GPS IMAGE TAGGER

A LOCATION AWARE CAMERA SYSTEM

CSC 59867 - Senior Project II

Final Report

Kanhar Munshi
Computer Science
City College of New York

Asher Snyder
Computer Science
City College of New York

Mentor
Professor Izidor Gertner
Computer Science Department
City College of New York

REVISION HISTORY

December 15, 2009, Revision 1.6
December 7, 2009, Revision 1.5
December 4, 2009, Revision 1.4
October 21, 2009, Revision 1.3
October 17, 2009, Revision 1.2
October 11, 2009, Revision 1.1

Table of Contents

Abstract

We have specified, designed, documented and implemented as our final senior design project the 'GPSImageTagger', an Android based software project consisting of three applications, GeoTag, GeoImageView and ImageSearch. The design of which requires all the three applications to work in conjunction, and extend off the existing open source Exchange Image Information Format (EXIF) by way of adding three additional image parameters, Latitude, Longitude and CustomUserInput into an Image as taken by a cell phone camera that runs the Android Operating System. The application was implemented using the Java Language, and provides an interface to capture images, obtain geographic co-ordinates and request additional image related user input, all of which are stored in the image's metadata. Images once extracted to a desktop environment can be viewed and searched by the ImageSearch Tool. Our application demonstrates a real world emulated deployment of an application that adds to current open source technology by adding levels of innovation and user enhancements. The emulated deployment of the application allowed us to incorporate agile testing methods in our work flow, providing rapid bug testing, with minimum hardware or time overheads.

CSC 59867 - Program Educational Outcomes

In participating in this senior year's computer science Senior Design course, under the mentorship of Professor Izidor Gertner, we were able to access, improve and innovate upon the latest Real Time Operating System – Android, and extend of a widely used open source image format, for storing digital images – Exchangeable Information Image Format (EXIF).

This hands on approach to cutting edge technology, and the implementation of a full life cycle in software engineering, has given us tremendous insight into real world challenges, thereby giving us valuable experience in future engineering projects of such scale and technological innovation.

The authors below separately discuss the course's educational benefits in specific terms, each applying their unique insight into computer technology and experience in the industry as reference frames, for such an analysis.

These responses are presented below categorized by the respective authors of this report as below:-

Kanhar Munshi, Computer Science, CCNY 2010

a. An ability to apply knowledge of computing and mathematics appropriate to the discipline

Our previous computer science courses prepared us with the necessary knowledge to tackle this project. Our project was an Image Geo Tagging Android application in which we used Object Oriented techniques such as inheritance, and polymorphism. These techniques allowed us to successfully complete the project.

b. An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution

In our project we identified the weaknesses in the EXIF image format. We proposed solutions to help solve these weaknesses. We implemented these solutions using a patchwork of existing tools, the result of which is a refreshingly innovative program that combines the hardware functionality of GPS, Camera Units, and Cell Phone Flash Memory. Our senior design class allowed us the opportunity to analyze the problems, and the necessary time and encouragement to implement the solutions.

c. An ability to design, implements, and evaluate a computer-based system, process, component, or program to meet desired needs

For our project we used an Android Emulator that allowed us to quickly deploy programs without an Android Phone to test on, thus demonstrating the computer science principle the efficiency advantage, using a virtual machine to emulate a physical machine.

We also used the following open source tools:

- Eclipse Eudora 4.5
- Java Development Kit 1.5
- Android Emulator 1.5
- Android SDK 1.6
- Eclipse DDMS Android File Explorer

- Tortoise SVN (for Source Control)\
- Dia (For UML and Class Diagrams)

d. An ability to function effectively on teams to accomplish a common goal

Our team met regularly and used GTalk, FTP and Dia to help communicate and collaborate. These tools allowed our team to function effectively and complete the project.

e. An understanding of professional, ethical, legal, security and social issues and responsibilities

In the Ethics component of our class we thoroughly discussed the professional, ethical, legal, security, social issues and responsibilities

f. An ability to communicate effectively with a range of audiences

Our Ethics course taught us the necessary demeanor and approach to communicate with a wide range of audiences, from the layman to the technical.

g. An ability to analyze the local and global impact of computing on individuals, organizations, and society

We saw firsthand through the many assignments in our Ethics components the local and global impact of computing on individuals, organizations, and greater society.

h. Recognition of the need for and an ability to engage in continuing professional development

Our Ethics component coupled with Capstone taught me to recognize my personal faults and to try to constantly improve my ability as a professional. As I continue to work I will always continue to learn and better myself.

i. An ability to use current techniques, skills, and tools necessary for computing practice

In our project we used technologies that we have never used before, and adapted them to the changing requirements of our application. we valuable skills in development of mobile applications.

j. An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices

The most important decisions we had to take were based on the choice of operating system, Android versus Moblin versus Windows Mobile. Each choice came with various pros and cons. In addition, we had to make tough design decisions on our classes, inheritance, and code length. Our understanding of OOP principles helped us to decide these many decisions

k. An ability to apply design and development principles in the construction of software systems of varying complexity

Throughout the length of the project the complexity of the project increased. As such, it was necessary to adapt and modify our initial designs. We used the principles of Agile Development and Extreme Programming to develop our project, as such we were able to iterate and make changes rapidly.

Asher Snyder, Computer Science, CCNY 2010

a. An ability to apply knowledge of computing and mathematics appropriate to the discipline

Our previous computer science courses prepared us with the necessary knowledge to tackle this project. Our project was an Image Geo Tagging Android application in which we used Object Oriented techniques such as inheritance, and polymorphism. These techniques allowed us to successfully complete the project.

b. An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution

In our project we identified the weaknesses in the EXIF image format. We proposed solutions to help solve these weaknesses. We implemented these solutions using a patchwork of existing tools, the result of which is a refreshingly innovative program that combines the hardware functionality of GPS, Camera Units, and Cell Phone Flash Memory. Our senior design class allowed us the opportunity to analyze the problems, and the necessary time and encouragement to implement the solutions.

c. An ability to design, implements, and evaluate a computer-based system, process, component, or program to meet desired needs

For our project we used an Android Emulator that allowed us to quickly deploy programs without an Android Phone to test on, thus demonstrating the computer science principle the efficiency advantage, using a virtual machine to emulate a physical machine.

