Final Report

JavaPEG

Version 2.5 (Beta)

Content

The Original Design

1. <u>Program Description:</u>

JavaPEG has the possibility to change the file names of jpeg images that contains Exif data, supported by many Digital Cameras for example Canon, Fujifilm, HP, Kodak, Nikon etc. The software supports creating thumb nails overview and each of which are linked with full sized image. It lets the user create image lists that are slide shows and the software has the functionality to export those image lists to the famous image viewers like irfanviewer and polyviewer. The image viewer is embedded within the software which has all the basic functionalities for navigations. It shows the metadata information (date, time, cameraModel, shutterSpeed, isoValue, apertureValues etc) about displayed and selected information. The software supports three languages English, Spanish and Swedish.

The *Rename Image* function works in simple five steps by selecting path to images and selecting destination folder and starting the process. The *View Image* works somewhat similar and thereby we can save the image list and export it to other viewers. The source code contains packages for each of these functions and respective methods. The software can be started by running the StartJavaPEG.java file in Eclipse or any other software. The program calls MainGUI class which creates the model of GUI and builds basic outlines and tabs of the user interfaces of software. This program navigates to all other classes and packages which user calls from GUI. JavaPEG is an open source software written only in Java programming language which we can run in any kind of operating system for example: Windows, Mac or Linux. The project can be found at the following link http://sourceforge.net/projects/javapeg/files/

2. <u>Architectural styles:</u>

JavaPEG has been implemented in Java and follows the object oriented concepts of this language. Object oriented architectural style is one of the main style in program. JavaPEG confirms the rules of OO architectural style as it contained the design paradigm which is based on different responsibilities. The application is divided in several objects which are reusable and self-efficient. The code for the extension could be reused to apply to another application by editing the parameter values.

Also it uses general architectural styles such as component-based, blackboard, pipe and filters styles. The software has various components and packages which are related to different functionalities which users can use. Those are implemented using component-based architectural style. The project is divided in several components like rename, validate, language which are actually reusable functional and logical components. They do communicate with each other with well-defined interfaces.

The project does not use client server architectural style as it does not communicate with any other server or any other software. And henceforth we can say that, it doesn’t have multi-tier architectures or peer to peer model. This even signifies that the software does not implement layered architectural styles.

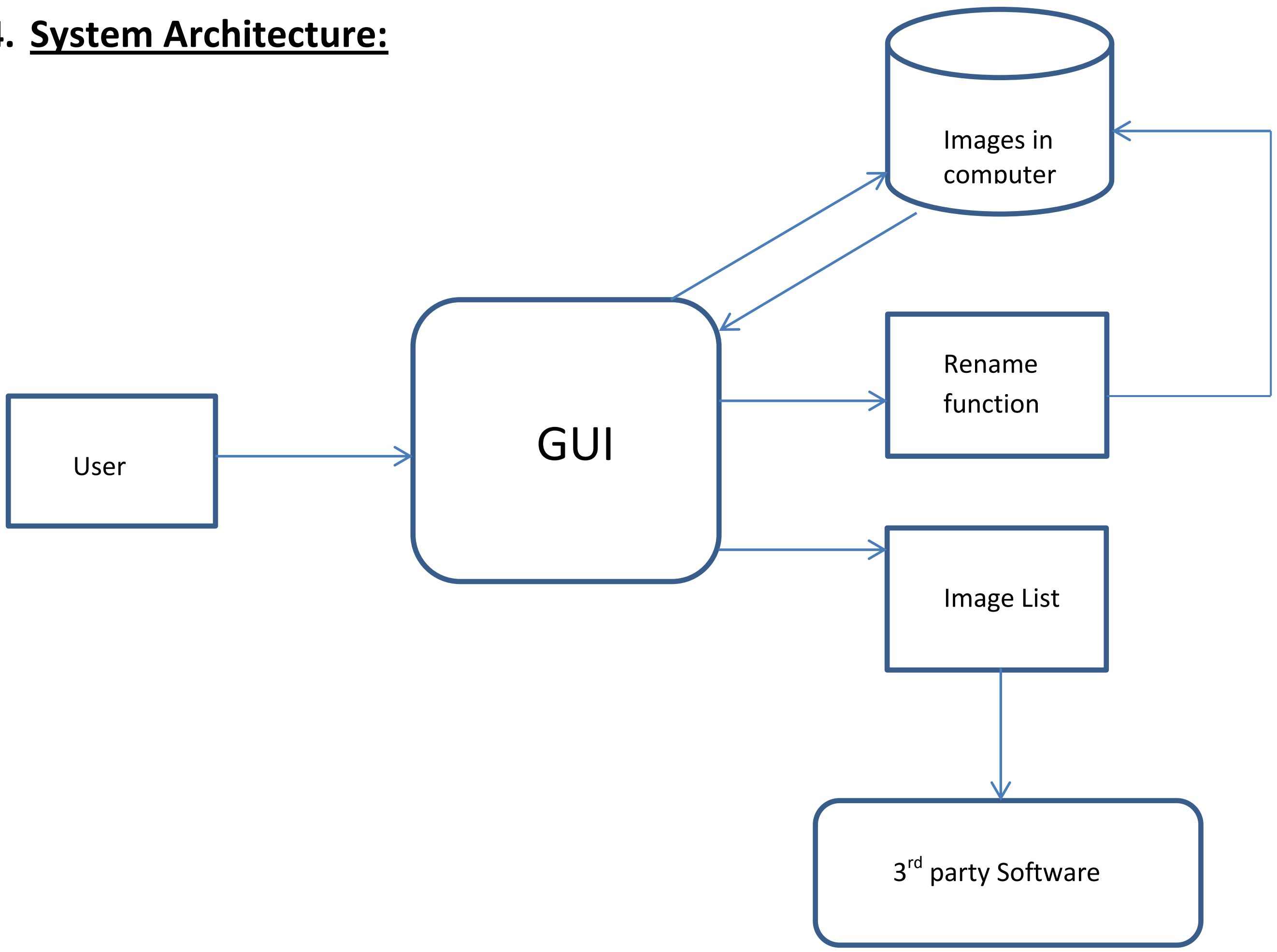
3. <u>Design patterns:</u>

The software project use Strategy Design Pattern and it can be noticed by looking at the code and general flow of the project. In the code of MainGUI file, the classes implement various views for example, ActionListener, ComponentAdapter and MouseAdapter etc. The project also contains various models and the programs which controls them.

There are three categories in design patterns for any kind of project which contains several patterns: Creational Patterns, Structural Patterns and Behavioral Patterns. JavaPEG follows builder pattern as a type of creational pattern as the builds complex panels and performs complex function from simple ones step by step. The rename function is the biggest function and the program contains several method, packages to implement it. Adapter pattern can also be seen in the project as there are classes which inherit properties of other classes and some of them implements and override them.

There are three types of patterns seen in Behavioral patterns. Command pattern is the pattern which accept user's request and display the proper view and perform proper action. In the extension the command pattern will get affected as user can request for 2 more views in thumbNailPanel**.** The iterator pattern is another pattern seen in behavioral category which allows navigating through collection of data using a common interface without knowing about the underlying implementation.

