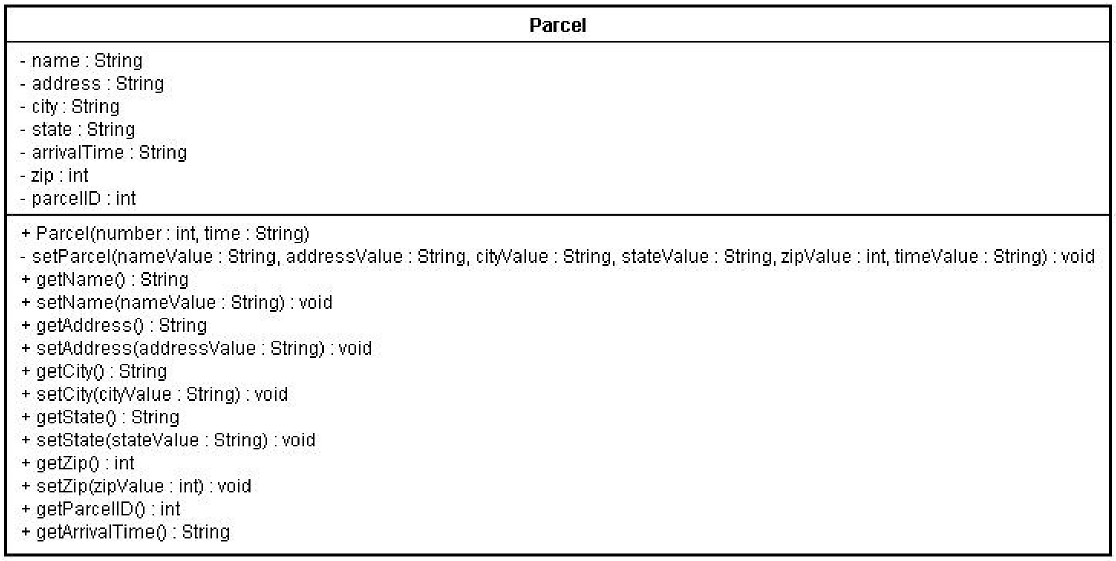
DUE: 4/23/2013

Project #1—Shipping Hub

A shipping company receives parcels at its shipping hub. Parcels will be represented as Parcel objects. The company then ships these parcels to a distribution center in one of the following states: Alabama, Florida, Georgia, Kentucky, Mississippi, North Carolina, South Carolina, Tennessee, Virginia or West Virginia. The company needs an application to track the Parcels that pass through its shipping hub. When the user clicks the application's Scan New JButton, the application generates an ID and arrival time for the new Parcel. Once a Parcel has been scanned, the user should be able to enter recipient's name and shipping address for the Parcel. JButtons should be provided for the user to remove or modify the information of Parcels that have already been scanned. The user should be able to navigate through the list of scanned Parcels by using the <Back or Next> JButtons. Finally, users should be able to view a numeric list of all Parcels destined for a particular state.



```
Parcel
- name : String
- address : String
- city : String
- state : String
- arrivalTime : String
- zip : int
- parcelID : int
+ Parcel(number : int, time : String)
- setParcel(nameValue : String, addressValue : String, cityValue : String, stateValue : String, zipValue : int, timeValue : String) : void
+ getName() : String
+ setName(nameValue : String) : void
+ getAddress() : String
+ setAddress(addressValue : String) : void
+ getCity() : String
+ setCity(cityValue : String) : void
+ getState() : String
+ setState(stateValue : String) : void
+ getZip() : int
+ setZip(zipValue : int) : void
+ getParcelID() : int
+ getArrivalTime() : String
```

Your application must "remember" each Parcel's shipping information, including recipient's name, address, city, state and zip code. However, multiple parcels can be shipped to the same person at the same location; therefore, each Parcel should be identified by unique identification number. For this application it is OK to start with 1 for the ID number and increment it by one with each new scan. The UML diagram above describes the instance variables , methods and constructors of the Parcel class involved (and required). Notice that there are no setParcelID or setArrivalTime method—this is because the user of class Parcel should not be able to modify the Parcel's identification number or arrival time after the Parcel object has been created. There ought to a default constructor (not in the UML above).

Initially, only the Scan New JButton (and from the menu Action->Scan New) is enabled. Clicking on it "opens up" the input fields for the customer's name and address. Parcel ID is assigned automatically and sequentially and a date/time is "stamped" in the Arrived at: field. The

If valid data is entered (consider using JFormattedTextField for the zip code; read the states from an external, comma delimited text file into a read-only State JComboBox) the Scan New JButton disables itself and it enables the Add JButton (with similar action in the Action JMenu).

