be set as shown in the above screenshot.
- **Press F2 on your keyboard to save your work**, then navigate back to the Class Browser of the JADE 2020 IDE.

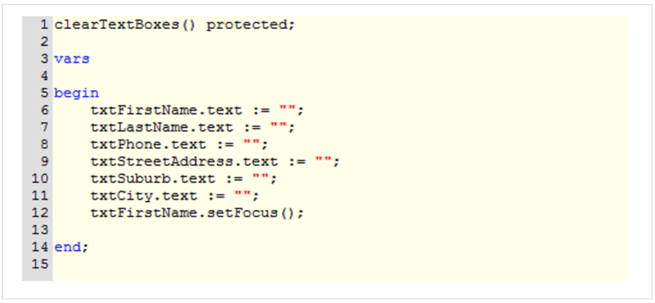


- Override the '**click**' method for the **btnCancel** button like this:



```
btnCancel_click(btn: Button input) updating;

vars

begin
    self.unloadForm();
end;

```

- Write a new form method called '**clearTextBoxes**' which will be inherited by the sub forms of this form at a later stage:

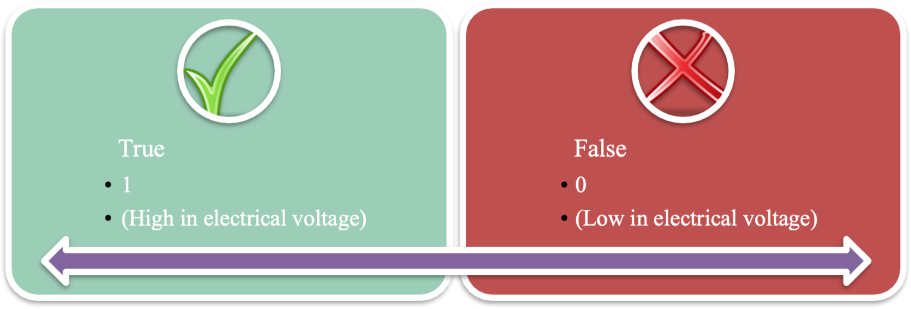


```
clearTextBoxes() protected;

vars

begin
    txtFirstName.text := "";
    txtLastName.text := "";
    txtPhone.text := "";
    txtStreetAddress.text := "";
    txtSuburb.text := "";
    txtCity.text := "";
    txtFirstName.setFocus();

end;

```

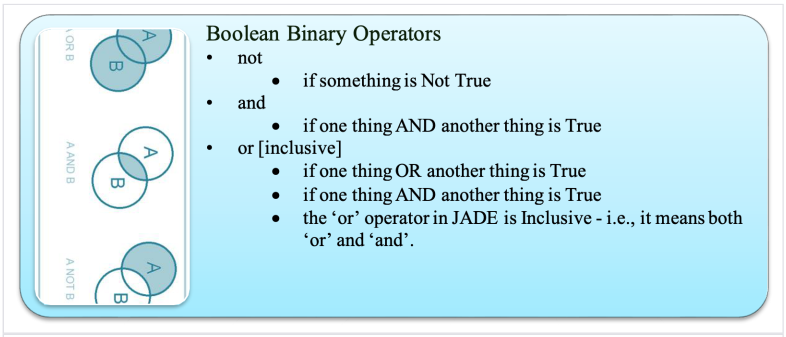
The following section of this exercise will have an **If-Elseif-Else** statement. It will then return a **Boolean** value.

Information -Boolean Type

- A Boolean type can only take one of two possible values:

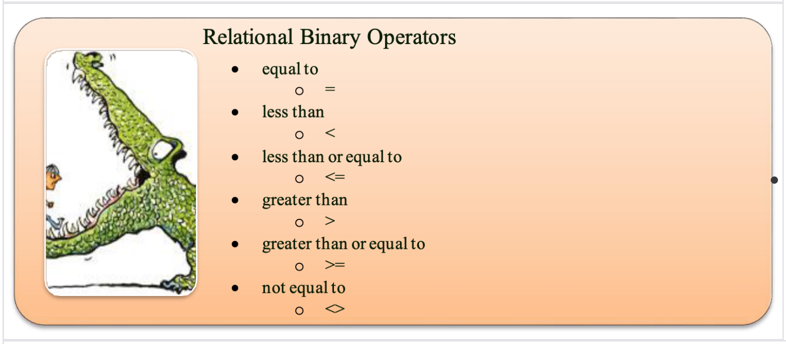


Information -Boolean Mathematics



Boolean Binary Operators

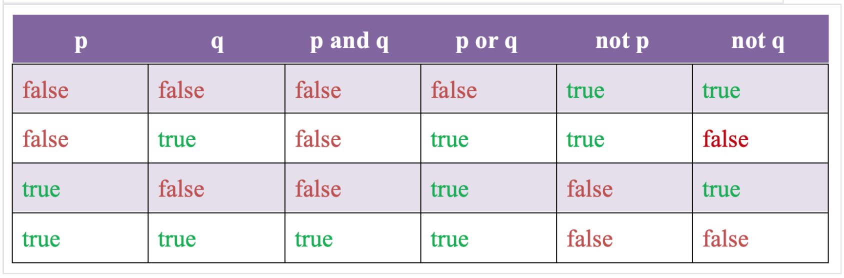
- not
  - if something is Not True
- and
  - if one thing AND another thing is True
- or [inclusive]
  - if one thing OR another thing is True
  - if one thing AND another thing is True
  - the ‘or’ operator in JADE is Inclusive - i.e., it means both ‘or’ and ‘and’.



Relational Binary Operators

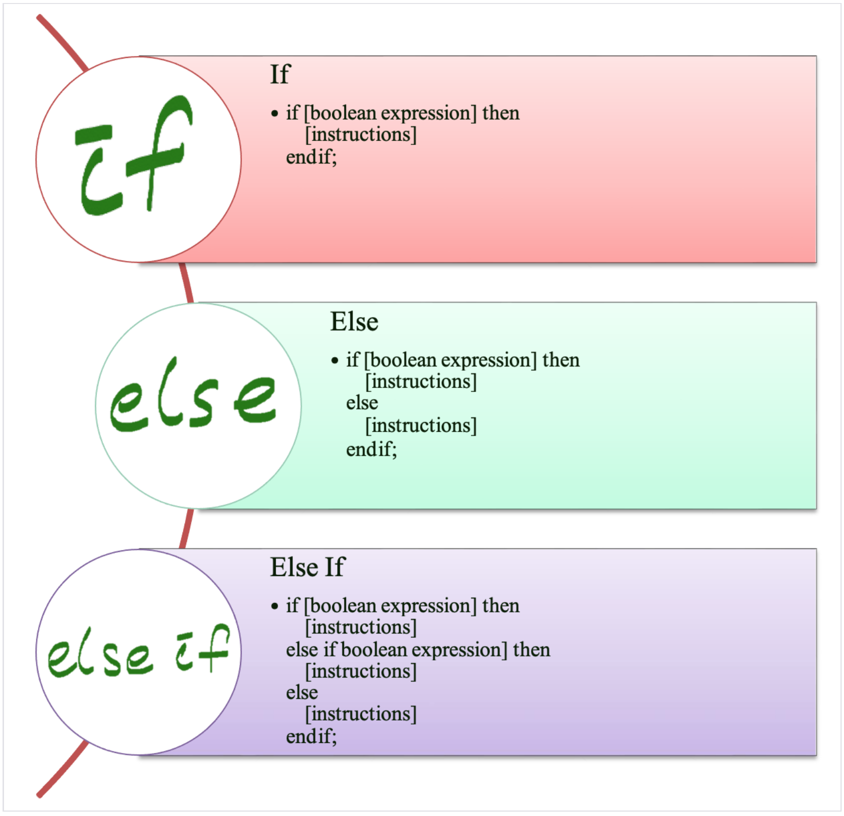
- equal to
  - =
- less than
  - <
- less than or equal to
  - <=
- greater than
  - >
- greater than or equal to
  - >=
- not equal to
  - <>

- Those of you who have taken MATH120 will find this very familiar...



p	q	p and q	p or q	not p	not q
false	false	false	false	true	true
false	true	false	true	true	false
true	false	false	true	false	true
true	true	true	true	false	false