4. System Architecture:

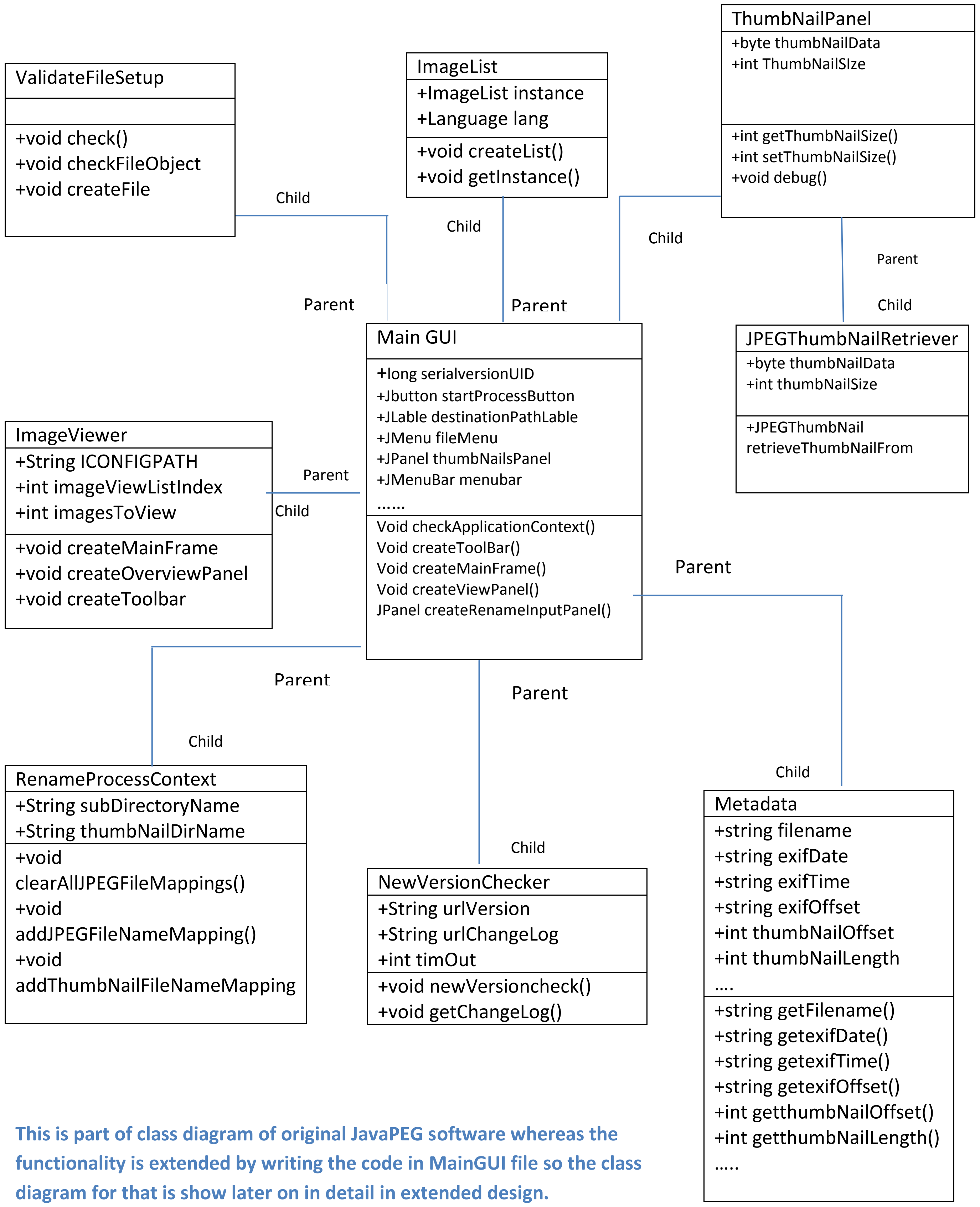


(No change after extension)

The system architecture is basically a service oriented architecture where services are provided to user as per user's requests. The user can see GUI of software and he/she can instruct software to perform it's functionality. There is and image explorer in JavaPEG software where we can explore images saved in the software. The selected images are viewed in the ThumbNailPanel. The extension of the system suggests improving the views in ThumbNailPanel so it's just a different look in GUI, however, the system's architecture won't be affected because of this modification.

The system supports creation of image lists which facilitates user to form the image lists by clicking on the proper option provided in GUI. Those image lists can be exported to 3rd party softwares like IrfanView or PloyView. The program also supports renaming the images at one go. The renamed images are saved in computer with predefined attribute names and at predefined location.

5. JavaPEG Data Module-Class Diagram(Original version)



This is part of class diagram of original JavaPEG software whereas the functionality is extended by writing the code in MainGUI file so the class diagram for that is show later on in detail in extended design.

Explanation:-

The diagram above is the general class diagram of JavaPEG. The software starts by running main method in StartJava() file and it calls MainGUI class. This is the main class which calls all the other classes and their including methods as per user's request. There are several packages in which all these classes are divided. The selected files are first validated in ValidateSetup class before performing any type of action.

The ImageList class is for making image lists and image viewer class is the biggest child class among all. Image viewer class can perform viewing of the images in another window of software. NewversionChecker class is used for ckecking any updates available. Metadata class is for viewing the metadata information of selected file. ThumbNailPanel class is responsible for generating the thumbnail panel in main frame. The extension idea lies here. JPEGThumbNailRetriever retrieves the selected thumbnail panel according to the option selected in the parent class thumbNailPanel. The extension is done in MainGUI file and details are reported later on.

The Extended Design

1. Changes needed and done:

There is the big change from the anticipated design and actual implementation. First I thought of making changes in *thumbNailPanel* and *thumbNailRetriever* class which later on found to be not useful. I tried to edit the code in MainGUI file only and that worked. Some of the important changes, modifications and strategies are reported as follows:

Problem 1 - Most of the time in implentation passed in figuring out where the code for particular function lies. The simplest approach I adopted was to look for a simple string
IMAGES IN SELECTED DIRECTORY
which is present at the top of *thumbNailPanel*. I tried several methods to find the string but couldn't able to find it in any .java file. The strings on the GUI are generally stored in the .xml file so that to change them later on but this string wasn't present on any .xml file.
I realized that the program support 3 languages so they cannot hard code this string so it may be present in 3 languages. So, I went to resources folder in JavaPEG software and headed to English language. There are 4 files present there which are of .en type. From them, *javapeg.en* contains the string for which I was looking for. And it was stored in *picture.panel.picturelabel* variable.

A Catch:-

The following piece of code illustrate the presence of string which made the further coding simpler.

```
JLabel thumbNailsTitleLable = new
JLabel(lang.get("picture.panel.pictureLabel"));
          …
thumbNailsBackgroundsPanel.add(thumbNailsTitleLable,
BorderLayout.NORTH);
```

So that string was placed at NORTH of *thumbNailPanel*.

Modifications in code is as follows:-

- The following code creates new Panel named as *compsToExperiment* which has a layout of GridLayout. Where we are going to add 3 buttons for the extension functionality.

```
final JPanel compsToExperiment = new JPanel();
compsToExperiment.setLayout(new GridLayout(1,3));
```

- The following code is then added to create 3 buttons on the panel at SOUTH.

```
JButton button1 = new JButton("Large Thumbnails");
final MainGUI temp = this;
button1.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e)
{
     selectLayout = 3;
     if(completePath!=null)
          loadThumbNails(new File(completePath));
     thumbNailsPanel.invalidate();
     thumbNailsPanel.repaint();
     thumbNailsPanel.updateUI();
}
});
```

The above code is written for adding a button named Large Thumbnails. And for each button an action listener is added which performs some action when that particular button is clicked. The *actionPerformed* method is written which selects the Layout as 3. 3 is the number of columns in each row which I considered at start but it doesn't signifies anything and it's for the internal operation only. Then method calls *loadThumbNails* method which loads the thumbnails (photos) in the t*humNailPanel* for this option.

If *completePath* is Null that means no directory is selected at that time even by clicking on any of 3 buttons no action will be performed.

- Similar code is written for button2 and button3 which identifies "Small Thumbnails" and "List" respectively.

- The following code will add 3 buttons in the newly created *compsToExperiment* panel. The position of those buttons will be kept at SOUTH for convenience.

```
compsToExperiment.add(button1);
compsToExperiment.add(button2);
compsToExperiment.add(button3);
thumbNailsBackgroundsPanel.add(compsToExperiment,BorderLayout.SOUTH);
```

- Now let's look at *loadThumbnails* method.

  This method loads the images in the center of *thumbNailPanel*. Each image is a separate button known as *thumbContainer* and the image gets loaded in *thumbContainer*. The significant modification is done as follows:-

```
JButton thumbContainer;
if(selectLayout != 1)
{
     thumbContainer = new JButton();
     thumbContainer.setIcon(new ImageIcon(tn.getThumbNailData()));
}
else
     thumbContainer = new JButton(jpegFile.getName());
```

  If *selectLayout* is not 1 means if the option selected is not of LIST then the photos will be loaded in the center of the panel. And if it is 1 then *thumbContainer* will have a button where Name of the file will be displayed and that too in list format.

```
int width = 0 ;
if(selectLayout != 1)
          width = thumbContainer.getIcon().getIconWidth();
if (width > iconWidth)
{
     iconWidth = width;
}
items++;
     ......

addThumbnail(thumbContainer, items);
```

  The width of t*humbContainer* was set in the above code. The variable "item" keeps track of the total images loaded till current time and gets incremented after each one. Later on, "*addThumbnail*" method is called by passing the button *thumbContainer* and items parameter.

- Now let's see *addThumbnail* method.

```
private void addThumbnail(JButton thumbNail, int items)
{
int columns = thumbNailsPanel.getVisibleRect().width ;
columns = (int)Math.floor( columns/100.0 );
if(selectLayout == 6)
      columns = columns;
else if(selectLayout == 3)
      columns = columns/2;
else
      columns = 1;
int width = (thumbNailsPanel.getVisibleRect().width - 22) / columns;

GridBagConstraints c = new GridBagConstraints();
c.fill = GridBagConstraints.HORIZONTAL;
c.weightx = 1.0;
c.weighty = 1.0;
c.gridx = items%columns;

int height = selectLayout==1?22:width;

thumbNail.setPreferredSize(new Dimension(width, height));
thumbNailsPanel.add(thumbNail, c);
}
```

In the above code the thumbnail is created and added in the *thumbNailPanel* in which *loadThumbnails* loads the images. The column variable stores the number of columns that should be present on each row of images in any layout.

By calculating manually the pixels the width of the panel comes out to be just above 600. And 6 columns were ideal for Small Thumbnails options. So for layout 6 which is actually Small Thumbnails column number remains the same that is 6. For Large Thumbnail option column numbers are divided by 2 so large images could be seen. For List option column number is set to 1 as there is a list format so only 1 column should be present of names of images.

The Width variable stores the width of each image in each layout option. 22 is the pixel size of the vertical scroll bar present on right so 22 is reduced and its divided by columns to get width at each time.

GridBagLayout is used to display the thumbnails in proper format. And we want to display the images graduating horizontally so constraint is set to Horizontal.

Height of each thumbnail is kept same as width of thumbnail except for the List option where it's kept as 22 which are equal to the size of scrollbar.

Finally the thumbnail means one image is added in *thumbNailsPanel* and the loop continues for the number of images in the buffer.