Screen Shots:

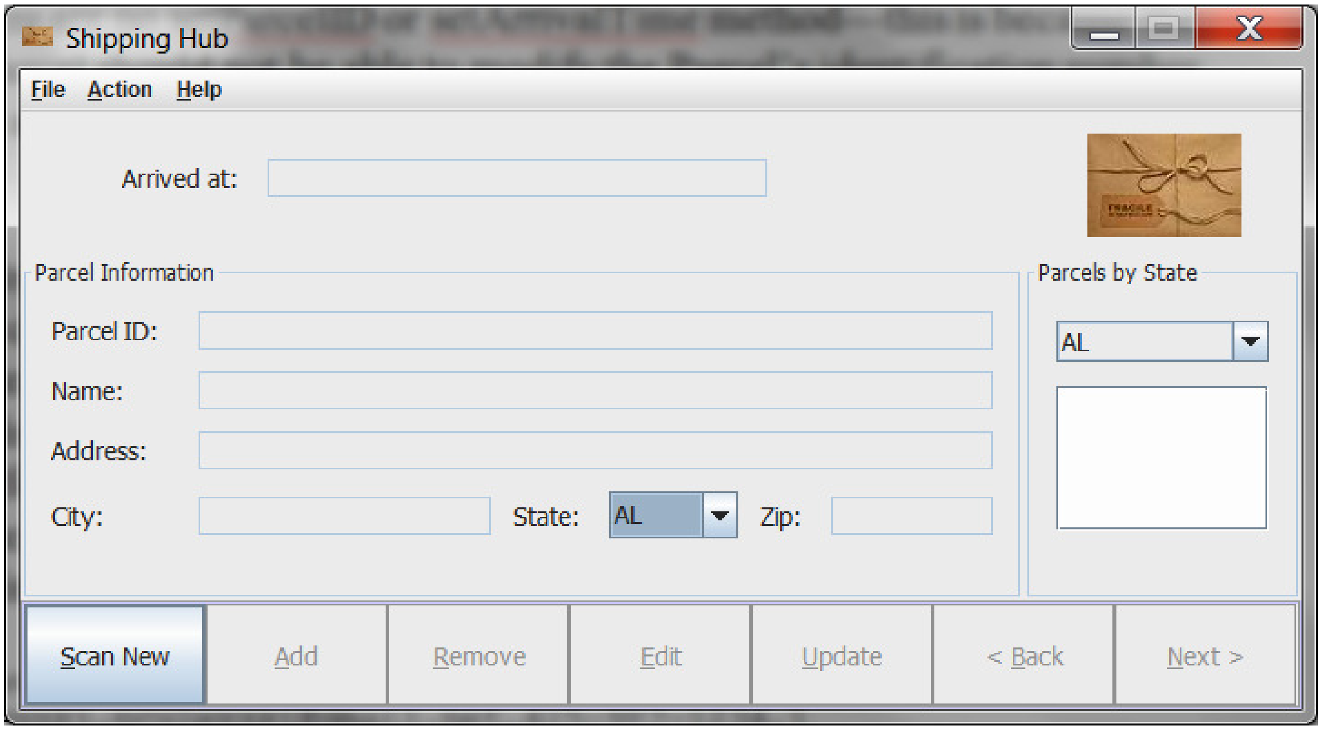
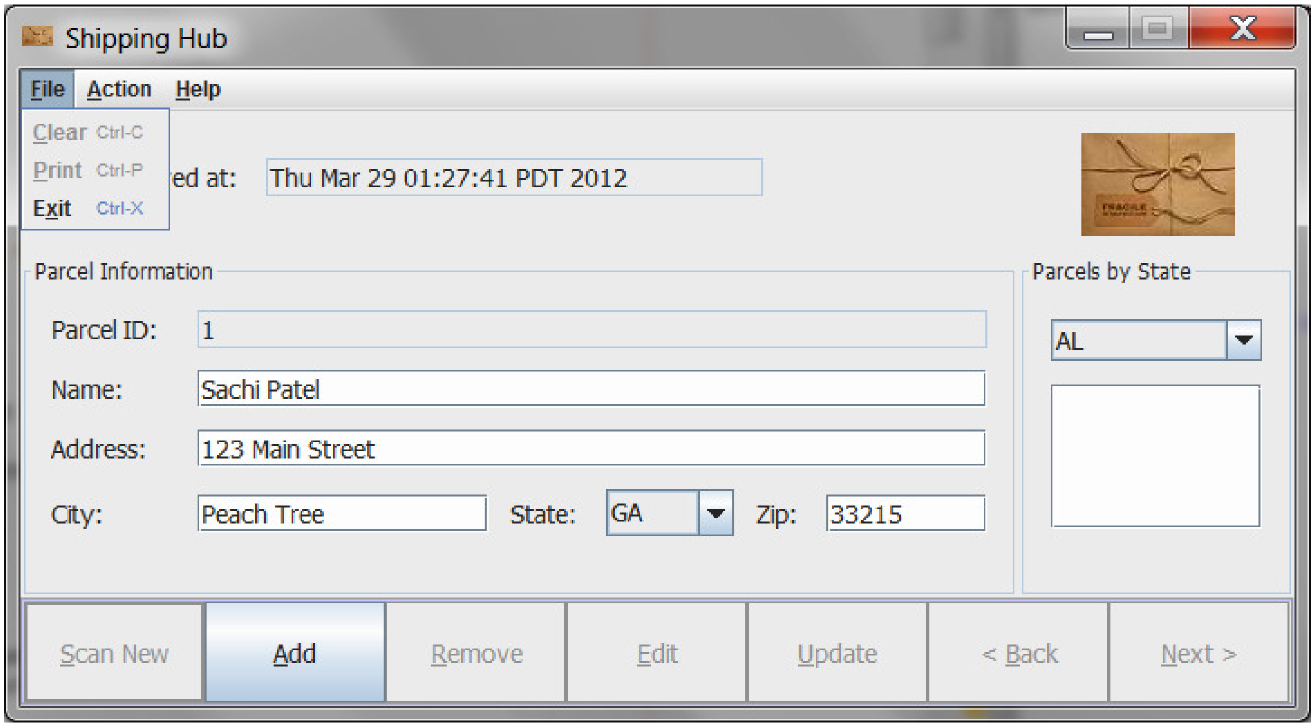


