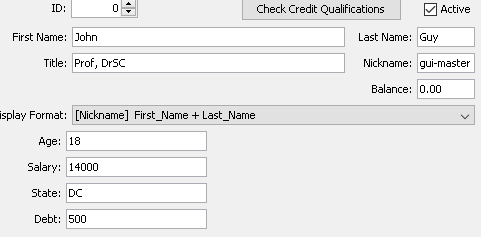
Advanced Programming Using Java - Lesson 2 (Chapter 1 Part B)

GOALS: Input/Output, Control Structures, Exceptions

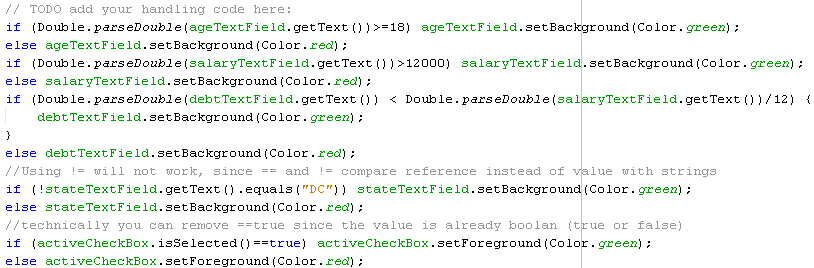
* Open acp1 (the ContactEditor). Expand Source Packages, examples, ContactEditor.java

Use comparison operators to check if a person qualifies for a store credit card:

* Drag a button to the form and change the text to “Check Credit Qualifications” in the Properties tab.
* Make 4 fields that describe the person’s financial qualifications. You can drag one label, one text field, then ctrl click so both are selected, then Ctrl C to copy, then Ctrl V to paste then move then keep pasting until there are 4 fields with labels (see picture below)
* In the Properties tab, change the text on the labels to “Age:”, “Salary:”, “State:”, “Debt:”, then also in properties turn on the Active checkbox’s “selected” option.
* Change the text of the matching text fields in that order to 18, 14000, DC, 500
* In the Code tab, change the Variable Names of the Text Fields to ageTextField, salaryTextField, stateTextField, and debtTextField. Then click the Active checkbox, and in the Code tab change the Variable Name to activeCheckBox



* Double-click your Check Credit Qualifications button you made, and edit the code to include the comparisons to make the fields red or green for whether the person is qualified (code is shown below). The program will autotype, so make sure you are watching the screen as you type so that you don’t type things twice. After you do the first two lines, copy and paste then change the name of the field (used twice on first line and again on next line), the comparison operator, and the number so the code is like the code shown. The checkbox background color is not visible, so change that to setForeground to change the text color for activeCheckBox (seen below). When you want statements on a separate line as shown below, or more than one statement is made in a scope, curly braces must be used:



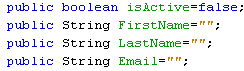
Add a While Loop

First get some example contact files in YAML format, containing FirstName, LastName, Email, and a Boolean called Active, from the acp1data folder in Resources:

* In File Explorer, go to Resources R:\Classes\AdvancedProgramming\acp1data
* Ctrl A to select all, Ctrl C to copy.
* Go to your home drive, then open your acp1 folder which contains your ContactForm project.
* Ctrl V to paste (3 files should appear: 0.yml, 1.yml, 2.yml)
* Double click one of them to make sure they are in the format described above

Create a new class that will be the model holding all data for the contacts:

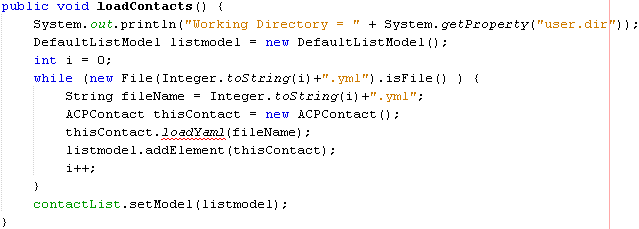
* Open NetBeans, and make sure you have the project with the ContactForm open such as File, Open Project, acp1
* File, New File, Java, Java Class, Next
  + Name it ACPContact
  + Make sure you have left the defaults such as : Project acp1, Location: Source Packages, Package: examples
  + Finish
* Add the following variables inside the curly braces of the ACPContact class:



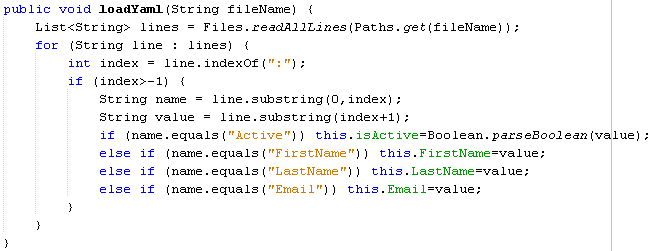
* Then for each of those lines, click anywhere on the line then click the light bulb that appears on the left of it, then “move initializer to constructors.” From now on the default value will be set whenever the constructor is called (whenever a new contact object is created).
* Click the “x” by ACPContact.java tab at the top to close it.