- The following code will add component listener in the list of listeners in *thumbNailsBackgroundsPanel.*

```
public void addListeners(){
     …..
thumbNailsBackgroundsPanel.addComponentListener(new
ComponentListener());
     …..
}
```

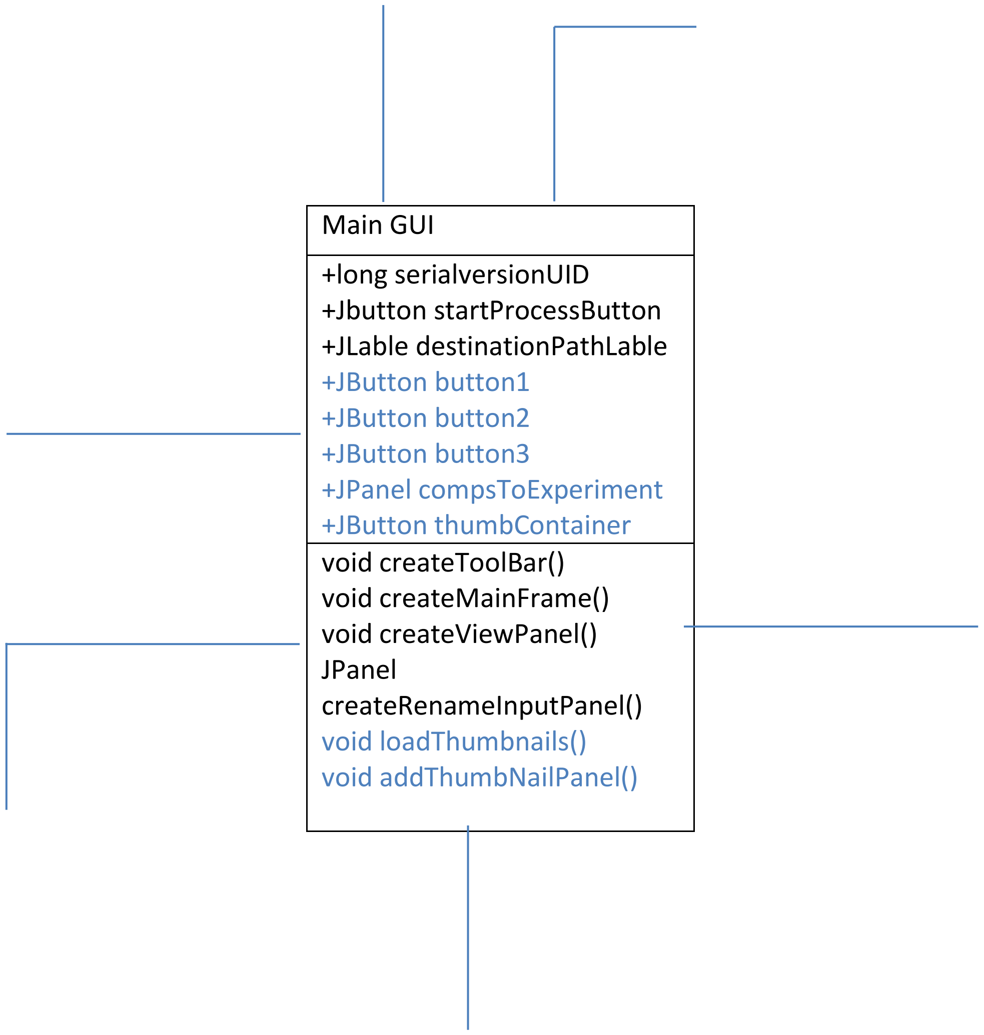
On each selection the images load again and they get resized according choice selected. The following code works for that;

```
private class ComponentListener extends ComponentAdapter {
@Override
public void componentResized(ComponentEvent e) {
     System.out.println(thumbNailsPanel.getVisibleRect().width);
}
     if(completePath!=null)
          loadThumbNails(new File(completePath));
     thumbNailsPanel.invalidate();
     thumbNailsPanel.repaint();
     thumbNailsPanel.updateUI();
}
```

Problem 2:-
GridLayout was the default Layout used by the software to display the images but it didn't work for the extension. Several time spent on how to display images with GridLayout. At last *GridBagLayout* is another layout which worked fine. In that, you can specify number of columns and rows to display images where as in GridLayout it was jumping to next row automatically when width ends and that caused the problem while resizing the panel window.

Problem 3:-
The listeners cause lot of problems in understanding the flow of project. Listeners are the procedures which are called when user click a button or do something with GUI. So, there is no serial flow of the program due to involvement of listeners. Listeners could execute anytime at dynamically while user is operating the software.

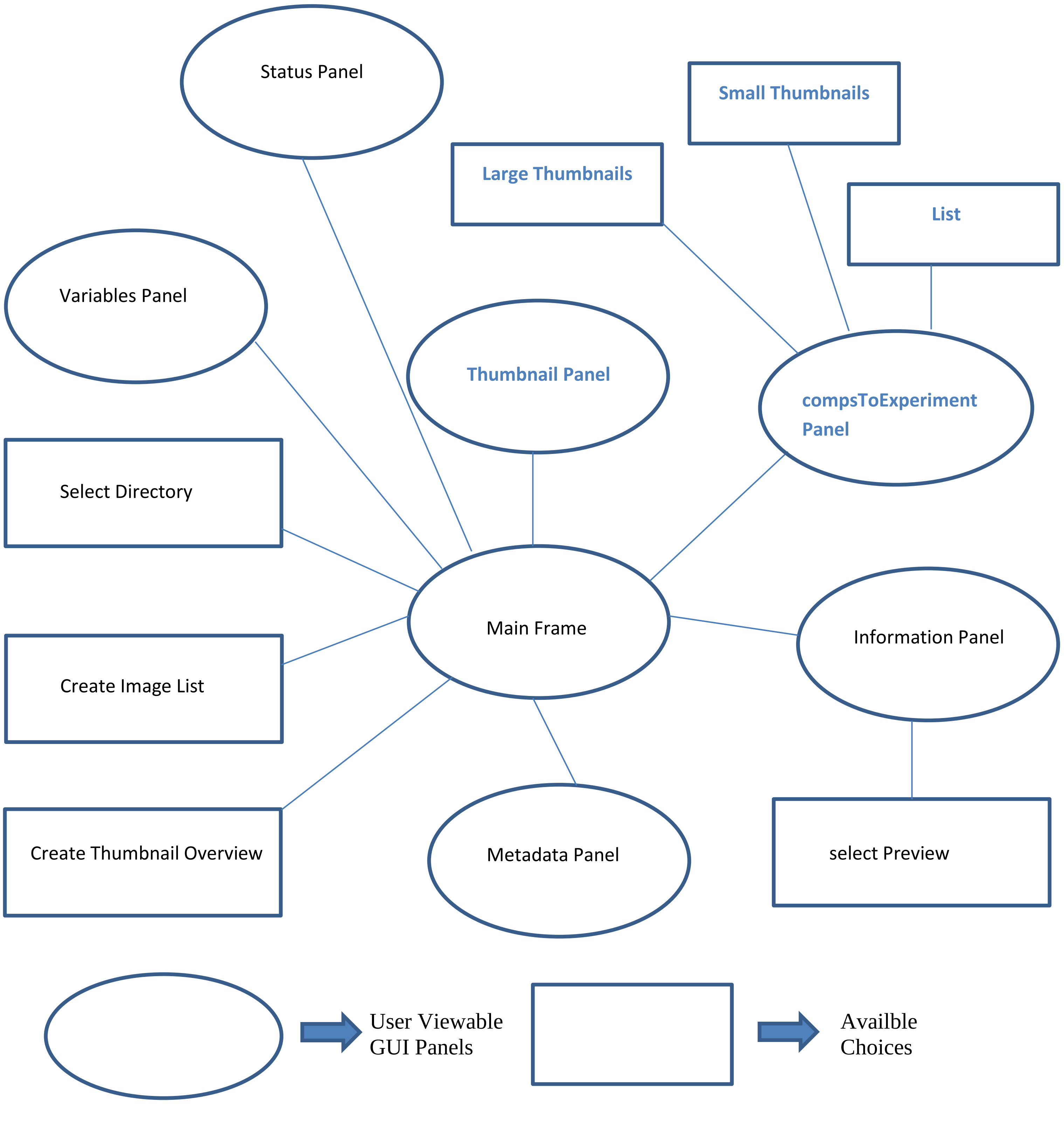
2. <u>Changes in Class Diagram:</u>



New methods are added in MainGUI class and new variables are also created as shown above.

(The changes in class diagram are marked in blue).