Figure 1. Initial screen (after splash screen)

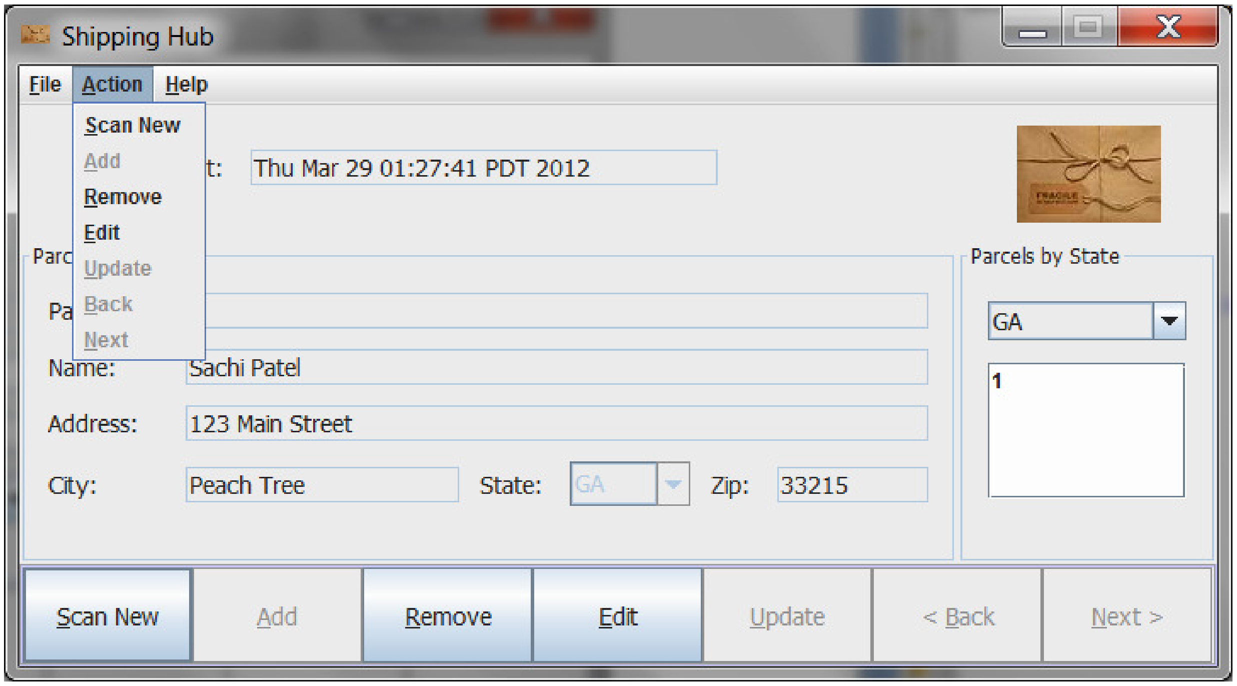


Thu Mar 29 01:27:41 PDT 2012

Parcel ID: 1
Name: Sachi Patel
Address: 123 Main Street
City: Peach Tree State: GA Zip: 33215

Figure 2. After Scan New is pressed and valid customer information is entered

Clicking the Add JButton sets the data to a Parcel object (instance of the ShippingHub GUI class) and adds that parcel to an ArrayList of shipped Parcels. The button disables itself, and it enables the Scan New, Remove, and Edit JButtons (with corresponding duplicate actions in the Action JMenu). This button also changes the selected item in the parcel by state JComboBox: you can see in the example screen shot below that all parcel by ID number are shown for GA since the parcel just added was shipped to GA. The navigational Back and Next JButtons are not enabled yet because there is only one parcel scanned so far. Adding more than one parcel should enable them however. You could use these navigational buttons to position a parcel that needs to be edited or deleted.