We also used the following open source tools:

- Eclipse Eudora 4.5
- Java Development Kit 1.5
- Android Emulator 1.5
- Android SDK 1.6
- Eclipse DDMS Android File Explorer
- Tortoise SVN (for Source Control)\
- Dia (For UML and Class Diagrams)

d. An ability to function effectively on teams to accomplish a common goal

Our team met regularly and used GTalk, FTP and Dia to help communicate and collaborate. These tools allowed our team to function effectively and complete the project.

e. An understanding of professional, ethical, legal, security and social issues and responsibilities

In the Ethics component of our class we thoroughly discussed the professional, ethical, legal, security, social issues and responsibilities

f. An ability to communicate effectively with a range of audiences

Our Ethics course taught us the necessary demeanor and approach to communicate with a wide range of audiences, from the layman to the technical.

g. An ability to analyze the local and global impact of computing on individuals, organizations, and society

We saw firsthand through the many assignments in our Ethics components the local and global impact of computing on individuals, organizations, and greater society.

h. Recognition of the need for and an ability to engage in continuing professional development

Our Ethics component coupled with Capstone taught me to recognize my personal faults and to try to constantly improve my ability as a professional. As I continue to work I will always continue to learn and better myself.

i. An ability to use current techniques, skills, and tools necessary for computing practice

In our project we used technologies that we have never used before, and adapted them to the changing requirements of our application.

j. An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices

The most important decisions we had to take were based on the choice of operating system, Android versus Moblin versus Windows Mobile. Each choice came with various pros and cons. In addition, we had to make tough design decisions on our classes, inheritance, and code length. Our understanding of OOP principles helped us to decide these many decisions

k. An ability to apply design and development principles in the construction of software systems of varying complexity

We noticed that even as we had to use different tools to implement our solution, a lot of the design and development principles remained the same, and this encouraged us to be ambitious in our approach, and ultimately this reflects in the scope and robustness of our final application.

GPSImageTagger Introduction

Overview

With the integration of GPS technology into cell phones, geo-coding (obtaining geographic co-ordinates) an image is now possible, thus allowing a camera system to be able to store in addition to an image of the environment, additional data about the environment following the GPS NMEA format which includes latitude, longitude, elevation, date time, speed amongst other satellite based variables.

Our camera application called Geo-Tag that we developed using the Android framework 1.5, and tested on an Android emulator, allows the user to use the onboard camera and GPS of the phone to capture an image, and image-specific geographic co-ordinates, along with additional metadata that the user may wish to enter after having captured the image.

Our implementation involved adding or overwriting fields where appropriate to an open source format called the **Exchangeable Image File Format** (EXIF) version 2.2, a format that is now widely used by digital cameras especially those using the standard lossy JPEG images.

We envision that this additional Meta data can be helpful in later indexing the images.

The GPSImageTagger Application is divided into three main applications that perform all its functional requirements and they are listed as follows:-

- GeoTag (Android)
- GeoImageView (Android)
- ImageSearch (Desktop)

Future references to the GPSImageTagger Application would refer to all the above three in sum.

GPSImageTagger Program Objectives

GeoTag - Android Based Application

- Should Provide an Android based Camera Application that captures an image.
- After capturing of image, Application Requests for additional Image or environment specific metadata as defined by the user, and entered via an Android QWERTY keyboard or using phone's touch screen.
- Application checks phone hardware features for GPS capability and requests current latitude and longitude information, which if available are stored within the EXIF format framework.

GeoImageView - Android Based Application

- Application provides an onboard Search Image Screen to query captured images using the metadata entered only while displaying selected images on the phone screen along with GPS co-ordinates and customer user input.
- Selecting on one image, displays the full image, along with a brief snapshot of image data stored in image.

Image Viewer - Desktop Based Interface

- Application assumes that the System User has the ability to push/pull images from the Android phone, a feature provided by default in a Windows Computer, where the Microsoft USB Controller automatically synchronizes external Memory Sticks with Desktop Hard Drives using a Windows File Explorer Interface.
- System User is expected to Extract the Images into a Folder on the Desktop
- Once images are so extracted, Application provides a native Java Application that allows System User to browse to folder where images are stored, and to search through images using custom user input.
- User can preview all images in a slideshow manner, or just select on one image.
- Selecting on one image, displays the full image, along with a brief snapshot of image data stored in image.

Overview of an Existing Exchangeable Image Information Specification Standard

Exchangeable image file format (EXIF) specification standard is a specification for the image file format used by digital cameras. The specification borrows a lot from the existing JPEG, TIFF Rev. 6.0, and RIFF WAV file formats, with the addition of specific metadata tags, a snapshot of which is displayed in the next section.

Background

EXIF was created by the Japan Electronic Industries Development Association (JEIDA). Version 2.1 of the specification is dated June 12, 1998. Though the specification is not currently maintained by any industry or standards organization, its use by camera manufacturers is nearly universal.

The metadata tags defined in the EXIF standard cover a broad spectrum:

- Date and time information: Digital cameras will record the current date and time and save this in the metadata.
- Camera settings: This includes static information such as the camera model and make, and information that varies with each image such as orientation (rotation), aperture, shutter speed, focal length, metering mode, and ISO speed information.

Technical

EXIF data is embedded within the image file itself. While many recent image manipulation programs recognize and preserve EXIF data when writing to a modified image, this is not the case for older programs. Many image gallery programs also recognize EXIF data and optionally display it alongside the images.

Improvements

Metadata Working Group was formed by a consortium of companies in 2006. It released its first document on 24 September, 2008[10], giving recommendations concerning the use of EXIF, IPTC and XMP metadata in images.

Extensible Metadata Platform (XMP) was created by Adobe Systems to be a better metadata format for photography and image processing. However, it is generally unsupported in cameras.

EXIF 2.1 Specification July 1998

Below is an extract of the EXIF 2.2 Format specification that shows the format Image Specific data is stored within EXIF images.

Tag Name	Field Name	Tag ID		Type	Count
		Dec	Hex		
A. Tags relating to image data structure					
Image width	ImageWidth	256	100	SHORT or LONG	1
Image height	ImageLength	257	101	SHORT or LONG	1
Number of bits per component	BitsPerSample	258	102	SHORT	3
Compression scheme	Compression	259	103	SHORT	1
Pixel composition	PhotometricInterpretation	262	106	SHORT	1
Orientation of image	Orientation	274	112	SHORT	1
Number of components	SamplesPerPixel	277	115	SHORT	1
Image data arrangement	PlanarConfiguration	284	11C	SHORT	1
Subsampling ratio of Y to C	YCbCrSubSampling	530	212	SHORT	2
Y and C positioning	YCbCrPositioning	531	213	SHORT	1
Image resolution in width direction	XResolution	282	11A	RATIONAL	1
Image resolution in height direction	YResolution	283	11B	RATIONAL	1
Unit of X and Y resolution	ResolutionUnit	296	128	SHORT	1
B. Tags relating to recording offset					
Image data location	StripOffsets	273	111	SHORT or LONG	*S
Number of rows per strip	RowsPerStrip	278	116	SHORT or LONG	1
Bytes per compressed strip	StripByteCounts	279	117	SHORT or LONG	*S
Offset to JPEG SOI	JPEGInterchangeFormat	513	201	LONG	1
Bytes of JPEG data	JPEGInterchangeFormatLength	514	202	LONG	1
C. Tags relating to image data characteristics					
Transfer function	TransferFunction	301	12D	SHORT	3 * 256
White point chromaticity	WhitePoint	318	13E	RATIONAL	2
Chromaticities of primaries	PrimaryChromaticities	319	13F	RATIONAL	6
Color space transformation matrix coefficients	YCbCrCoefficients	529	211	RATIONAL	3
Pair of black and white reference values	ReferenceBlackWhite	532	214	RATIONAL	6
D. Other tags					
File change date and time	DateTime	306	132	ASCII	20
Image title	ImageDescription	270	10E	ASCII	Any
Image input equipment manufacturer	Make	271	10F	ASCII	Any
Image inputequipment model	Model	272	110	ASCII	Any
Software used	Software	305	131	ASCII	Any
Person who created the image	Artist	315	13B	ASCII	Any
Copyright holder	Copyright	3432	8298	ASCII	Any