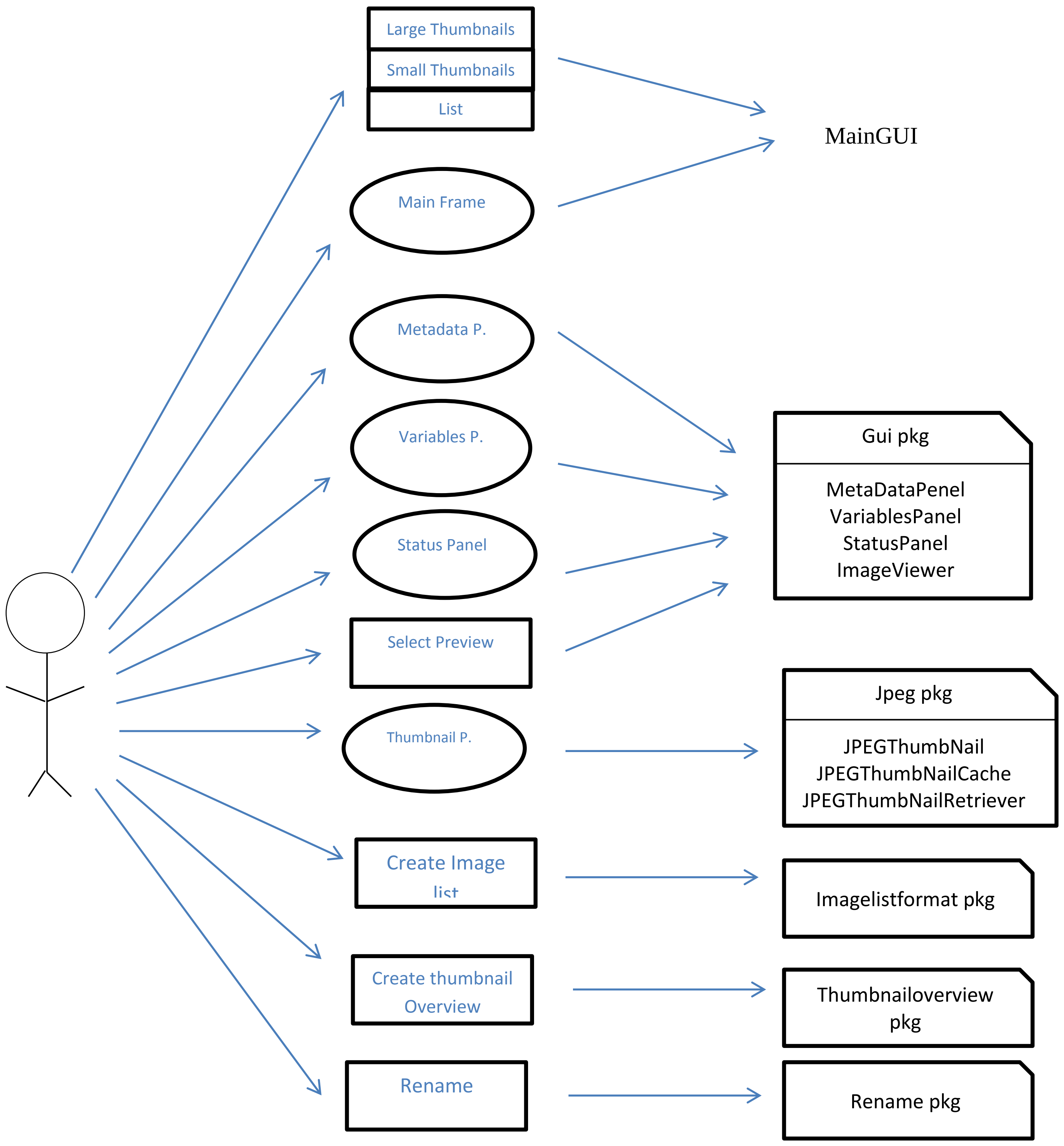
3. Display Module



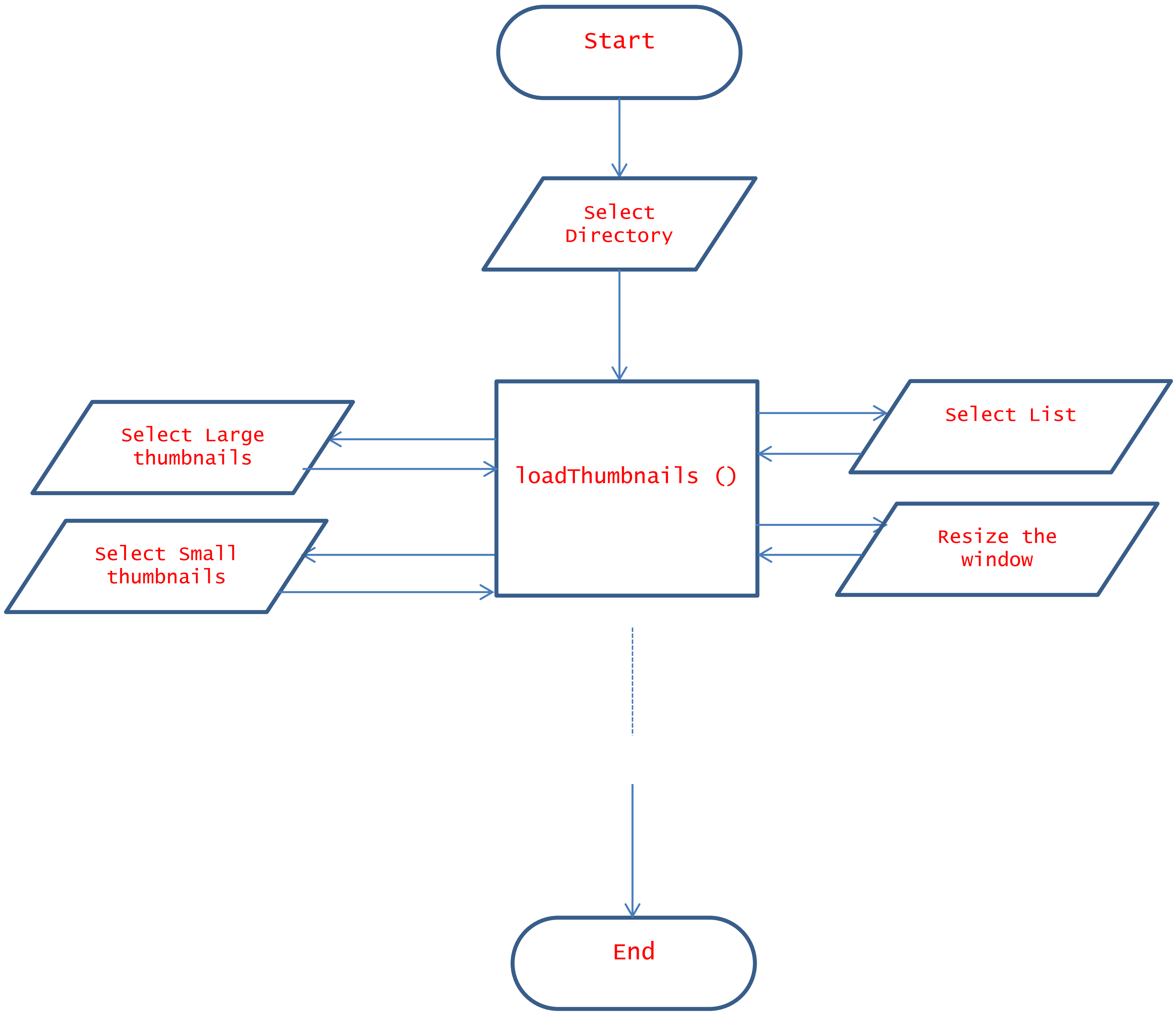
(Modifications in Display module are shown in Blue font)

The Display module is the user interface which can be viewable to user providing certain choices to select. All panels are created over Main Frame. *thumbNailPanel* has images to show which changes automatically when user selects one of the option from available choice of buttons below. Over the main frame and at the south of *thumbNailPane*l there is new panel called *compsToExperiment* panel on which three buttons resides which call changes in *thumbNailPanel*. The three buttons provide the choice for Large Thumbnails, Small Thumbnails and List option.