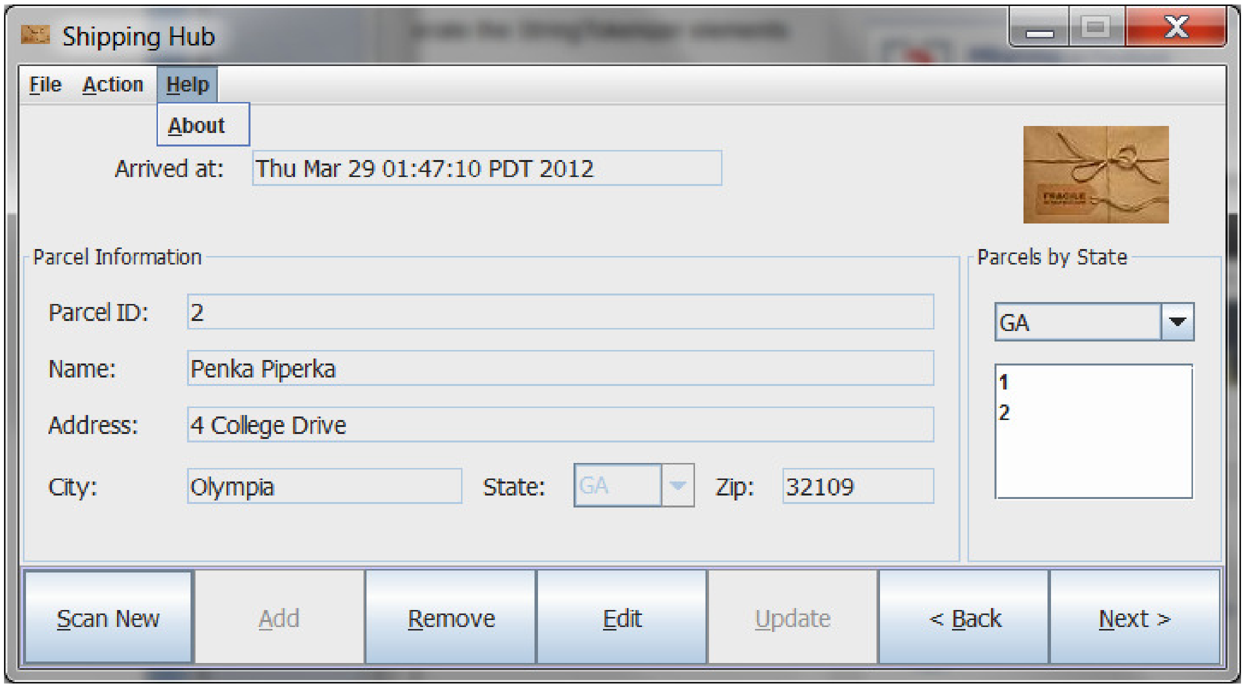
Thu Mar 29 01:27:41 PDT 2012

Name: Sachi Patel

Address: 123 Main Street

City: Peach Tree State: GA Zip: 33215

Figure 3. After Add is pressed and the very first parcel information is scanned



Arrived at: Thu Mar 29 01:47:10 PDT 2012

Parcel ID: 2

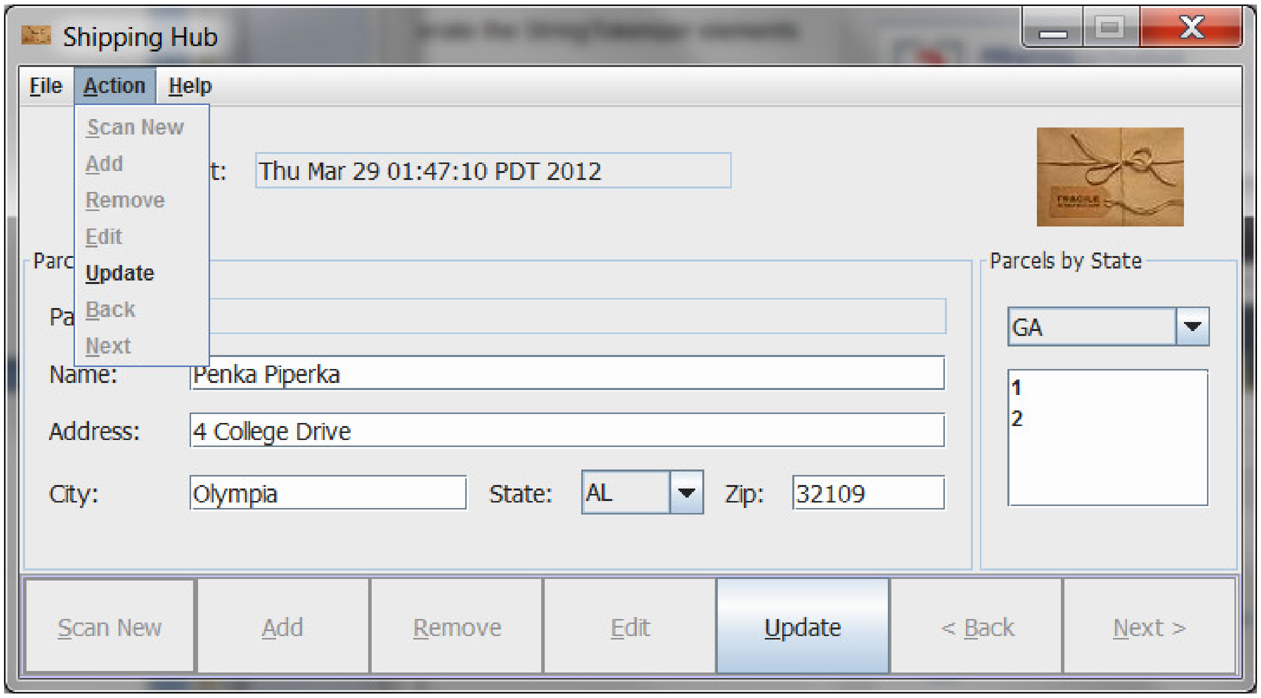
Name: Penka Piperka

Address: 4 College Drive

City: Olympia State: GA Zip: 32109

Figure 4. After a second parcel is scanned

The Edit JButtons enables the customer info fields (but not the parcel ID) for editing and disables all other JButtons. Once changes are made, pressing the Update button might bring a confirmation dialog (Save changes? yes/no). This button disables itself either way and if changes are made (see that the state on the second record in the screen shots below was changed from GA to AL with corresponding change to the parcel by state). All other JButons other then the Add are enabled when the Update JButton is clicked.