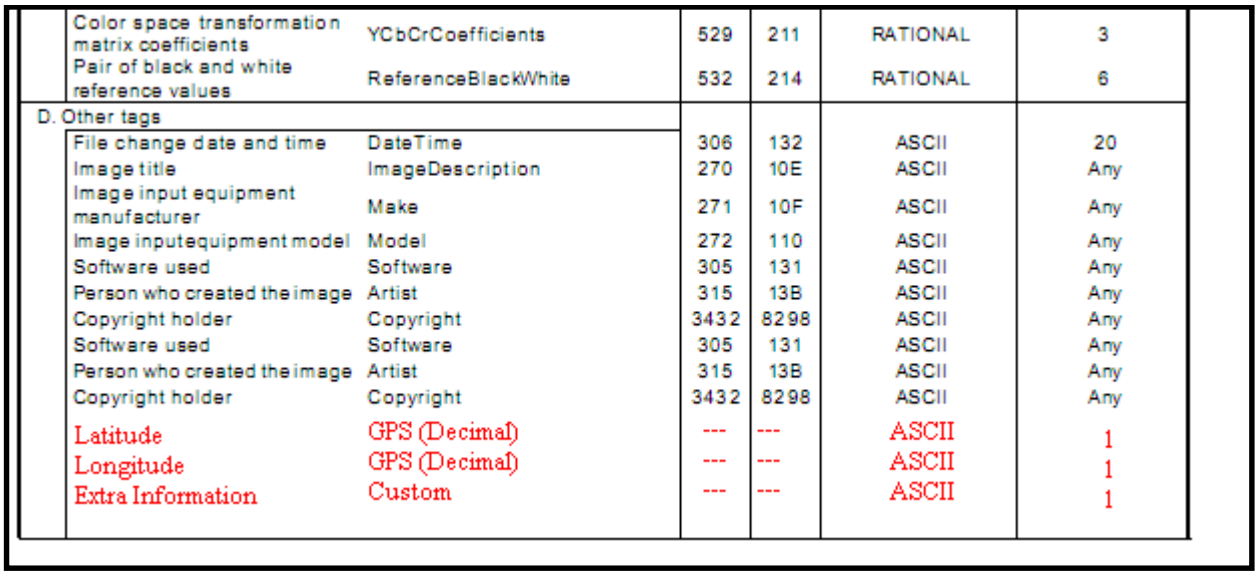
Figure 1: TIFF Rev 6.0 Attribute Information Used in EXIF

EXIF 2.2 Format Modification

Our implementation required the modification of this format to add the following fields, marked in red in the picture attached below.

We therefore modified the EXIF 2.2 format by adding the following to the Sub Category – "**Other Tags**", all of which are eventually represented in an ASCII format, and have a one to one relationship with an image. For example one Image can only have one set of GPS Co-Ordinates and one User Input or Extra Information associated with it. Below are the two main additions in our program:-

- **GPS Information**
  - Latitude (Type Decimal)
  - Longitude (Type Decimal)
- **User Input / Extra Information**
  - Custom (Type varchar)



	Color space transformation matrix coefficients	YCbCrCoefficients	529	211	RATIONAL	3
	Pair of black and white reference values	ReferenceBlackWhite	532	214	RATIONAL	6
D. Other tags						
	File change date and time	DateTime	306	132	ASCII	20
	Image title	ImageDescription	270	10E	ASCII	Any
	Image input equipment manufacturer	Make	271	10F	ASCII	Any
	Image inputequipment model	Model	272	110	ASCII	Any
	Software used	Software	305	131	ASCII	Any
	Person who created the image	Artist	315	13B	ASCII	Any
	Copyright holder	Copyright	3432	8298	ASCII	Any
	Software used	Software	305	131	ASCII	Any
	Person who created the image	Artist	315	13B	ASCII	Any
	Copyright holder	Copyright	3432	8298	ASCII	Any
	Latitude	GPS (Decimal)	---	---	ASCII	1
	Longitude	GPS (Decimal)	---	---	ASCII	1
	Extra Information	Custom	---	---	ASCII	1

Figure 2: Modifications are Visible in Red

EXIF Modification Implementation

In adding the field for Custom User Input into an existing Image, we had to essentially recalculate the byte addresses (start and end) that would be required to insert our new attribute, within the image.

Instead we used a work around where in we used an existing attribute definition, specifically with respect to attribute length, byte size, and Field Type called EXIF_TAG_IMAGE_DESCRIPTION that had exactly similarly properties to what we envision Custom User Input would require.

We then reproduced its memory space, field type and byte size, and over wrote it with our custom user input, and finally inserted it directly into the EXIF format as shown below:-

```
// Establish EXIF in Image Format
if (jpegMetadata != null) {exif = jpegMetadata.getExif();}
outputSet = exif.getOutputSet();

// Add Custom EXIF Field(s)
if (outputSet != null)
{
    TiffOutputField imageDescription = outputSet.findField(TiffConstants.EXIF_TAG_IMAGE_DESCRIPTION);

        imageDescription = new TiffOutputField
                (
                    ExifTagConstants.EXIF_TAG_IMAGE_DESCRIPTION,
                    TiffFieldTypeConstants.FIELD_TYPE_ASCII,
                    description.length(),
                    description.getBytes()
                );
        TiffOutputDirectory exifDirectory = outputSet.getOrCreateExifDirectory();
        exifDirectory.add(imageDescription);
        Location loc = mLocationListener.getLocation();
        if (loc != null)
            outputSet.setGPSInDegrees(loc.getLongitude(), loc.getLatitude());
}

// Create Final Output Stream write/update EXIF metadata to output stream
try
{
    tempFile = File.createTempFile("temp-" + System.currentTimeMillis(), ".jpeg");
    os = new FileOutputStream(tempFile);
    os = new BufferedOutputStream(os);
    new ExifRewriter().updateExifMetadataLossless(file, os, outputSet);
}
```

Figure 3: Attribute CustomUserInput Insertion Procedure Code Snapshot

GPSImageTagger System Roles

Camera User

Camera User refers to the user that owns and operates the camera and is depicted in the use case diagram by:

The Camera User can:

- Capture Images
- Enter Image metadata
- Search through Images onboard Camera Application

The Camera User is represented in Collaboration Diagrams as :-

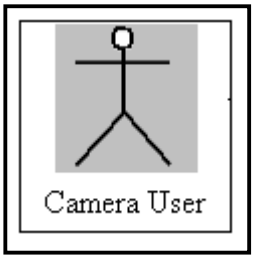


Figure 4: Camera User

System User

System User refers to the user that connects the camera to a Desktop computer, and has unrestricted access to the file management system of the desktop and by extension the camera.

The System User can:

- Download Images using Native Android File Management System (DDMS)
- Search through Images
- Preview image Metadata
- The Camera User is represented in Collaboration Diagrams as :-

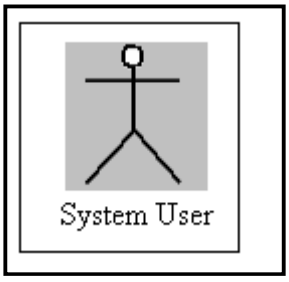


Figure 5: System User

GPSImageTagger Use Case Diagrams

The use case diagram clearly depicts that the System User (SU) cannot add user inputted metadata tags, thereby increasing consistency and integrity of such tags, which can intuitively only be entered after the picture is taken.