4. User Interface diagram.

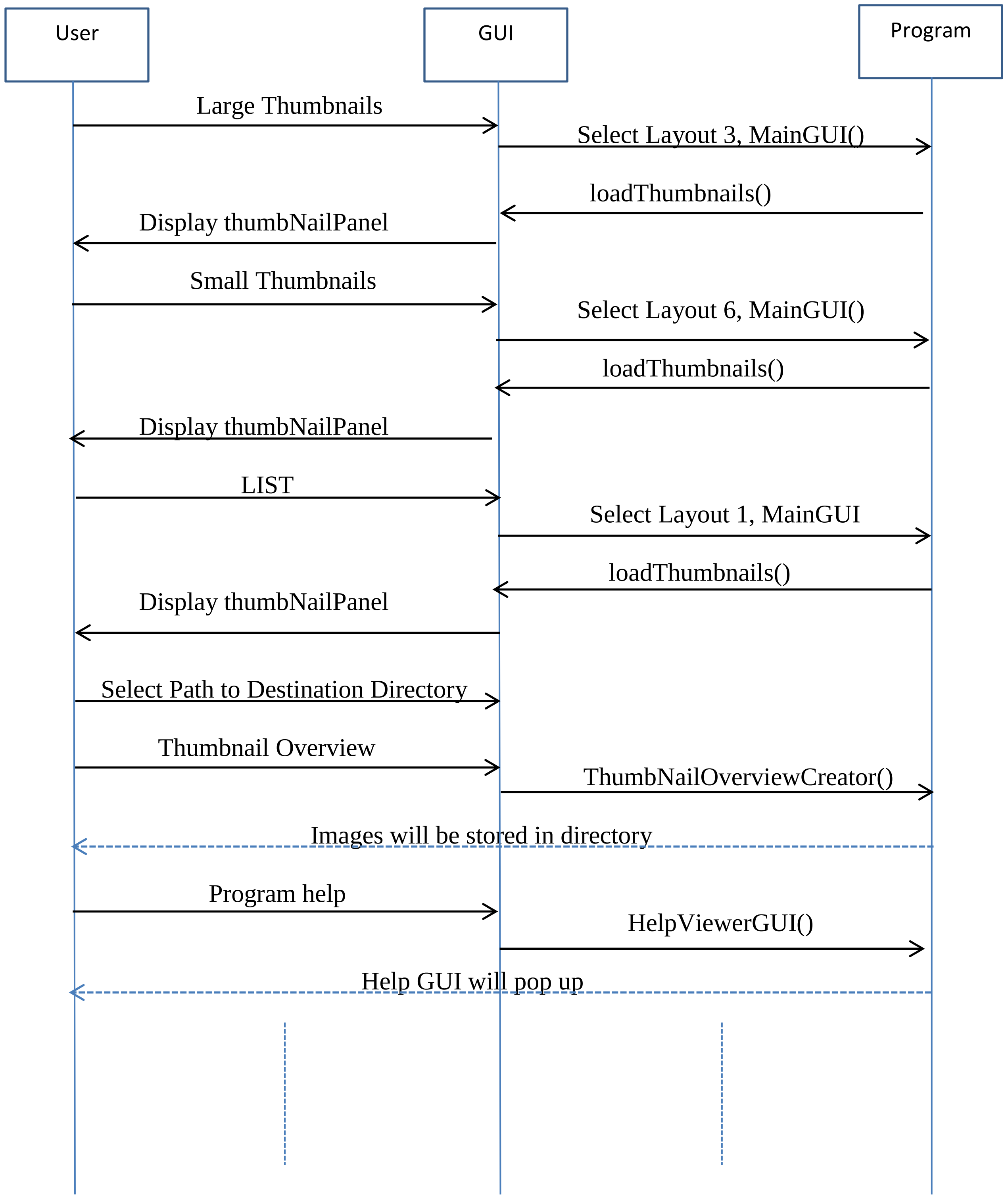


5. Flowchart for new feature:-



The flowchart is shown only for extended feature. It is noticeable that loadThumbnails() method will be called several times that after selection on choices and after resizing the window panel. Other functions of the software are not shown in the flowchart.

6. **Message sequence diagram**



The above message sequence diagram is self-explanatory. Messages will be passed from user to GUI and GUI will call appropriate class to execute that function and it will be displayed to use

7. <u>Evaluation:</u>

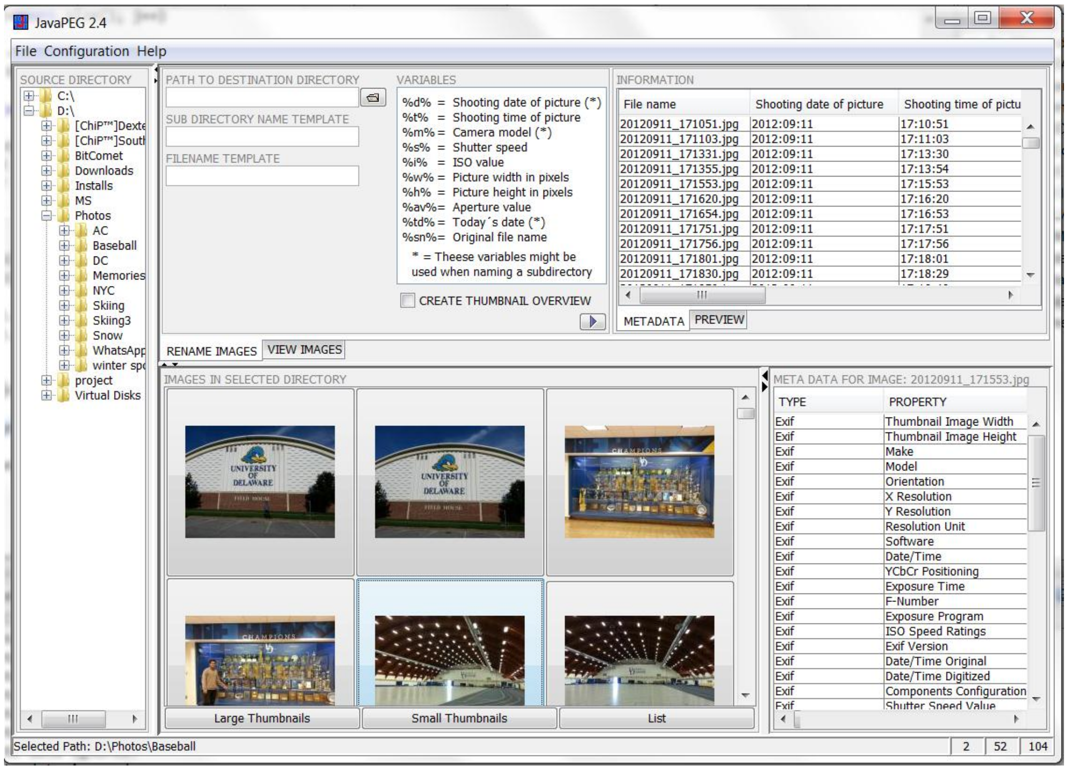
There are not many changes in the new system design. Only one class of MainGUI is edited and the changes occurred in particular sections only. There were no bugs found in the old systems except errors in the test cases. So, building new functionality got easier.

The JavaPEG software could not be evaluated against any other paid software in market for the reason that it was designed for simplicity and user friendliness. JavaPEG is goal concentric software and anyone can edit the code according to his requirements. There are no errors found till now in the software. Also, there is no integrity violation after renaming the images with image function. MD5 checksum is used to check the integrity of new images with older.

There are no evaluation metrics available to score the software but software support project goals so it could be ranked on top. The original design and new design doesn’t have much difference so both could be ranked equally but if they are evaluated against each other then new design will appeal more to user.

Implementation of Extension

1. Screenshots of new feature:-

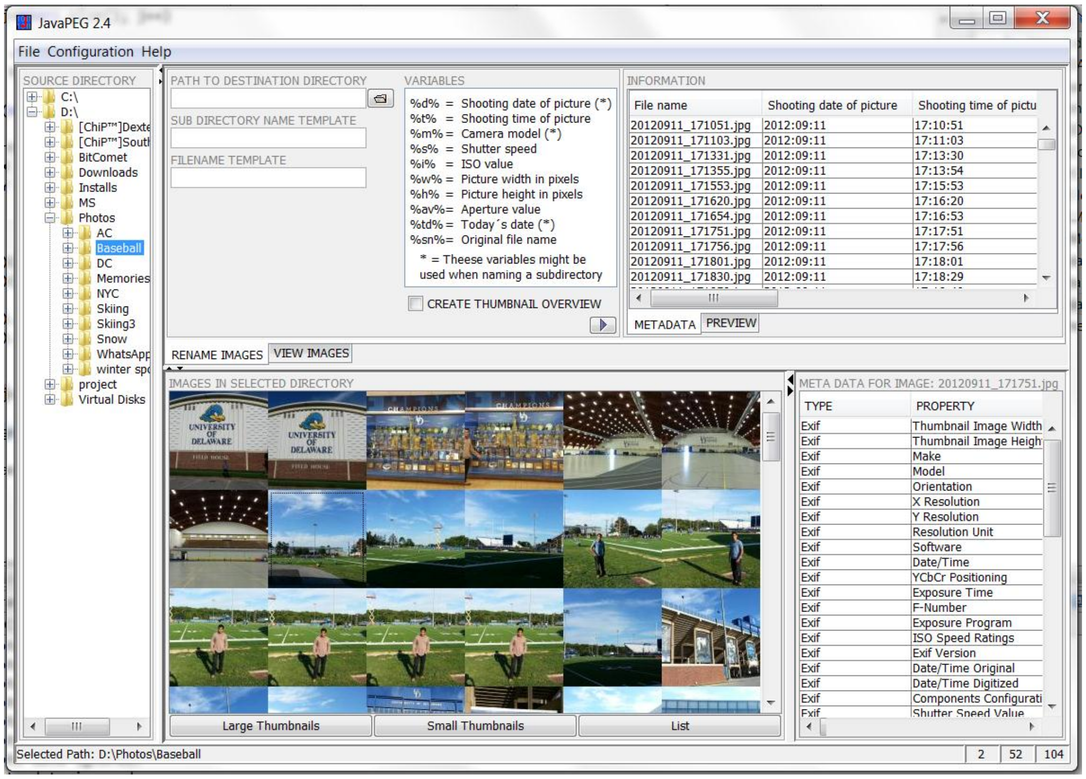


%d% =	Shooting date of picture (*)
%t% =	Shooting time of picture
%m% =	Camera model (*)
%s% =	Shutter speed
%i% =	ISO value
%w% =	Picture width in pixels
%h% =	Picture height in pixels
%av% =	Aperture value
%td% =	Today´s date (*)
%sn% =	Original file name

* = Theese variables might be used when naming a subdirectory

File name	Shooting date of picture	Shooting time of pictu
20120911_171051.jpg	2012:09:11	17:10:51
20120911_171103.jpg	2012:09:11	17:11:03
20120911_171331.jpg	2012:09:11	17:13:30
20120911_171355.jpg	2012:09:11	17:13:54
20120911_171553.jpg	2012:09:11	17:15:53
20120911_171620.jpg	2012:09:11	17:16:20
20120911_171654.jpg	2012:09:11	17:16:53
20120911_171751.jpg	2012:09:11	17:17:51
20120911_171756.jpg	2012:09:11	17:17:56
20120911_171801.jpg	2012:09:11	17:18:01
20120911_171830.jpg	2012:09:11	17:18:29

TYPE	PROPERTY
Exif	Thumbnail Image Width
Exif	Thumbnail Image Height
Exif	Make
Exif	Model
Exif	Orientation
Exif	X Resolution
Exif	Y Resolution
Exif	Resolution Unit
Exif	Software
Exif	Date/Time
Exif	YCbCr Positioning
Exif	Exposure Time
Exif	F-Number
Exif	Exposure Program
Exif	ISO Speed Ratings
Exif	Exif Version
Exif	Date/Time Original
Exif	Date/Time Digitized
Exif	Components Configuration
Exif	Shutter Speed Value