t: Thu Mar 29 01:47:10 PDT 2012

Name: Penka Piperka

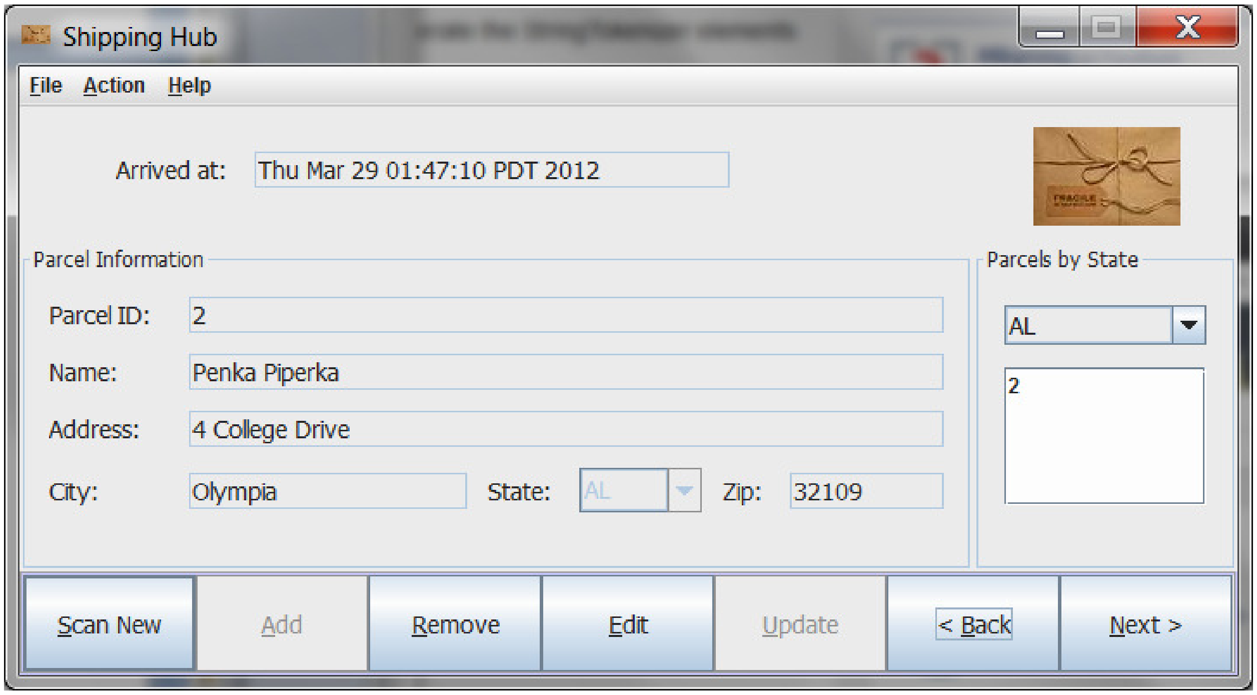
Address: 4 College Drive

City: Olympia State: AL Zip: 32109

GA

1
2

Figure 5. After Edit is pressed



Arrived at: Thu Mar 29 01:47:10 PDT 2012

Parcel ID: 2

Name: Penka Piperka

Address: 4 College Drive

City: Olympia State: AL Zip: 32109

AL

2

Figure 6. After Update is pressed

Your program must contain/do:

1. The project should start with a Splash Screen that closes itself after so many seconds and it should contain an About form activated from the Help menu.
2. A Parcel class that contains in order at least arrival time, parcel ID, name, address, city state, and zip code as instance variables, appropriate set and get methods, and at least two constructors (one of which should be the default constructor). Alternatively and better (extra credit) the Parcel class could contain an instance field from a Person class which will contain the name and address information with appropriate constructors and set/get methods).
3. Enable and disable of JButtons and JMenuItems as appropriate by calling methods to do so.
4. Ability to add new, modify existing ones, delete, and traverse through parcels.
5. A data structure of Parcels to hold the parcels data. This could be an array, ArrayList (preferable), vector, or any other structure you desire. I recommend that you have a second one for the parcels by state listing.
6. Javadocs, description of the program, and comments everywhere.
7. Menus that synchronize with corresponding buttons and with at least the following menu choices:
   - File with Clear, Print, and Exit menu items.
   - Action with Scan New, Add, Remove, Edit, Update, Back and Next (Next after last should lead to first and Back after first should take one to the last record) menu items.
   - Help with About menu item for an About form.
   - Menu choices are enabled/disables appropriately.
8. Read the states from an external, comma-delimited states text file and populate both comboboxes with the states read--this is required and not extra credit.
9. ToolTip text for the buttons at least, explaining their function.
10. Provide validation for at least zip code field. Consider using JFormattedTextField with appropriate masking for valid zip codes.
11. Make the Sate JCombobox not editable, but consider expanding it to include all 50 states.