Use Case Diagram

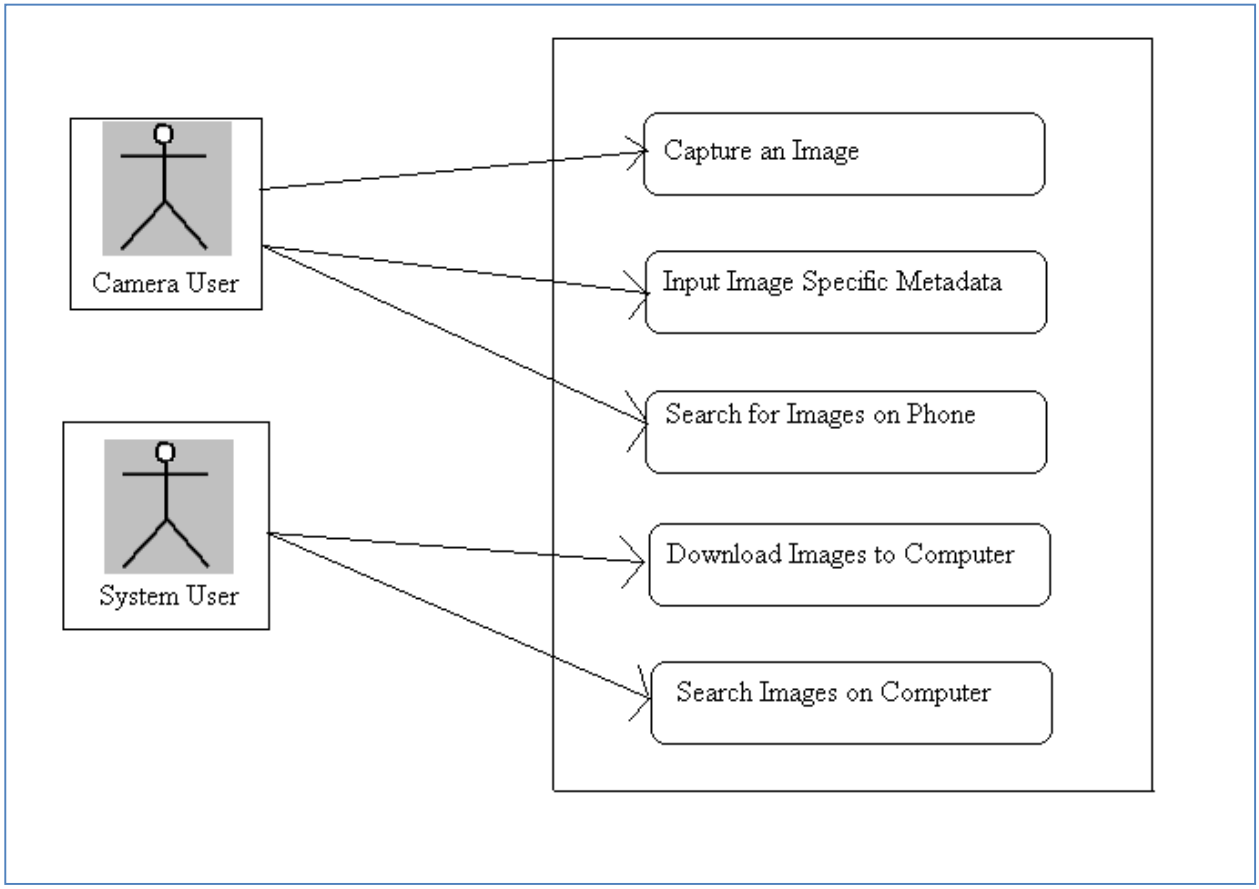


Figure 6: Use Case Collaboration Diagram

GPSImageTagger Collaboration Diagrams

The application is divided into three sub applications, two that are based on the Camera and one that is based on the Desktop that is expected to be at some point in time connected to the Camera.

Each application has specific roles, although **ImageSearch** and **GeoImageView** programs have similar functionality, but they differ in that one is Desktop based where as the other is Android Based.

The three main collaboration diagrams required for the GPSImageTagger Application are the:-

Sub Application: GeoTag (Android)

- Based: Android Emulator Version 1.5 or Android phone
- Functionality: Camera User takes a Picture

Sub Application: GeoImageView (Android)

- Based: Android Emulator Version 1.5 or Android phone
- Functionality: Camera User Searches through onboard pictures

Sub Application: ImageSearch (Desktop)

- Based: Desktop Client Operating System that Synchronizes with Android Phone
- Functionality: System User Searches through onboard pictures

All the above three constitute the GPSImageTagger application.

GeoTag: Camera User (CU) takes a picture

The application must make the following decisions for every picture it takes, the process flow show below incorporates all combinations of the following:-

- Does the Camera Hardware support GPS Positioning?
  - If not Camera defaults Latitude and Longitude to 0.0
- Is GPS Positioning working, and enabled?
  - If not Camera defaults Latitude and Longitude to 0.0, and logs specific error.
- Did the User Input any Image Specific Data?
  - If User does not input any data, System accepts that as an Input equivalent to an empty string. The memory address having been already reserved, must be filled