Create a loadContacts method, and call the method in the form constructor so all contacts are loaded when the program starts:

* Go back to the ContactEditor tab
* Click on the list on your form, Code tab, then change Variable Name to contactList
* Click the Source tab at the top.
* Find the ContactEditor constructor at the beginning. It is inside the ContactEditor Class. The constructor has no return type specified so it will just say “public ContactEditor” (it returns a ContactEditor when used after the “new” operator—this is how constructors work, and how new objects are always created in Java and many other programming languages)
  + Create a loadContacts method before the constructor:



* + Click the “DefaultListModel” line you will see a bulb on the left of the line. Click the bulb to explain the error. Click Add import for DefaultListModel and then a new line will appear at the top of the code to import that Java library
  + Click the loadYaml line, then click the bulb on the left to create a loadYaml method in ACPContact
    - Click the ACPContact.java tab.
    - Remove the placeholder code throw new UnsupportedOperationException
    - Add the following code to the method instead, which will analyze the file (for more complex Yaml files, snakeYaml can be used, but for this, just find the colon to find the value):



* + - Click the Files line, then click the bulb beside it to import the Files class [If the bulb says to create classes, try retyping the word “Files” so the IDE searches Java again for the built-in classes, then do this step again].
    - The readAllLines method could fail and cause the program to crash if the file has a problem. Click the line then the lightbulb, then “Surround Block with try-catch”. Take a look at the code that is added—now all the code you wrote should be inside a “try” block (in its curly braces), and then afterward is a catch statement that determines what to do if there was a problem (shows the error on the debugging console in your IDE such as on the NetBeans bottom panel). Also, at the very top, the import java.io.IOException should now have been added automatically.
    - Add the additional imports needed by clicking each line with an underline as long as they are valid Java classes typed correctly as in code above, such as the Files class.
      * If you get an error that says “List does not take parameters”, go to the top of the source code and change “import java.awt.List” to “import java.util.List”
    - Also in the ACPContact class add the following method to tell Java how to display the object in the list:



* + Inside the ContactEditor constructor (in the curly braces), add code that loads the contacts into the list (in Java custom lists use the ArrayList constructor) AFTER the existing initComponents line:

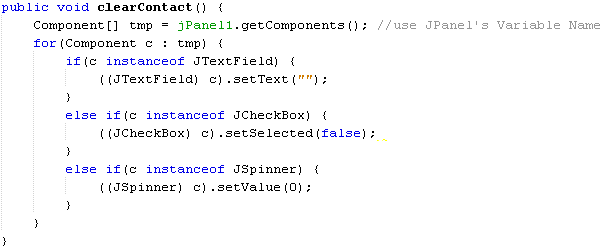


* + Run it. If you did everything right so far, the list should now contain (only) the contacts from the 3 files:



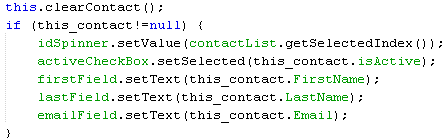
Add a feature to actually load the rest of the data when the person is clicked:

* Write a clearContact method in the ContactEditor class, such as before the loadContacts method:



**BONUS: add Age, Salary, State & Debt to the class, the load method, and the save method.**

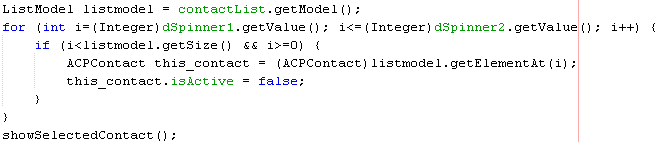
Make the Contact Load When Clicked

* In the ContactEditor, click Design
* Click the white box for First Name then click “Code” tab on right panel, then change Variable Name to firstField
  + Also name the other fields: lastField, idSpinner, emailField
* Click the list. Click the Events tab on the right panel. Beside “valueChanged” hit the down arrow on the dropdown box, then choose the method name (that will be created automatically).
  + You should see the contactListValueChanged method and your cursor will be there. If you can’t see your cursor, hit an arrow key on the keyboard and the window will scroll to contactListValueChanged. Enter the following function call (we will create the function afterward): 
  + Click the light bulb and then click “create method showSelectedContact”
  + Scroll down to the new showSelectedContact method that appeared further down, remove the throw line, then add the following code (we will add the showContact method afterward):
  + Click the showContact line then click the bulb, then click “create method showContact.”
  + Scroll down to the showContact method that appeared further down, remove the throw line, then add the following code to show the contact: 

See if it worked: run the program, click an email, and see if the contact loads and other fields become blank.

User a For-Loop Structure to Make A Set of Users Inactive

* Go to ContactEditor and click Design
* Drag a button, then next to it drag a spinner, than another spinner
* Click the button, and in Properties change the text to “Deactivate Range:”
* Click the first spinner, then in the Code tab (near the properties tab on the right) change Variable Name to dSpinner1, and the next one’s Variable Name to dSpinner2
* Double-click your “Deactivate Range” button and add a loop that will deactivate a certain range of people (you should see an ActionPerformed method with your cursor in it—if you don’t see your cursor, hit an arrow key to find the correct action handler) then add the following code which includes a for-loop, to do multiple operations, in this case using a variable named i as the counter, starting at the first spinner’s value and ending at the second’s:



See if it worked: run the program, change second spinner to 1 then click Deactivate. Click the first Email then the second & see if Active is unchecked for both but not the third (if there are 3 entries, they are numbered 0 to 2).

If there is an error saying cannot convert the ACPContact to String, remove Type Parameters (such as, leave it blank instead of saying “<String> “) from the List’s Code tab in Design View. This will allow other types of objects to be added to the list’s model.