Record Indexer Client

Contents

Introduction

For this project you will implement the Record Indexer Client program. This will require you to implement the Client GUI and integrate it with your Server to realize the system's functionality. Much of the Client's behavior is described in the document entitled Record Indexer Overview. Other details about the program's behavior can be ascertained by running the provided demo. Based on these sources you should already be familiar with the basic functionality of the program. This document provides additional details about the specific requirements for the project.

Code Organization

The source tree contains several sub-directories, two of which are named `src` and `test`. The `src` directory will contain all of the "real" Java code for your project. The `test` directory will contain all of the testing Java code for your project. The Client classes should be placed in packages such as `client`, `client.this`, `client.that`, etc. If there are classes that are used by both the Server and the Client, these could be placed in packages such as `shared`, `shared.this`, `shared.that`, etc.

Running Your Client

Your Client program should accept two command-line arguments which specify the host name and TCP port number where the Server is running. If the host name argument is the empty string, use "localhost" as the host name. If the port number argument is the empty string, use the same default port number used by your Server.

The TAs will use the ANT target named `client` to run your Client at pass off, as shown below.

```
ant client –Dhost=<HOST NAME> –Dport=<PORT NUMBER>
```

To make the `client` target work properly, you will need to modify the `client` target in your ANT `build.xml` file to specify the full package name of the Java class that implements your Client.

GUI Structure and Behavior

Your GUI should mimic the structure and layout of the demo GUI. This applies to the main Indexing Window as well as the various dialog windows.

The main Indexing Window should be resizable, but the dialog windows should not be.

All dialog windows should be modal (i.e., the user must close the dialog window before being allowed to interact with other parts of the GUI).

User Login

When no one is logged in, your program should display the Login Dialog. This includes both when the program initially starts, and when the user logs out (but doesn't exit). The Indexing Window should not be visible when no one is logged in.

If the user clicks the "Exit" button in the Login Dialog, the program should terminate.

If the user clicks the "Login" button, the program should pass the specified username and password to the Server for validation. If login is successful,

1. Display a Welcome Dialog that welcomes the user by name, and displays the number of records previously indexed (i.e., submitted) by the user.
2. When the user closes the Welcome Dialog, close the Login Dialog, and display the main Indexing Window. The state of the Indexing Window should be restored to the same state it was in the last time the user logged out or exited. (See the next section for details.)

If login fails,

1. Display an error message dialog indicating the problem.
2. When the user closes the error message dialog, the Login Dialog should remain visible so the user can try again.

Saving Per-User State

When a user logs out or exits the program, the current state of their work should be saved to disk. The next time the user logs in, the state of their work should be restored from disk. Program state should be saved on a per-user basis, since over time there may be multiple users that use the program on the same machine. In order to prevent users from interfering with each other, each user's state should be saved separately.

The following information should be saved at logout/exit and restored at login:

Batch State

1. Batch image
2. Record field values

Window State

1. Position of the Indexing Window on the desktop
2. Width and height of the Indexing Window
3. Positions of the horizontal and vertical split panel dividers

Image State

1. Zoom level

2. Scroll position
3. Highlights visible setting
4. Image inverted setting

File Menu

The “Download Batch” menu item should only be enabled when the user is not currently working on a batch. When the user selects the “Download Batch” menu item, the program should display the Download Batch Dialog.

The “Logout” menu item should always be enabled. When the user selects the “Logout” menu item, the program should:

1. Save the state of the user’s work on disk
2. Close the Indexing Window
3. Display the Login Dialog

The “Exit” menu item should always be enabled. When the user selects the “Exit” menu item, the program should:

1. Save the state of the user’s work on disk
2. Terminate the program

Button Bar

The “Zoom In” and “Zoom Out” buttons should adjust the scale (i.e., zoom level) of the image. Zooming should be done relative to the center point of the currently visible area. That is, as zooming occurs, the point at the center of the view should remain fixed.

The “Invert Image” button should turn image inversion on and off.

The “Toggle Highlights” button should turn field highlights on and off.

The “Save” button should save the state of the user’s work to disk. (See the section entitled *Saving Per-User State* for details.)

The “Submit” button should submit the current batch to the Server, and put the Indexing Window in an empty state. When a batch is submitted, all record field values entered by the user should be sent to the Server. The Server should save the indexed values, and increment the number of records indexed by the user.

Image Panel

If no one is logged in, or no batch is currently being indexed by the logged in user, the Image Panel should be empty.

The Image Panel should highlight the currently-selected record field (if highlights are turned on).

The Image Panel should invert the image (if image inversion is turned on).

The user should be able to select a record field by clicking on it with their mouse.

The user should be able to move the image in the panel by dragging it with their mouse.

The user should be able to zoom the image in and out with their mouse scroll wheel (just like the “Zoom In” and “Zoom Out” buttons in the button bar). Zooming should be done relative to the center point of the currently visible area. That is, as zooming occurs, the point at the center of the view should remain fixed.

Table Entry Tab

If no one is logged in, or no batch is currently being indexed by the logged in user, the Table Entry Tab should be empty.