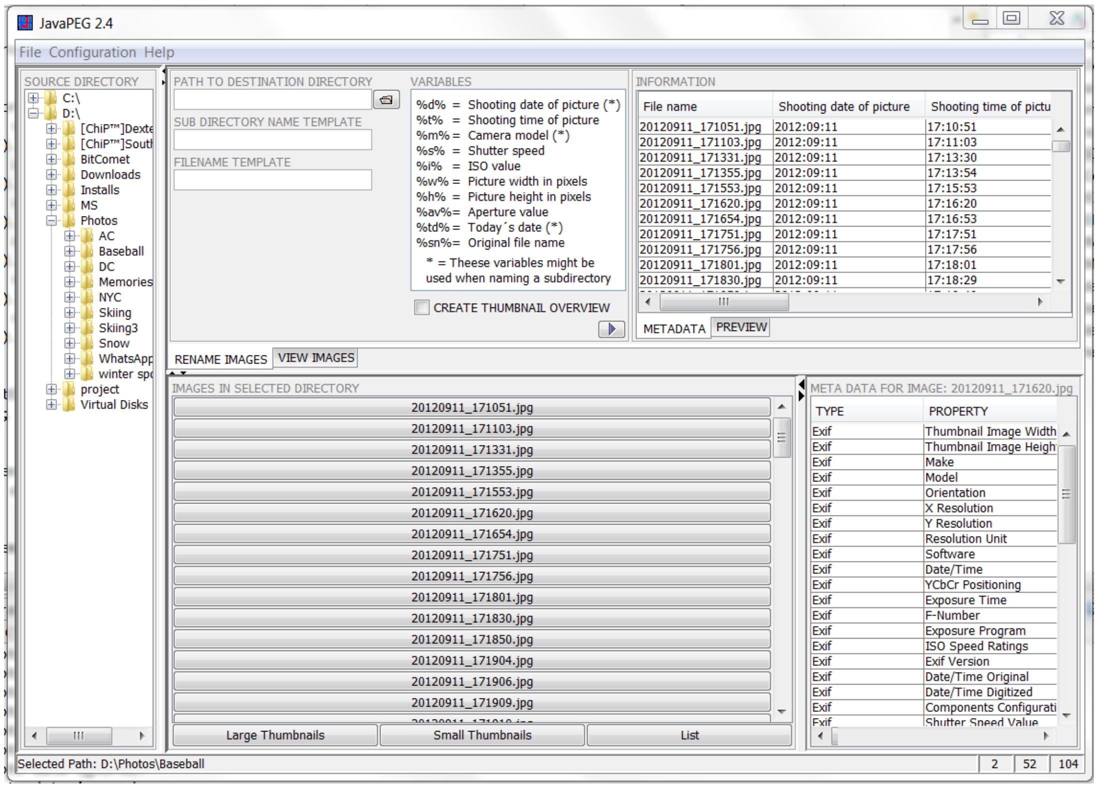


%d% =	Shooting date of picture (*)
%t% =	Shooting time of picture
%m% =	Camera model (*)
%s% =	Shutter speed
%i% =	ISO value
%w% =	Picture width in pixels
%h% =	Picture height in pixels
%av% =	Aperture value
%td% =	Today´s date (*)
%sn% =	Original file name

* = Theese variables might be used when naming a subdirectory

File name	Shooting date of picture	Shooting time of pictu
20120911_171051.jpg	2012:09:11	17:10:51
20120911_171103.jpg	2012:09:11	17:11:03
20120911_171331.jpg	2012:09:11	17:13:30
20120911_171355.jpg	2012:09:11	17:13:54
20120911_171553.jpg	2012:09:11	17:15:53
20120911_171620.jpg	2012:09:11	17:16:20
20120911_171654.jpg	2012:09:11	17:16:53
20120911_171751.jpg	2012:09:11	17:17:51
20120911_171756.jpg	2012:09:11	17:17:56
20120911_171801.jpg	2012:09:11	17:18:01
20120911_171830.jpg	2012:09:11	17:18:29

TYPE	PROPERTY
Exif	Thumbnail Image Width
Exif	Thumbnail Image Heigh
Exif	Make
Exif	Model
Exif	Orientation
Exif	X Resolution
Exif	Y Resolution
Exif	Resolution Unit
Exif	Software
Exif	Date/Time
Exif	YCbCr Positioning
Exif	Exposure Time
Exif	F-Number
Exif	Exposure Program
Exif	ISO Speed Ratings
Exif	Exif Version
Exif	Date/Time Original
Exif	Date/Time Digitized
Exif	Components Configurati
Exif	Shutter Speed Value


JavaPEG 2.4

File Configuration Help

SOURCE DIRECTORY

- C:\
- D:\
  - [ChiP™]Dexte
  - [ChiP™]Soutl
  - BitComet
  - Downloads
  - Installs
  - MS
  - Photos
    - AC
    - Baseball
    - DC
    - Memories
    - NYC
    - Skiing
    - Skiing3
    - Snow
    - WhatsApp
    - winter spo
  - project
  - Virtual Disks

PATH TO DESTINATION DIRECTORY

SUB DIRECTORY NAME TEMPLATE

FILENAME TEMPLATE

VARIABLES

%d% =	Shooting date of picture (*)
%t% =	Shooting time of picture
%m% =	Camera model (*)
%s% =	Shutter speed
%i% =	ISO value
%w% =	Picture width in pixels
%h% =	Picture height in pixels
%av%=	Aperture value
%td% =	Today´s date (*)
%sn%=	Original file name

* = Theese variables might be used when naming a subdirectory

CREATE THUMBNAIL OVERVIEW

INFORMATION

File name	Shooting date of picture	Shooting time of pictu
20120911_171051.jpg	2012:09:11	17:10:51
20120911_171103.jpg	2012:09:11	17:11:03
20120911_171331.jpg	2012:09:11	17:13:30
20120911_171355.jpg	2012:09:11	17:13:54
20120911_171553.jpg	2012:09:11	17:15:53
20120911_171620.jpg	2012:09:11	17:16:20
20120911_171654.jpg	2012:09:11	17:16:53
20120911_171751.jpg	2012:09:11	17:17:51
20120911_171756.jpg	2012:09:11	17:17:56
20120911_171801.jpg	2012:09:11	17:18:01
20120911_171830.jpg	2012:09:11	17:18:29

METADATA PREVIEW

RENAME IMAGES VIEW IMAGES

IMAGES IN SELECTED DIRECTORY

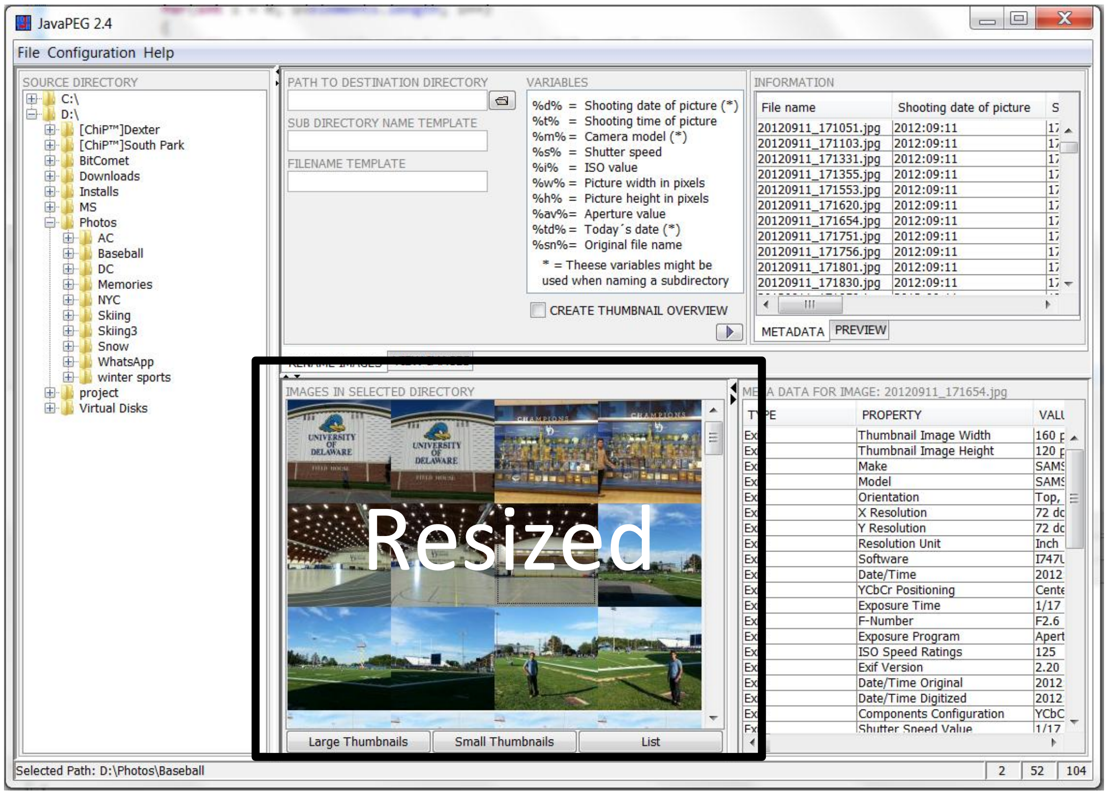
20120911_171051.jpg
20120911_171103.jpg
20120911_171331.jpg
20120911_171355.jpg
20120911_171553.jpg
20120911_171620.jpg
20120911_171654.jpg
20120911_171751.jpg
20120911_171756.jpg
20120911_171801.jpg
20120911_171830.jpg
20120911_171850.jpg
20120911_171904.jpg
20120911_171906.jpg
20120911_171909.jpg

