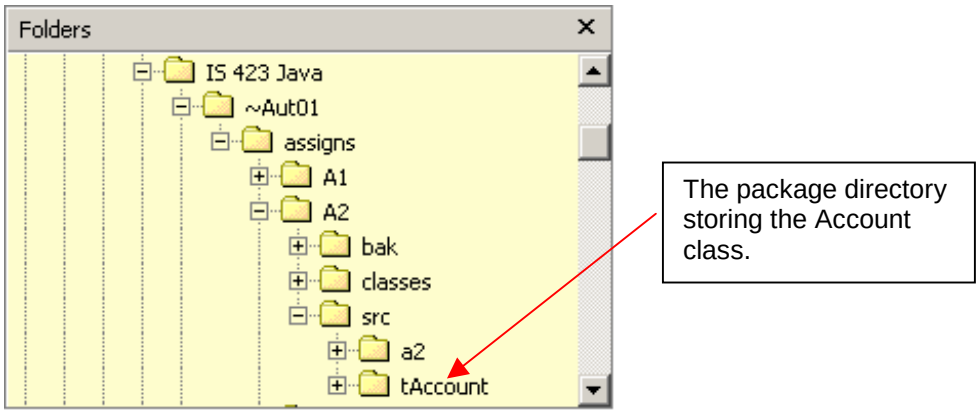
For this assignment you are required to create an Account class and instantiate a number of objects from this class. The Account class includes the following instance variables:

fAcctNo:	(String)
fSSN:	(String)
fBalance:	(double)

You also need to define several methods for this class. The constructor method should be passed values for the three instance variables and should use them to define each instance variable. In addition to the constructor method, define the following methods:

getAcctNo:	an accessor method that returns the current value of fAcctNo.
getSSN:	an accessor method that returns the current value of fSSN.
getBalance:	an accessor method that returns the current value of fBalance.
deposit:	an action method that *increases* fBalance by an amount passed to the method.
withdrawal:	an action method that *decreases* fBalance by an amount passed to the method.

As a **required** part of the assignment, you **must** create your class definition in a separate package – see pages 164-172 in your text for information on how to do this. You must import this package into your project. Note that if you create your package in a directory that is a subdirectory of your project "src" directory (see below), then JBuilder will take care of the ClassPath for you.



Your program needs to be able to create and store more than one account object. Use an array to do this. The array (and information about the array such as its length and the current number of elements used) must be available in several methods of the main JFrame class. If you declare these variables at the beginning of the JFrame class (before the JBuilder defined constructor) as

private class members, they will be available to all the methods defined for this class (all of its actionPerformed events). Their scope will also be extended so that as long as the program is running, the variables will be in scope and their values will be retained. For example, in the code below, the two variables numAccts and MAXACCTS have a class-wide scope.

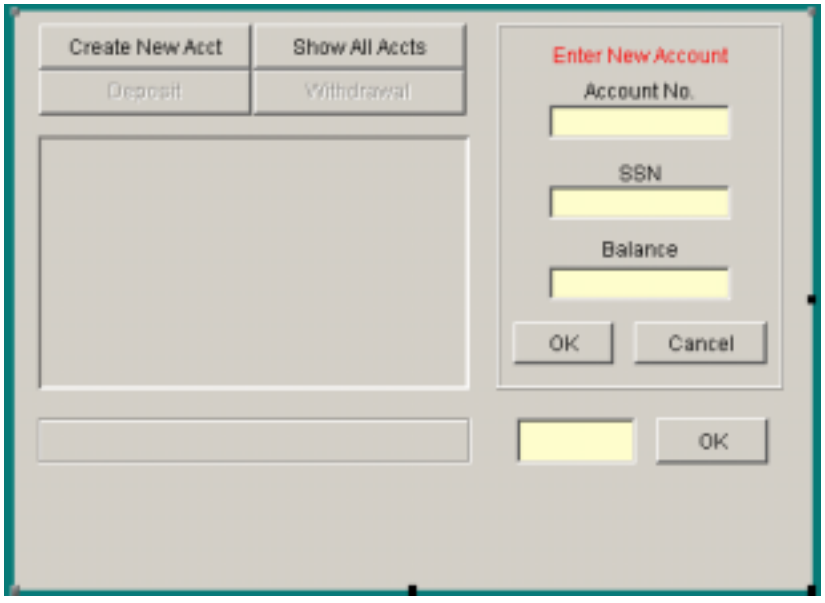
```
public class A2Frame extends JFrame {

    //create my class variables
    private int numAccts = 0;
    private static final int MAXACCTS = 10;
```

Build a GUI that supports the ability to:

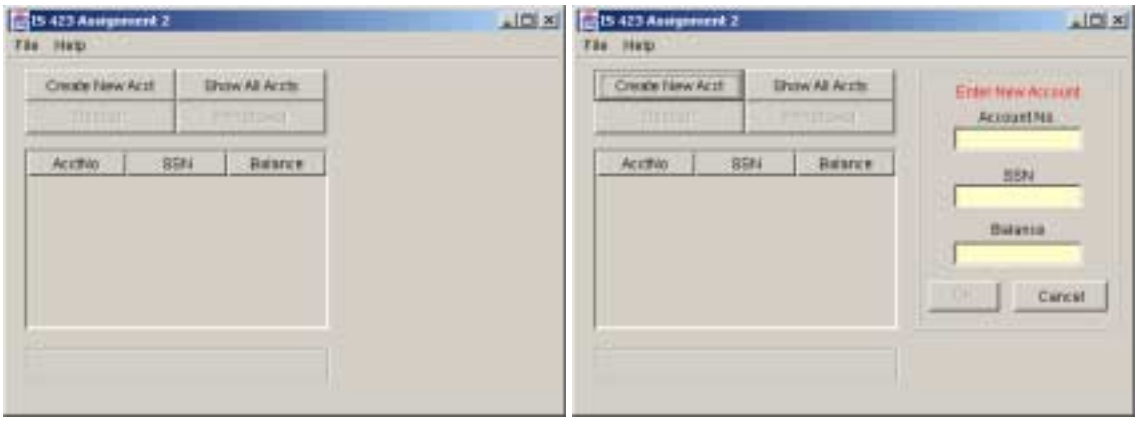
- create new accounts
- show all accounts
- process a deposit or withdrawal on a specific account

The following image shows an example form that you should emulate. The form is shown in design mode with a JPanel component at the right that relates to new account information. There is also a JLabel at the bottom left that initially contains no text. The rectangle below the four JButtons is a JScrollPane with a JTable inside it.



The JPanel on the right is initially not visible at run time (nor is the text field and command button at the bottom of the form) The panel becomes visible when the “Create New Acct” button is pressed. The running application is shown below before and after the “Create New Acct” button is pressed. The panel should become invisible again after the “OK” or “Cancel” buttons have been pressed.

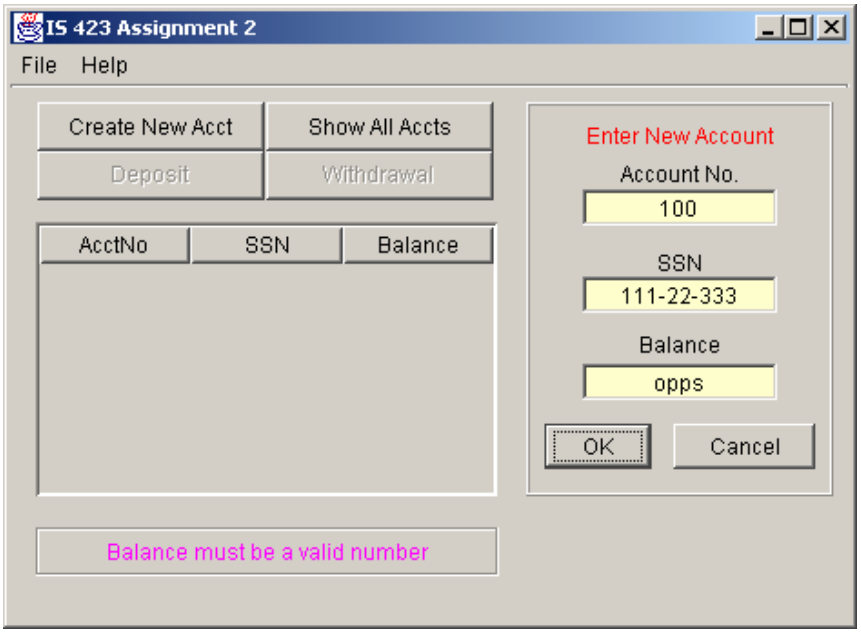
Notice that the “OK” button is not enabled and will not become enabled until the user has entered information into all three of the text fields.