12. Print the form for the currently shown parcel. Use the provided PrintUtilities class.
13. A free image of a parcel.

EXTRA CREDIT (Optional) There a number of ways to obtain extra credit for this project. Modify the application so that when the user double clicks a Parcel's ID in the parcelStateJList, that Parcel's information will be displayed in a Parcel Information JPanel. Saving the parcel processing with ability to retrieve the already scanned parcels can also harness you an extra point. Ability to search for a parcel by name or zip will also bring some extra cookies, especially if the search is a binary search (which will require sorting the parcel by that field). Adding additional classes, such as a Person class, will yield another extra cookie.

Project 1 Grading Guidelines: Your project will be graded on the following:

- Correct solution to the proposed problem.
- GUI design (as outlined above).

- Elegance, simplicity, style, and readability of the algorithms and code.
- Comments:
  - ✓ Use of header.
  - ✓ Description of project.
  - ✓ Description of procedures.
- 30 points maximum. Extra credit points are optional and not required.

Project 1 Recommendations:

The best advice I can give is to ***draw a detailed flowchart prior to coding this project***—it saves a lot of headaches later! Other than that, here are few hints:

⇒ Consider using a JListBox to list the parcel numbers for each state.

⇒ Selecting a member from the JListBox should display all the information about that member on the main form.

⇒ Include and follow Javadoc conventions. Make sure you generate beautiful Javadoc html files and include them in your project.

⇒ Make sure you synchronize the menu choices with the corresponding buttons.

⇒ Enable and disable buttons and menus with appropriate context but only ***after*** you solve the functionality of each button.

⇒ Check your scan parcel records against known results (see screen captures above and below).

⇒ Consider using two ArrayLists: one that contains Parcels entered by user and another for used to modify and display the Parcel objects for a specific state.

⇒ Provide icon and image for all form (but not for the Splash screen). Make sure the maximization of the forms is disabled and that it has one (Scan New) button as a default.

⇒ Make your About form nontrivial.