Large Thumbnails Small Thumbnails List

META DATA FOR IMAGE: 20120911_171620.jpg

TYPE	PROPERTY
Exif	Thumbnail Image Width
Exif	Thumbnail Image Heigh
Exif	Make
Exif	Model
Exif	Orientation
Exif	X Resolution
Exif	Y Resolution
Exif	Resolution Unit
Exif	Software
Exif	Date/Time
Exif	YCbCr Positioning
Exif	Exposure Time
Exif	F-Number
Exif	Exposure Program
Exif	ISO Speed Ratings
Exif	Exif Version
Exif	Date/Time Original
Exif	Date/Time Digitized
Exif	Components Configurati
Exif	Shutter Speed Value

Selected Path: D:\Photos\Baseball 2 52 104


JavaPEG 2.4

File Configuration Help

SOURCE DIRECTORY

- C:\
- D:\
  - [ChiP™]Dexter
  - [ChiP™]South Park
  - BitComet
  - Downloads
  - Installs
  - MS
  - Photos
    - AC
    - Baseball
    - DC
    - Memories
    - NYC
    - Skiing
    - Skiing3
    - Snow
    - WhatsApp
    - winter sports
  - project
  - Virtual Disks

PATH TO DESTINATION DIRECTORY

SUB DIRECTORY NAME TEMPLATE

FILENAME TEMPLATE

VARIABLES

%d% =	Shooting date of picture (*)
%t% =	Shooting time of picture
%m% =	Camera model (*)
%s% =	Shutter speed
%i% =	ISO value
%w% =	Picture width in pixels
%h% =	Picture height in pixels
%av%=	Aperture value
%td% =	Today´s date (*)
%sn%=	Original file name

* = Theese variables might be used when naming a subdirectory

CREATE THUMBNAIL OVERVIEW

INFORMATION

File name	Shooting date of picture	S
20120911_171051.jpg	2012:09:11	1
20120911_171103.jpg	2012:09:11	1
20120911_171331.jpg	2012:09:11	1
20120911_171355.jpg	2012:09:11	1
20120911_171553.jpg	2012:09:11	1
20120911_171620.jpg	2012:09:11	1
20120911_171654.jpg	2012:09:11	1
20120911_171751.jpg	2012:09:11	1
20120911_171756.jpg	2012:09:11	1
20120911_171801.jpg	2012:09:11	1
20120911_171830.jpg	2012:09:11	1

METADATA PREVIEW

IMAGES IN SELECTED DIRECTORY

Large Thumbnails Small Thumbnails List

META DATA FOR IMAGE: 20120911_171654.jpg

TYPE	PROPERTY	VALU
Ex	Thumbnail Image Width	160 p
Ex	Thumbnail Image Height	120 p
Ex	Make	SAMS
Ex	Model	SAMS
Ex	Orientation	Top,
Ex	X Resolution	72 dc
Ex	Y Resolution	72 dc
Ex	Resolution Unit	Inch
Ex	Software	I747U
Ex	Date/Time	2012
Ex	YCbCr Positioning	Cente
Ex	Exposure Time	1/17
Ex	F-Number	F2.6
Ex	Exposure Program	Apert
Ex	ISO Speed Ratings	125
Ex	Exif Version	2.20
Ex	Date/Time Original	2012
Ex	Date/Time Digitized	2012
Ex	Components Configuration	YCbC
Ex	Shutter Speed Value	1/17

Selected Path: D:\Photos\Baseball 2 52 104

●First Image shows the execution of option 1 that is Large Thumbnails. ●Second Image shows execution of option 2 that is Small Thumbnails. ●Third Image shows execution of option 3 that is List. ●Last Image shows execution after resizing the window

2. Code:-

- **To add new JPanel - compsToExperiment**

```
final JPanel compsToExperiment = new JPanel();
compsToExperiment.setLayout(new GridLayout(1,3));
```

- **To add JButton1 – Large Thumbnails**

```
JButton button1 = new JButton("Large Thumbnails");
final MainGUI temp = this;
button1.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e)
{
     selectLayout = 3;
     if(completePath!=null)
          loadThumbNails(new File(completePath));
     thumbNailsPanel.invalidate();
     thumbNailsPanel.repaint();
     thumbNailsPanel.updateUI();
}
});
```

- **To add JButton2 – Small Thumbnail**

```
JButton button1 = new JButton("Small Thumbnails");
final MainGUI temp = this;
button1.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e)
{
     selectLayout = 6;
     if(completePath!=null)
          loadThumbNails(new File(completePath));
     thumbNailsPanel.invalidate();
     thumbNailsPanel.repaint();
     thumbNailsPanel.updateUI();
}
});
```

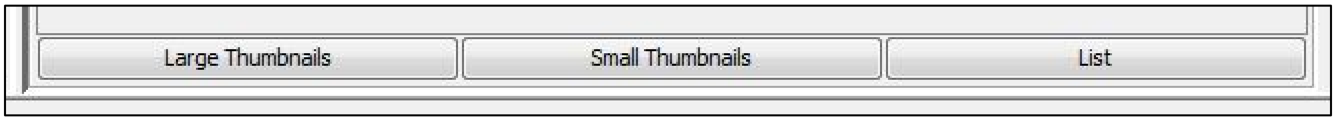
- **To add JButton3 – List**

```
JButton button1 = new JButton("Large Thumbnails");
final MainGUI temp = this;
button1.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e){
     selectLayout = 3;
     if(completePath!=null)
          loadThumbNails(new File(completePath));
     thumbNailsPanel.invalidate();
     thumbNailsPanel.repaint();
     thumbNailsPanel.updateUI();}});
```

Code to add:-

```
compsToExperiment.add(button1);
compsToExperiment.add(button2);
compsToExperiment.add(button3);
```

Following changes will occur in GUI

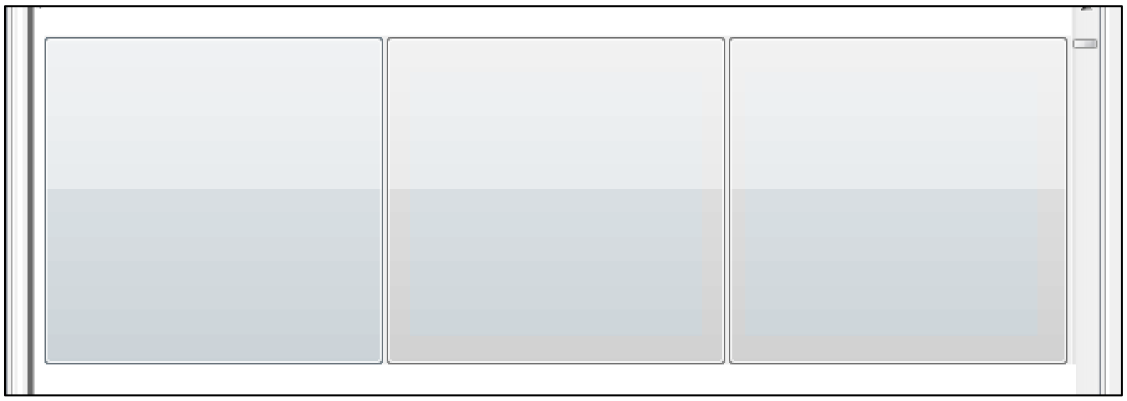


- **<u>To create new JButton tumbContainer – To display an image in one button and so on</u>**

```
JButton thumbContainer;
if(selectLayout != 1)
{
     thumbContainer = new JButton();
     thumbContainer.setIcon(new ImageIcon(tn.getThumbNailData()));
}
else
     thumbContainer = new JButton(jpegFile.getName());
```

- **<u>To call addThumbnail – it will call addThumbnail to add thumbContainer button</u>**

```
int width = 0 ;
if(selectLayout != 1)
          width = thumbContainer.getIcon().getIconWidth();
if (width > iconWidth)
{
     iconWidth = width;
}
items++;
     ......

addThumbnail(thumbContainer, items);
```



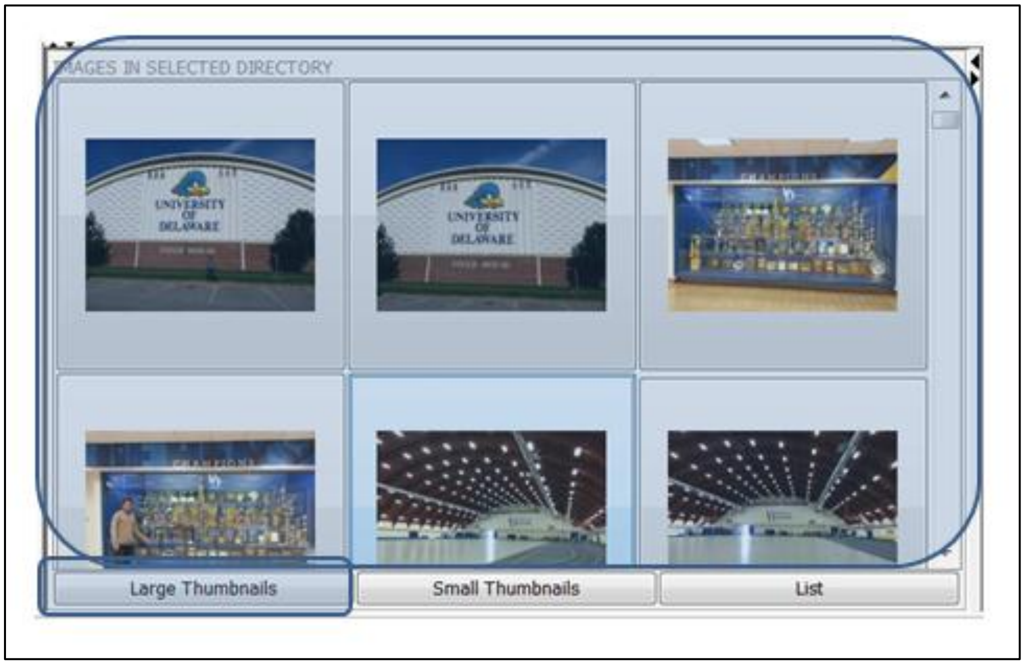
These are the three buttons in a row...one by one they will be added and displayed after *loadThumbnails()* method is called.