When a batch is being indexed, the table should contain a “Record Number” column that displays record numbers and is read-only. The table should also contain editable columns for all of the project fields in the proper order.

When the screen space allocated to the Table Entry Tab is too small to fully display the table, scroll bars should be provided so the user can scroll the view.

The TAB key should move the field selection in a left-to-right, top-to-bottom order.

Unrecognized field values should be highlighted red.

Right-clicking on an unrecognized field value (i.e., one that is highlighted red) should bring up a context menu containing a “See Suggestions” menu item. Selecting the “See Suggestions” menu item should display the Suggestions Dialog. If the user selects a suggested value and clicks the “Use Suggestion” button, the selected value should replace the unrecognized value in the table.

Form Entry Tab

If no one is logged in, or no batch is currently being indexed by the logged in user, the Form Entry Tab should be empty.

When a batch is being indexed, the left side of the Form Entry Tab should contain a list of record numbers that allows the user to select the current record. The right side of the Form Entry Tab should contain labeled text boxes for all of the project fields in the proper order. These text fields should be editable.

When the screen space allocated to the Form Entry Tab is too small to fully display the record number list or the form fields, scroll bars should be provided so the user can scroll the view.

The TAB key should move the field selection in a top-to-bottom order.

Unrecognized field values should be highlighted red.

Right-clicking on an unrecognized field value (that is highlighted red) should bring up a context menu containing a “See Suggestions” menu item. Selecting the “See Suggestions” menu item should display the Suggestions Dialog. If the user selects a suggested value and clicks the “Use Suggestion” button, the selected value should replace the unrecognized value in the form.

Synchronization of Image Panel, Table Entry Tab, and Form Entry Tab

The currently-selected record field is displayed in three views: the Image Panel, the Table Entry Tab, and the Form Entry Tab. All of these views allow the user to move the current selection from one field to another. Regardless of how the current selection is changed, all three views must be kept in synch with each other. For example, if the user clicks on a record field in the Image Panel, thus changing the current selection, the current selection in the Table Entry Tab and Form Entry Tab must be updated to show the new selection. Similarly, if the user selects a different cell in the Table Entry Tab, the Image Panel and Form Entry Tab must be updated to show the new selection. The same is true if the current selection is changed through the Form Entry Tab.

Field Help Tab

The Field Help Tab should always display the field help for the currently-selected record field. If no batch is currently being indexed, the Field Help Tab should be empty.

If the screen space allocated to the Field Help Tab is too small to entirely display the field help contents, scroll bars should be provided so the user can scroll the view.

Image Navigation Tab

The Image Navigation Tab gives the user a birds-eye-view of the batch image, and draws a gray rectangle over the portion of the batch image that is currently visible in the Image Panel. This is most useful when the batch image is only partially visible in the Image Panel, in which case the Image Navigation Tab gives the user a better idea of what part of the batch image they are looking at in the Image Panel.

The user can also scroll the Image Panel by dragging the gray rectangle in the Image Navigation Tab. This allows the user to precisely select which part of the batch image they are looking at in the Image Panel.

The Image Navigation Tab should always display the entire batch image (scaled as needed to keep it visible within the space available). The aspect-ratio (i.e., width-to-height ratio) of the batch image should always be preserved as the image is scaled.

If no batch is currently being indexed, the Image Navigation Tab should be empty.

Synchronization of Image Panel and Image Navigation Tab

The Image Panel and the Image Navigation Tab must always be kept in synch with each other. The gray rectangle in the Image Navigation Tab must always accurately reflect the part of the batch image that is visible in the Image Panel. Any of the following events will require the Image Navigation Tab to be updated:

- Image Panel is resized
- User zooms in or out in the Image Panel
- User scrolls the Image Panel
- Image Navigation Tab is resized

Quality Checker

When the user selects the "See Suggestions" menu item either in the Table Entry Tab or Form Entry Tab, the Suggestions Dialog should be displayed.

The Suggestions Dialog should display all known field values that are edit distance one or two from the value provided by the user. (Refer to the Spell Corrector project for information on how to calculate edit distance.)

The list of suggested values should display a scroll bar if it is too long to fit in the screen space allocated to the list.

Quality Checker Unit Testing

When you implement the Quality Checker feature, you will need to write code to:

1. Check to see if a user-provided field value is a "known value" or not
2. Given an unknown user-provided field value, calculate the suggested values to display to the user (i.e., known values that are edit distance one or two from the user-provided value)

Write automated unit tests to verify that this code is working properly. Add your tests to the `client.ClientUnitTests` class, or create additional test classes of your own. If you add more unit test classes, add their names to the `testClasses` array in the `client.ClientUnitTests` `main` method. The TAs will run your unit tests by running the following ANT command:

```
ant client-tests
```

When you are done, all of your unit tests must run successfully (i.e., no failures).

Source Code Evaluation

After you pass off your project, your source code will be graded by a TA on how well you followed the good programming practices discussed in class. The following criteria will be used to evaluate your source code:

- (20%) Effective class, method, and variable names
- (20%) Effective decomposition of classes and methods
- (20%) Code layout is readable and consistent
- (20%) Effective organization of classes into Java packages
- (20%) High-quality unit test cases implemented with JUnit