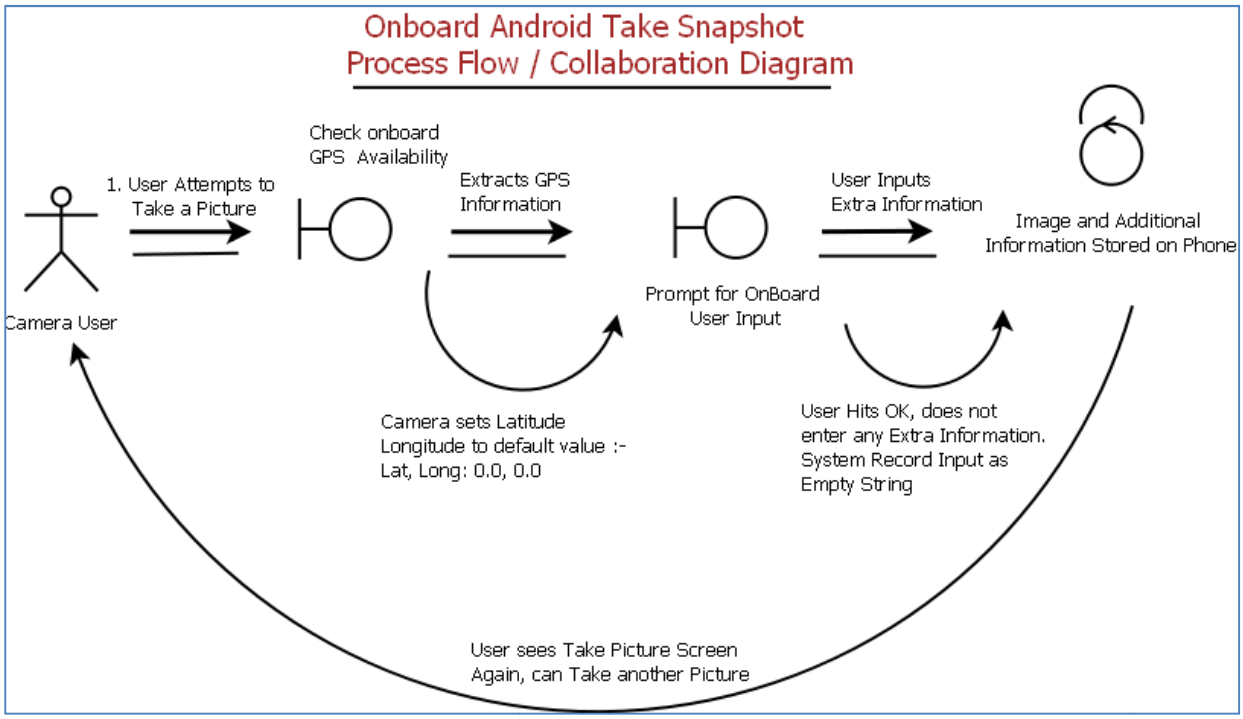


Figure 7: Collaboration Diagram Take Picture

Exceptional Scenario

- **Org.apache.sanselan.ImageWriteException** (Inability to Write into Image Designated Memory Address)
- **Org.apache.sanselan.ImageReadException** (Inability to Read from Image Designated Memory Address)
- **Java.io.IOException** (Could be many reasons that can give rise to this error, including a thread access violation, where two threads are accessing a shared resource, or just an Input Output exception that inhibits the system from accessing a memory address)

GeoImageView: Camera User (CU) Searches for Image

The following variables or user input are to be accounted for within the Onboard Image Search process flow:-

- Does the User Enter a Search String?
    - Special Case: If user does not enter a Search String, System returns all Images on Camera.
- Is the Search String Found?
    - System returns “System returns Exception: Search String Not found in Image Collection”
- If Image found, will the user Request Additional Details?
- Will System User Want to Search for Additional Images?

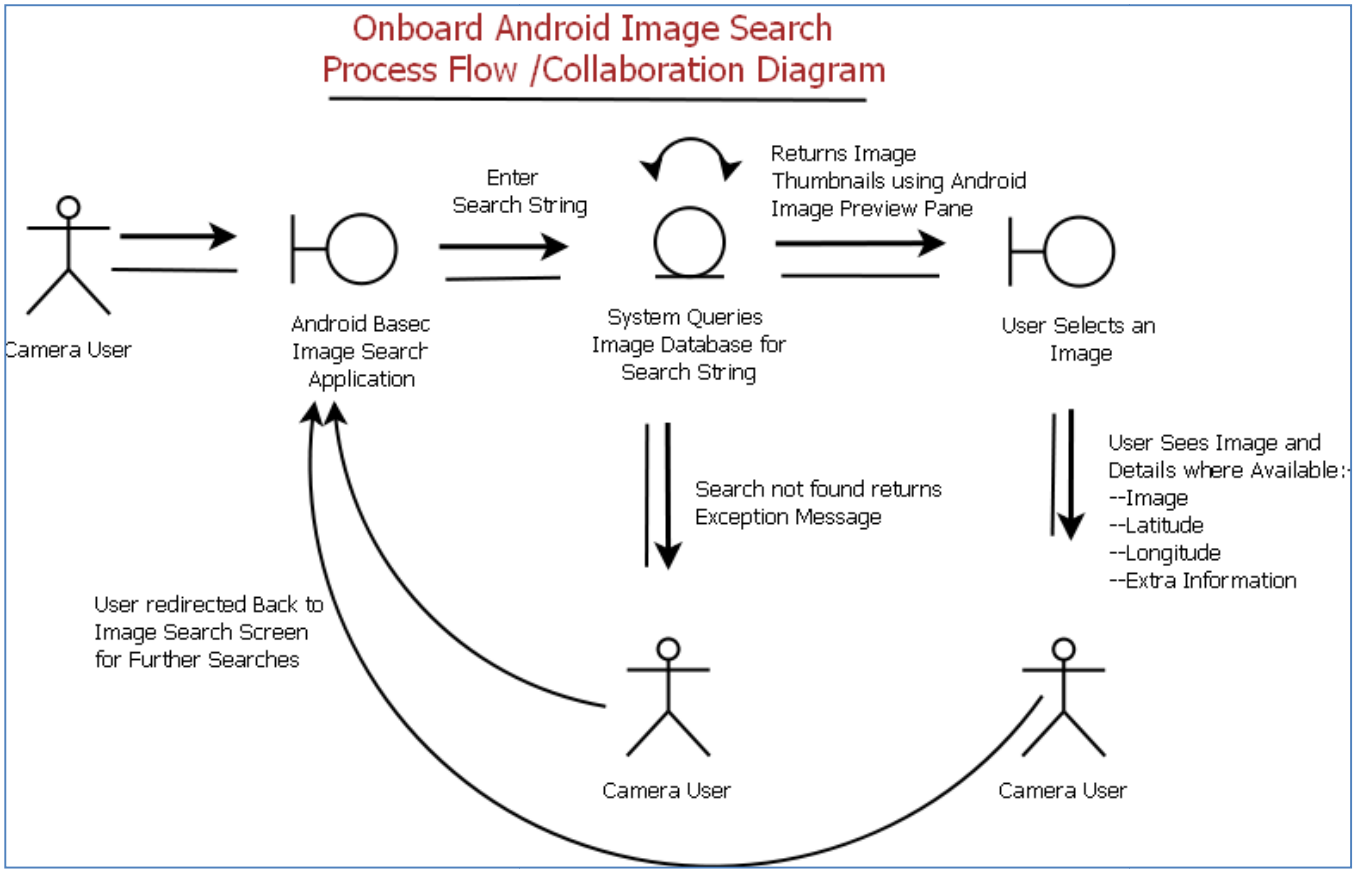


Figure 8: Collaboration Diagram View Images Onboard Camera

Exceptional Scenario:

- **Org.apache.sanselan.ImageWriteException** (Inability to Write into Image Designated Memory Address)
- **Org.apache.sanselan.ImageReadException** (Inability to Read from Image Designated Memory Address)
- **Java.io.IllegalAccessException** (If system, reads miscalculates Memory Address bounds, in order to read/display image, Java would return an Access Violation Exception, which will be caught, logged, and application will gracefully exit cleaning up resources in the finally block)

ImageSearch: System User Opens Image Desktop Viewer Application

The following variables or user input are to be accounted for within the Desktop Image Search process flow, which are intuitively similar to the onboard image search process:-

- Does the User Enter a Search String?
    - Special Case: If user does not enter a Search String, System returns all Images on Camera.
- Is the Search String Found?
    - System returns “System returns Exception: Search String Not found in Image Collection”
- If Image found, will the user Request Additional Details?
- Will System User Want to Search for Additional Images?