Information - If-Elseif-Else Statements



If

```
• if [boolean expression] then
      [instructions]
  endif;
```

Else

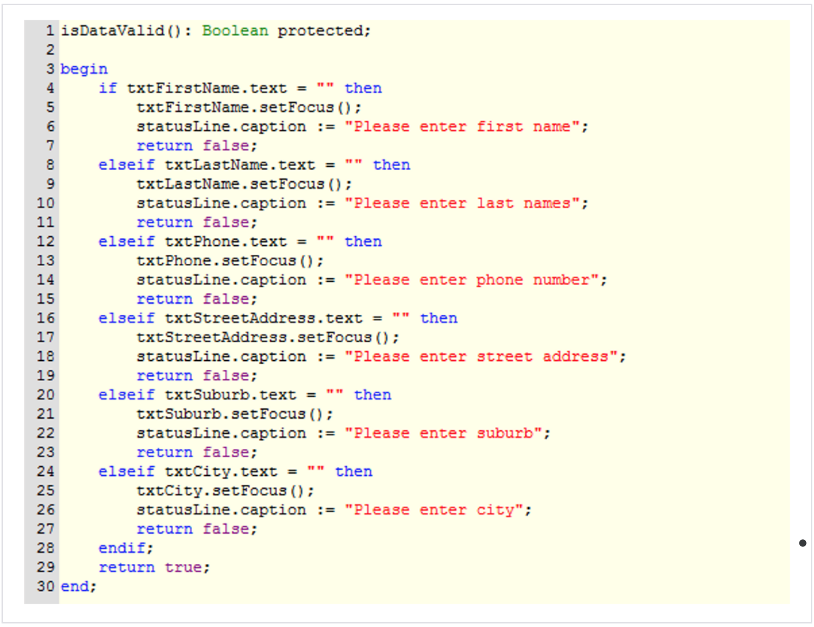
```
• if [boolean expression] then
      [instructions]
  else
      [instructions]
  endif;
```

Else If

```
• if [boolean expression] then
      [instructions]
  else if boolean expression] then
      [instructions]
  else
      [instructions]
  endif;
```

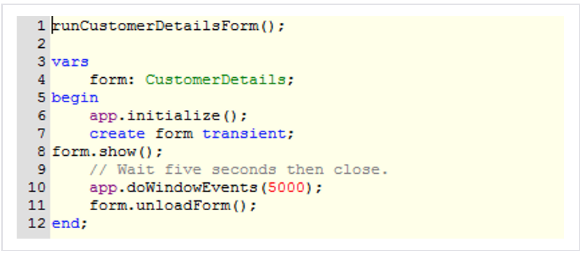
How To Work With Boolean Types

- Write another new form method called '**isDataValid**', which will also be inherited by the sub forms of this form at a later stage.



```
isDataValid(): Boolean protected;

begin
    if txtFirstName.text = "" then
        txtFirstName.setFocus();
        statusLine.caption := "Please enter first name";
        return false;
    elseif txtLastName.text = "" then
        txtLastName.setFocus();
        statusLine.caption := "Please enter last names";
        return false;
    elseif txtPhone.text = "" then
        txtPhone.setFocus();
        statusLine.caption := "Please enter phone number";
        return false;
    elseif txtStreetAddress.text = "" then
        txtStreetAddress.setFocus();
        statusLine.caption := "Please enter street address";
        return false;
    elseif txtSuburb.text = "" then
        txtSuburb.setFocus();
        statusLine.caption := "Please enter suburb";
        return false;
    elseif txtCity.text = "" then
        txtCity.setFocus();
        statusLine.caption := "Please enter city";
        return false;
    endif;
    return true;
end;
```

- Normally, you can run a form directly from JADE Painter, but sometimes a form may require some initialisation code from the core application.
    - To run this form, create a **JadeScript method** named '**runCustomerDetailsForm**' and make sure that the form is displayed correctly.