⇒ Use the provided PrintUtilities class to print the form.

⇒ Provide searchParcel() method that allows the user to locate particular parcel. Search fields could be the zip code or name of customer.

⇒ Make sure the main form centers as it starts and it is not resizable or maximizable.

⇒ Make the Scan New JButton default.

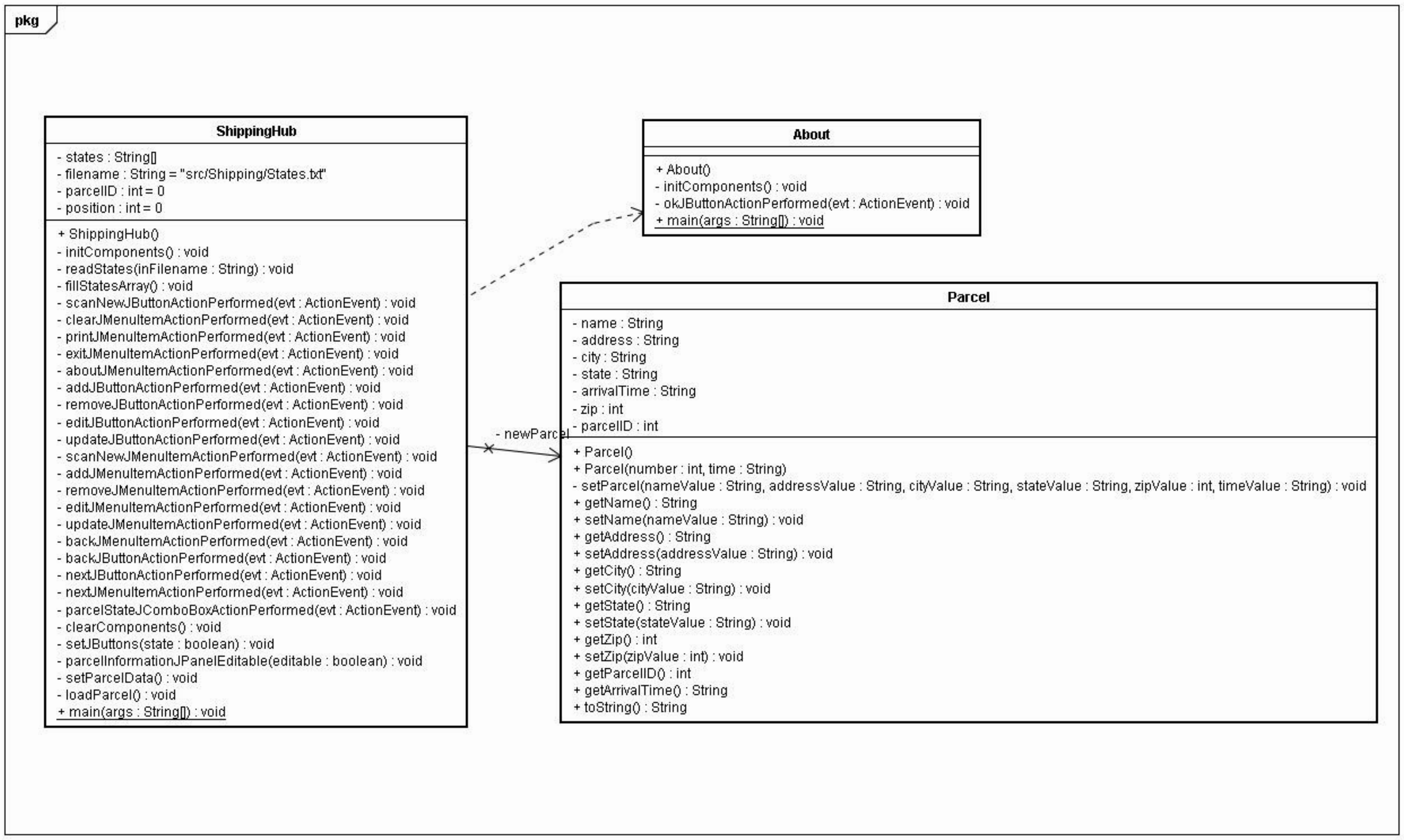
⇒ Removing a parcel need not change the sequence of parcel's Ids. In other words, if one scans 6 parcels and then deletes parcel #2, the new parcel to be scanned will still have ID 7.

⇒ Provide validation for adding a parcel--all fields are required--do not crash on empty fields.

⇒ Provide Tooltip at least for each JButton (and menus) giving directions and help to the user by describing their functionality or requirements for proper operation.

⇒ Design the GUI with JPanels added to the main form.