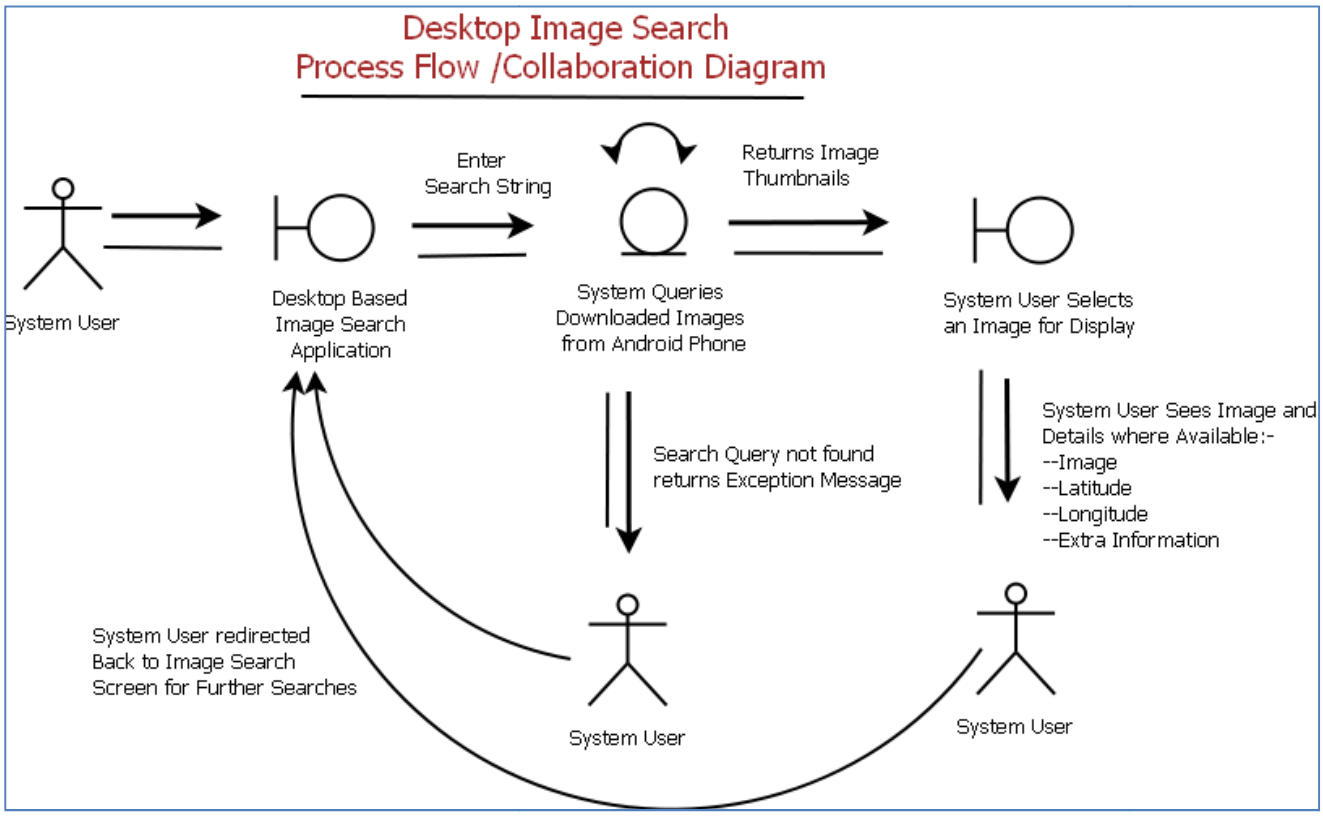


Figure 9: Collaboration Diagram - System User Opens ImageSearch Application (Desktop)

Exceptional Scenario:

- **Java.io.ImageWriteException** (Inability to Write into Image Designated Memory Address on Desktop)
- **Java.io.ImageReadException** (Inability to Read from Image Designated Memory Address on Desktop)
- **Java.io.IllegalAccessException** (If system, reads miscalculates Memory Address bounds, in order to read/display image, Java would return an Access Violation Exception, which will be caught, logged, and application will gracefully exit cleaning up resources in the finally block)
- **Java.io.FileNotFoundException (**Since the System User must ensure that image have been properly downloaded onto Desktop it is conceivable that an error on his part, would cause the system to be unable to find images through which to search through. Also if images are moved after application is searching through Image Collection, an error can arise)

GPSImageTagger Class Diagram

The class diagram for the three applications is presented below, with inheritance references being displayed using an arrow in a top-bottom hierarchical manner.

Application: GeoTag

Class: ExtraInfo

Class what serves to represent all of the User Defined Input saved inside the EXIF image. Since this was added to the EXIF Image Specification, a separate instance serves to delineate native image parameters with newly extended Image parameters. Contains all additional data our program adds to EXIF format, hence the name ExtraInfo, including GPS latitude, longitude and an instance of the **ExtraInfoData** Class.

Class: ExtraInfoData

Class that serves to represent only the User Inputted Extra Information saved in the EXIF image. This further aids in delineation of the native data segments of the image.

Class: TakeSnap

Class that instantiates Location Listener, requesting GPS location updates to be received by it from the Android.Location.LocationListener, sets the Window Display Format and screen content, and registers event handlers for OnClick and OnKeyDown (<ENTER>) .

The OnKeyDown method invokes all requisite functions required for taking the picture, geo-coding the picture, prompting user for additional information, storing that additional information in the existing EXIF format, storing image on Android SD Card, and after successfully completing sequence of events, allowing user to take another picture.

Class: MyLocationListener

Class that implements Android.Location.LocationListener in getting GPS Location on demand, and also registering an asynchronous delegate onLocationChanged to receive location updates.

GeoTag Class Diagram

The Diagram below highlights visually the class inheritance structure.