- **<u>To add the button - the addThumbnail method</u>**

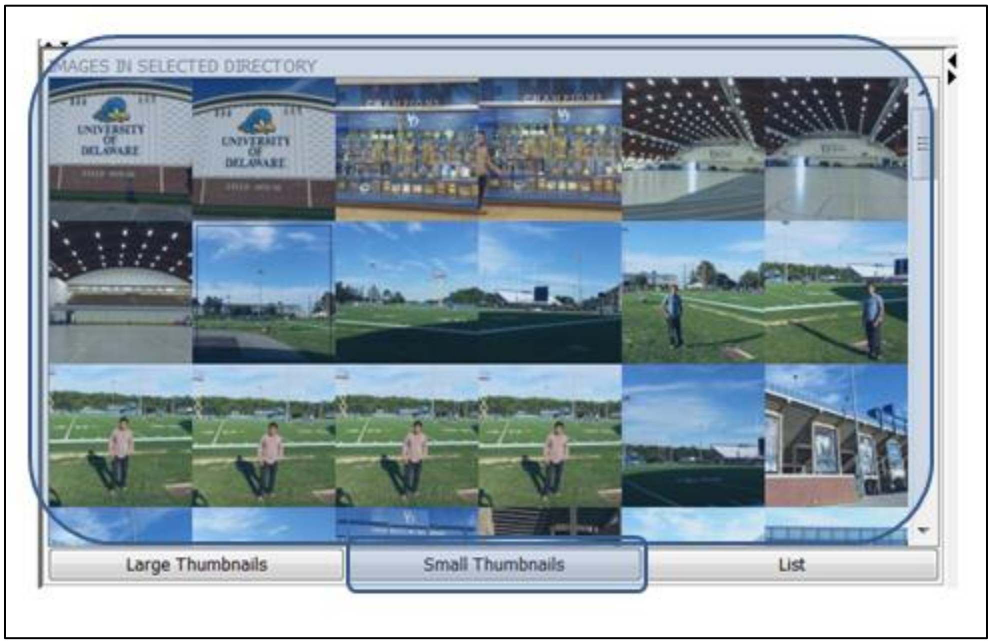
```
private void addThumbnail(JButton thumbNail, int items)
{
int columns = thumbNailsPanel.getVisibleRect().width ;
columns = (int)Math.floor( columns/100.0 );
if(selectLayout == 6)
      columns = columns;
else if(selectLayout == 3)
      columns = columns/2;
else
      columns = 1;
int width = (thumbNailsPanel.getVisibleRect().width - 22) / columns;

GridBagConstraints c = new GridBagConstraints();
c.fill = GridBagConstraints.HORIZONTAL;
c.weightx = 1.0;
c.weighty = 1.0;
c.gridx = items%columns;

int height = selectLayout==1?22:width;

thumbNail.setPreferredSize(new Dimension(width, height));
thumbNailsPanel.add(thumbNail, c);
}
```

<u>Heirarchy</u> - *Buttons → Component listener →loadThumbails → addThumbnail*

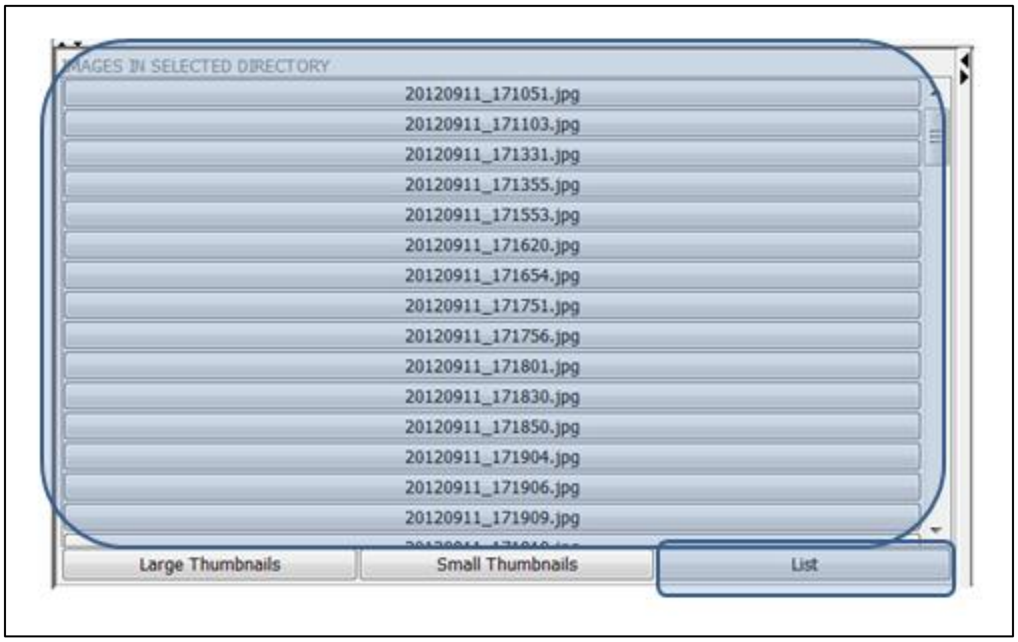
After clicking on first option that is Large Thumbnails, listener will perform its action and call *loadThumbnails*, following screen will be displayed.



After clicking on second option that is Small Thumbnails, listener will perform its action and call *loadThumbnails*, following screen will be displayed.



After clicking on third option that is List, listener will perform its action and call *loadThumbnails*, following screen will be displayed.



20120911_171051.jpg
20120911_171103.jpg
20120911_171331.jpg
20120911_171355.jpg
20120911_171553.jpg
20120911_171620.jpg
20120911_171654.jpg
20120911_171751.jpg
20120911_171756.jpg
20120911_171801.jpg
20120911_171830.jpg
20120911_171850.jpg
20120911_171904.jpg
20120911_171906.jpg
20120911_171909.jpg

3. Instructions to run the software:-

1. Open Java – Eclipse.
2. Select File and then select Import from drop down menu.
3. Select General and then Existing Projects into Workspace from the subdirectory, hit Next.
4. Select "Select archive file", browse the .zip file provided in the submission, click Finish.
5. In Package Explorer at left, select JavaPEG and click on Run option () on above.

   (If it ask to select which file to run then select StartJavaPEG.java which contains main method. Also, it will sometime prompt that program contains errors, proceed launch with error? Say yes because errors were found in open source project only and those are in Test cases not in source code.
   The software runs perfectly and for better user experience use Maximized mode of software)

Reflection on work and Learning

There are lots of new things to learn from this main project. This was not the project of coding or building the software with large code. Coding was the easy task though but important in extending the functionality point of view.

At the start of project selection of the software was the major part of worry. There were lots of options available to select the open source project. They were in different programming languages so I choose one in Java in which I am comfortable. In java, there were like 35000 open source softwares available for consideration. Many software were on small scale including <1000 LOC and some were huge containing >100000 LOC. So, finding the software of medium complexity which is easy to understand but still complex enough was the tough job. Two to three days were spent on selecting the project and finally JavaPEG was the one in which I developed my interest.

In next phase the reverse engineering was the toughest job in whole project, I found. There are lot of resources available on the internet for architectural styles and patterns but it’s painful to say confidently that which style is present in your project. Finally, after acquiring enough knowledge of the project code and after supervised learning of styles and patterns I came up with some results.

After reverse engineering I had to choose the functionality which I could extend in my open source software. This wasn’t tough though because if you compare your software with similar software in the market then you will find lot of other functionalities that are absent in your software. So I learnt about one of the functionality to extend the views for Thumbnail Panel.

Coding was the last part which shows your programming capability. I got to learn about how to find code of the function in your software, also how to analyze it and identify the flow of software. New global variables were used, new buttons and panel were added. I faced lot of problems in finding the procedure to add them. Debugging was the part of learning where I get to learn many things such as how to solve compile time and runtime errors, how to imagin the future risks and hazards.

In short, the project was important and interesting from all points of view; coding as well as software engineering.

Conclusion

Certain benefits and conclusions from new system are mentioned below:

- Design supports both product and project goals, system could execute the extended function successfully.
- There is not much technology used so it is definitely feasible from technology point of view. There are no risks or hazards to the system.
- The design support future development and the written code by developer could be reused.
- The system was not at all complex because every function was well divided in packages. The design allow scalability.
- The new design is consistent with development and operating environments. There is no integrity violation.
- Design allows ease of maintenance. Maintenance is not needed but it could be done easily.
- There are no risks or hazards after adding the new functionality which simplifies that the older design is scalable.
- The GUI is user friendly and notations are self-explanatory. Help is provided for each function of JavaPEG