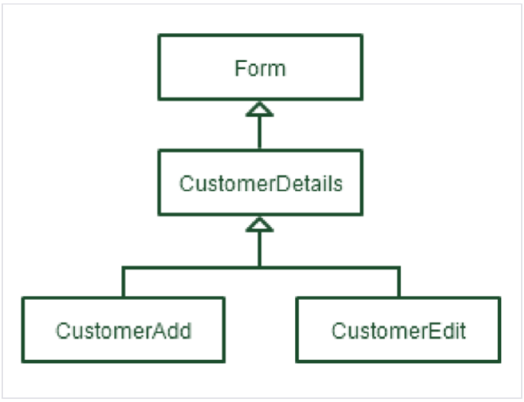


```
runCustomerDetailsForm();

vars
    form: CustomerDetails;
begin
    app.initialize();
    create form transient;
form.show();
    // Wait five seconds then close.
    app.doWindowEvents(5000);
    form.unloadForm();
end;
```

Exercise 3: Working With Sub Forms

Information - Sub Forms

- We anticipate the need for two very similar forms in our '**SimpleBankModel**' application for two separate use cases: adding new customers and editing the details of existing customers. For example, our forms may be organised as follows.

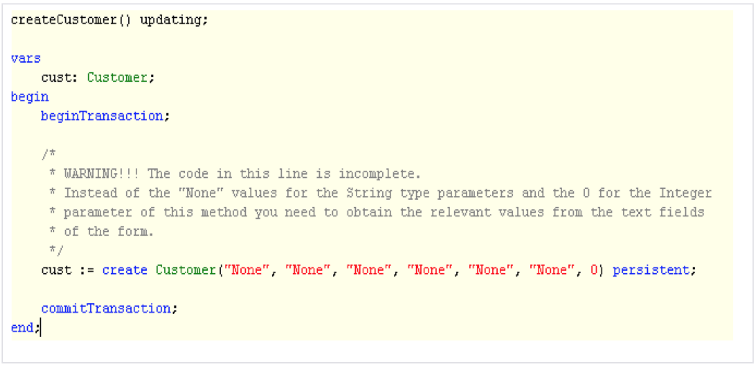


- To save time, JADE allows us to create sub forms (similar to sub classes).

How To Work With Sub Forms

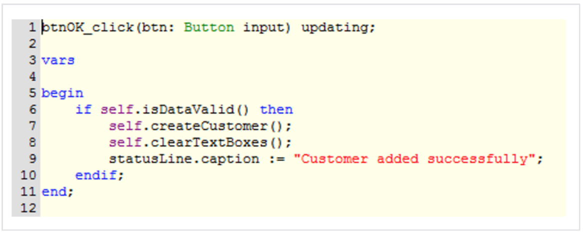
- Start by adding a new form called '**CustomerAdd**' in JADE painter, but be sure to specify that it will be a sub form of the '**CustomerDetails**' form.
    - The '**CustomerEdit**' form will be added in the next tutorial.
- Add a new method in the '**CustomerAdd**' form called '**createCustomer**'.
    - This method in the form is intended to collect data from the text fields of the form and pass it to the newly created Customer instance.

- Complete the missing code as indicated in the comments:



```
createCustomer() updating;

vars
    cust: Customer;
begin
    beginTransaction;

    /*
     * WARNING!!! The code in this line is incomplete.
     * Instead of the "None" values for the String type parameters and the 0 for the Integer
     * parameter of this method you need to obtain the relevant values from the text fields
     * of the form.
     */
    cust := create Customer("None", "None", "None", "None", "None", "None", 0) persistent;

    commitTransaction;
end;
```

- You need to override the '**click**' method of the '**btnOK**' button in the '**CustomerAdd**' form as shown in the next code snippet.
  - However, the form elements inherited from the '**CustomerDetails**' form (e.g., the btnOK button) will not be shown at first .
  - In order to show such elements, select the **View | Show Inherited** menu.
  - Then you may override the '**click**' method. Note the compounded method name.



```
btnOK_click(btn: Button input) updating;

vars

begin
    if self.isDataValid() then
        self.createCustomer();
        self.clearTextBoxes();
        statusLine.caption := "Customer added successfully";
    endif;
end;
```

- In order to be able to test your code, add a new **JadeScript method** called '**runCustomerAddForm**'.
  - This method should be very similar to the '**runCustomerDetailsForm**' you created earlier, except that you will not need to include the line to unload the form.
- If you did everything correctly, you should be able to run the '**CustomerAdd**' form, enter new customer details, and create a new **Customer instance** in the JADE database.
  - Please verify this by inspecting the list of created Customer instances.
- **Q: How would you evaluate the user experience satisfaction from your interaction with this form if you did not know that all input fields must have a value in them?**

Last modified: Monday, 17 January 2022, 4:33 PM