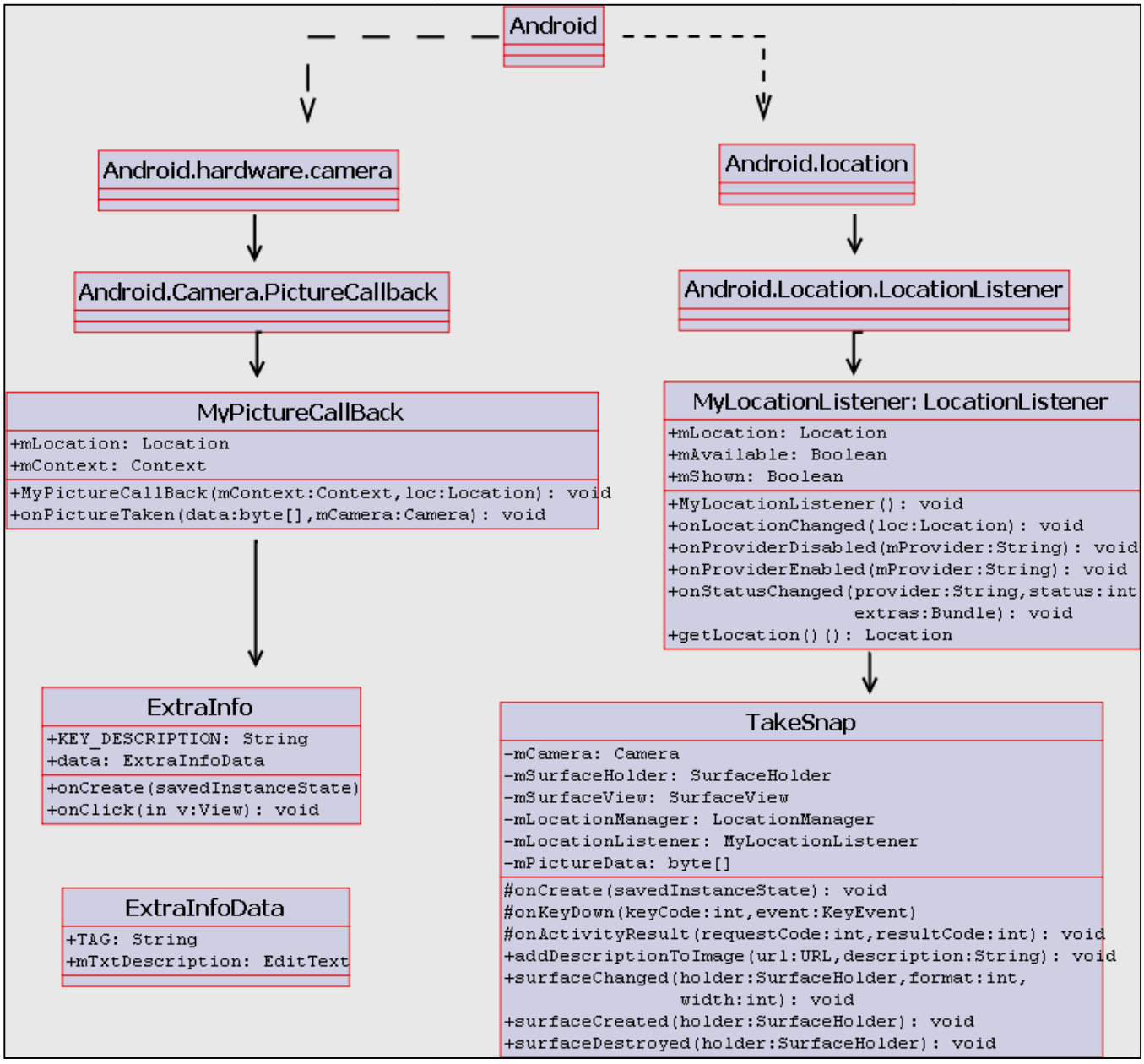


Figure 10: Class Diagram GeoTag: TakeSnap

Application: GeoImageView

The class structure in Android Image Search involves two main classes namely:-

ImageSearch

The class ImageSearch registers all Graphical User Interface (GUI) event handlers with their associated functions.

Its most important programmatic function is to set the directory of the Android SD Card to "/sdcard/DCIM/Camera" following which it instantiates the necessary EXIF reader class to read the Images in the SD card directory.

ImageAdapter

The class ImageAdapter extends the Android Widget BaseAdapter in addition to creating local copies of Image Metadata for each image in an Object Array. This is required when a specific image is selected, and all image metadata is to be displayed accordingly, but also since a portion of the metadata is the basis of the search query.

Class Diagram

The class diagram below visually presents the above class structure.

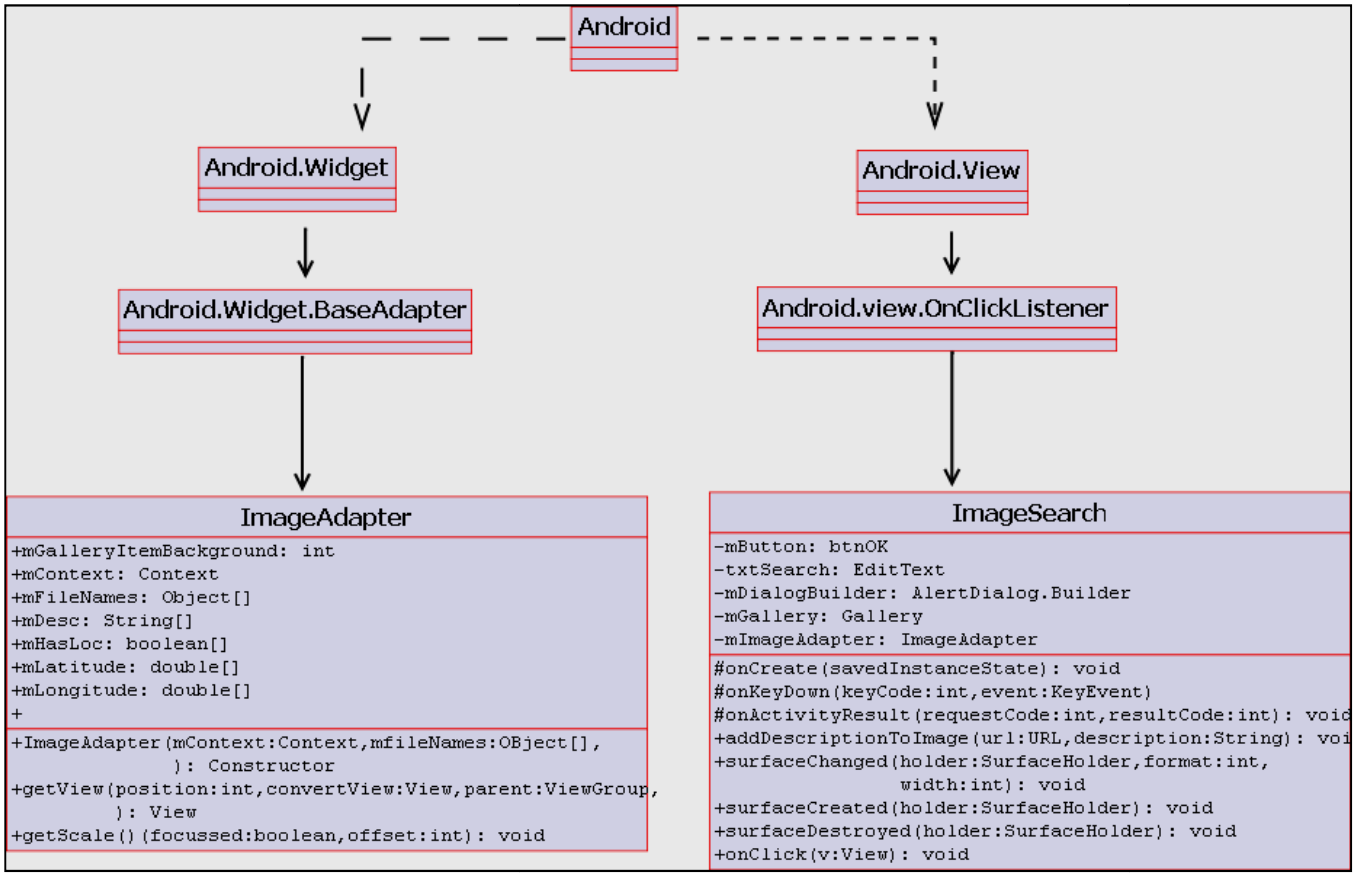


Figure 11: GeoImageView Class Diagram

ImageSearch

Class: ApplicationFrame

ApplicationFrame is the class that instantiates all visual elements, in their correct place, including the jSplitPanel, jScrollPanel, btnOpen, btnSearch while registering them to their proper event handlers. All elements of Application frame inherit from javax.swing and are standard swing objects intended to provide advanced functionality over standard java visual elements according to the Java Swing 3.1 Specification.