⇒ Create a Parcel class used by the main GUI class for your project. The following class diagram should help:


pkg
ShippingHub
- states : String[]
- filename : String = "src/Shipping/States.txt"
- parcelID : int = 0
- position : int = 0
+ ShippingHub()
- initComponents() : void
- readStates(inFilename : String) : void
- fillStatesArray() : void
- scanNewJButtonActionPerformed(evt : ActionEvent) : void
- clearJMenuItemActionPerformed(evt : ActionEvent) : void
- printJMenuItemActionPerformed(evt : ActionEvent) : void
- exitJMenuItemActionPerformed(evt : ActionEvent) : void
- aboutJMenuItemActionPerformed(evt : ActionEvent) : void
- addJButtonActionPerformed(evt : ActionEvent) : void
- removeJButtonActionPerformed(evt : ActionEvent) : void
- editJButtonActionPerformed(evt : ActionEvent) : void
- updateJButtonActionPerformed(evt : ActionEvent) : void
- scanNewJMenuItemActionPerformed(evt : ActionEvent) : void
- addJMenuItemActionPerformed(evt : ActionEvent) : void
- removeJMenuItemActionPerformed(evt : ActionEvent) : void
- editJMenuItemActionPerformed(evt : ActionEvent) : void
- updateJMenuItemActionPerformed(evt : ActionEvent) : void
- backJMenuItemActionPerformed(evt : ActionEvent) : void
- backJButtonActionPerformed(evt : ActionEvent) : void
- nextJButtonActionPerformed(evt : ActionEvent) : void
- nextJMenuItemActionPerformed(evt : ActionEvent) : void
- parcelStateJComboBoxActionPerformed(evt : ActionEvent) : void
- clearComponents() : void
- setJButtons(state : boolean) : void
- parcelInformationJPanelEditable(editable : boolean) : void
- setParcelData() : void
- loadParcel() : void
+ main(args : String[]) : void
About
+ About()
- initComponents() : void
- okJButtonActionPerformed(evt : ActionEvent) : void
+ main(args : String[]) : void
- newParcel
Parcel
- name : String
- address : String
- city : String
- state : String
- arrivalTime : String
- zip : int
- parcelID : int
+ Parcel()
+ Parcel(number : int, time : String)
- setParcel(nameValue : String, addressValue : String, cityValue : String, stateValue : String, zipValue : int, timeValue : String) : void
+ getName() : String
+ setName(nameValue : String) : void
+ getAddress() : String
+ setAddress(addressValue : String) : void
+ getCity() : String
+ setCity(cityValue : String) : void
+ getState() : String
+ setState(stateValue : String) : void
+ getZip() : int
+ setZip(zipValue : int) : void
+ getParcelID() : int
+ getArrivalTime() : String
+ toString() : String

You might find these partial comments worthwhile—each counts as a -1 point.

```
/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'Comments by the prof:
'Great effort.  Here are suggestions for improvement:
'1. Use JavaDoc comments throughout the program, not just at the top heading.
'2. An image of a package logo (or other) would enhance the look of the form.
'3. Did not implement printing of form--it is required.
'4. Make the Scan New button default--pressing the Enter should fire it.
'5. Make both comboboxes read-only.
'6. Validate states and zip entries.
'7. It is good style to tab code inside a method.
'8. Provide buttons for adding, updating, editing, exiting the application and navigating
about the scanned parcels.
'9. Declare all constants as finals if there are any.
'10. Display Arrived at date in read-only date text field.
'11. Change the tab sequence to indicate correct order of entry.
'12. Name the classes and project appropriately.
'13. The parcel class should contain at least two constructors, set and get methods and a
toString method.
'14. Follow the Java naming convention for naming variables, methods and classes.
'15. Read the states data from an external file or database.
'16. Include a separate Parcel class for the project.
'17. Check for valid state and zip--consider using JFormattedTextField for the zip.
'18. Update Parcels by state dynamically as parcels are scanned.
'19. Enable menu choices that synchronize with buttons' functionality.
'20. Provide a sound data structure for the Parcels.
'21. Missing menus.
'22. Disable maximization or resizing of the form.
'23. Add an About form which describes the project.
'24. Add a Splash screen that starts the project, displayed in the middle of the screen.
'25. Add icons to the form(s).
'26. Consider separate forms/classes for adding/editing a member.
'27. Provide ToolTip texts for at least each button to explain their functionality.
```

```
'28. Make changes to the database dynamic. For example, if a parcel is deleted, that
parcel should not show up in the JListBox in Parcels by state after being deleted.
'29. Add ability to search for a parcel in the database.
'30. Give title to form.
'31. Program crashes on empty or invalid input.
'32. Enabling and disabling of JButtons or menus is not correct.
'
'The ones that apply to your project are:
'
'4, 5, 12, 25
'26/30
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*/
```