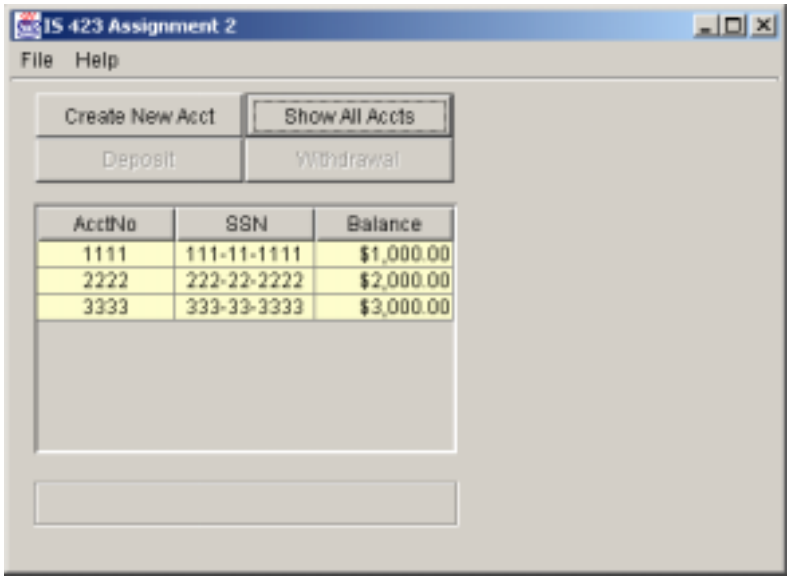
All components have a setVisible() method that can be used to make them visible and invisible. The setVisible() method uses a boolean (true/false) parameter to accomplish this. When a panel is not visible, the components it contains are also not visible.

After you place the JPanel on the frame at design time, be sure you set its Layout property to “null”. Otherwise you will not be able to place the components inside the panel where you want them.

Your application should use the JLabel at the bottom left to inform the user of possible error conditions as well as other information. For example, you cannot create a new account if the balance information is not a valid number. The contents of the label alerts the user to this situation when the OK button is clicked.

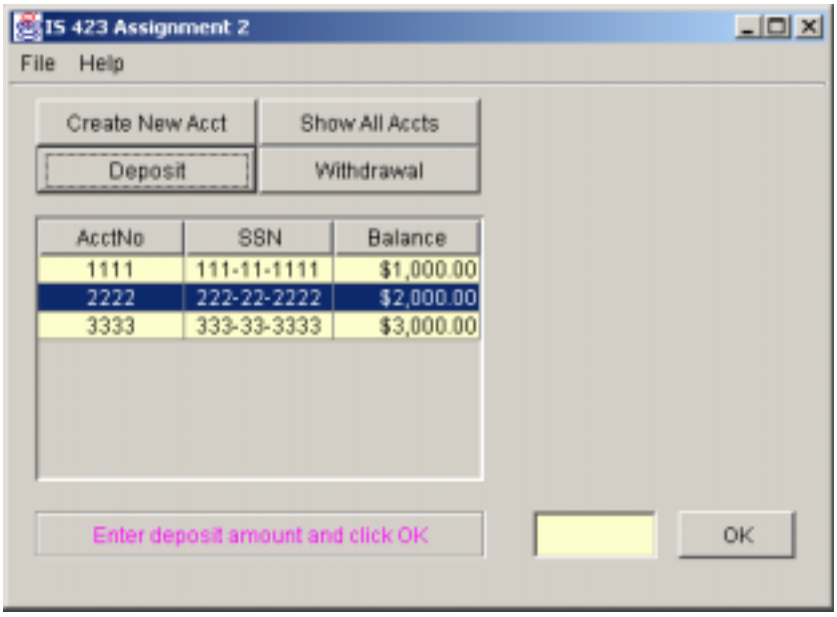


Clicking on the “Show All Accts” button should cause the account objects stored in the array to be displayed in the JTable component. The image below shows the information on three account objects created by the user.



Notice that the Deposit and Withdrawal buttons are not enabled (they have not been enabled on any of the images). To make the buttons enabled, the user must click on one of the accounts in the JTable. Use a mousePressed event for the table to detect the user selection of a row in the table.

In the image below, the user has clicked on account 2222 to select it and then clicked on the “Deposit” button. Notice the text in the label at the bottom of the screen and the fact that the text box and command button used to enter the deposit are now visible.



Your application needs to take the deposit (or withdrawal) information and apply it to the selected account object. You should be able to use the JTable component's getSelectedRow() method to help you access the correct object in your array. Remember that both arrays and JTable rows are numbered starting at zero.

After the user enters a deposit (or withdrawal) amount and clicks on OK, the Deposit and Withdrawal buttons should be disabled until the user clicks again on a row in the table.

To Turn In

Use the same procedure as you did for Assignment #1. That is, please store your project on the computer's C: drive in a folder named "is423". Within that folder, the project and its files should be stored in a folder named A2. After you have finished your project, use a ZIP utility to zip the project folder (A2 in this case) and all its subfolders and email it as an attachment to is423wb@u.washington.edu. I will grade the project and return an evaluation form and grade.

Cautions

Whenever you use an array, be sure that your code is protected from the possibility that the bounds of the array might be exceeded.

Also, be sure you are using the methods of the Account class to perform the various transactions.