Class: ImageFileFilter

The ImageFileFilter class implements the FileNameFilter class restricting the ImageTypes to search through to the following as displayed by the picture.

```
String[] extensions = { "jpg", "JPG", "jpeg", "JPEG", "gif", "GIF","png", "PNG" }
for (String ext : extensions)
    if (name.endsWith(ext))
        return true;
return false;
```

Class: MyMouseAdapter

The MyMouseAdapter provides a custom way to intercept the MouseClick, in order to achieve core program objectives, which include displaying image metadata when image is clicked along with the image itself.

Class Diagram

All of the above is visually presented below:-

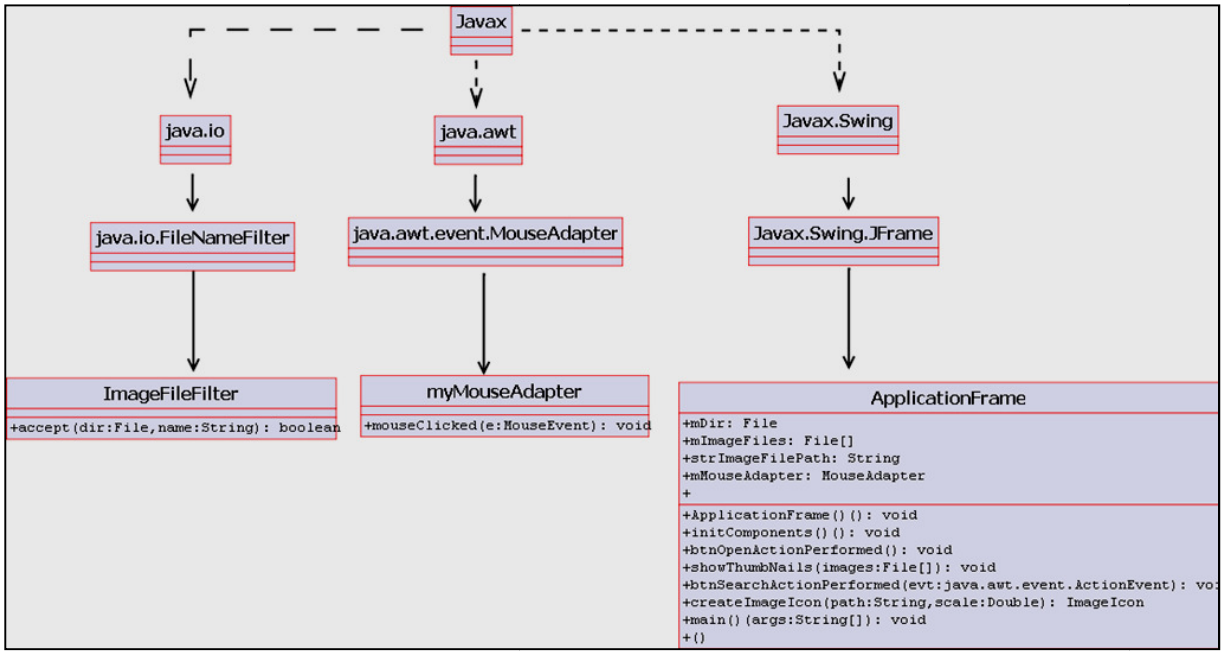


Figure 12: Desktop Based Image Search Class Diagram

GPSImageTagger Interfaces

There are three interfaces in the system:

- Interface A: Capture Image Interface
- Interface B: Store Meta Information Interface
- Interface C: Search Image-Meta Information

Each interface is equipped with its own exceptional scenario handling.

Interface Implementation Defaults

The interfaces A, B were implemented using Native Android Drawing Area Classes that derived from the Android. Widget base class including the following:-

- Android.Widget.Button
- Android.Widget.EditText
- Android.Widget.Gallery
- Android.Widget.ImageView

The interface C was implemented using Sun's Microsystems Java Foundation Classes (JFC) **Swing** Toolkit that were designed to provide a more advanced Application Programming Interface than the earlier Java sponsored **Abstract Window Toolkit (AWT).**

Interface A: Capture Image

The interface represents the default Android Emulator's Screen capture facility, represented as a screen of checkered squares with a smaller horizontal grey area that bounces off all the edges, but freezes when an image is taken. As with all other interfaces, the default android date time, wireless connectivity information and battery display is viewable on the top right.

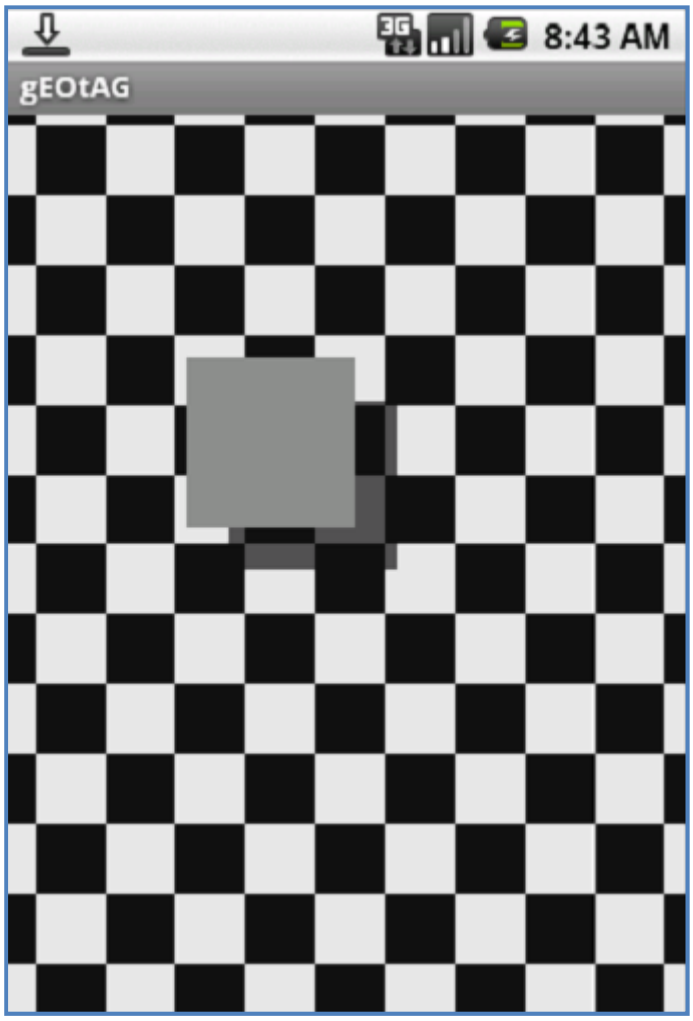


Figure 13: Capture Image Interface

Interface B: Store Image Metadata Information

The interface allows for the entry of image or environment related information for every image taken. It appears without user input and if the button Save is entered, stores the relevant user input into the image‘s metadata in EXIF format.

If the cancel button is entered, an empty string is stored for the images metadata.

Figure 14: Enter Image Specific Metadata

Interface C: Search Image-Meta Information

This screen allows for the input of search text to query the image collection on the phone for, in terms of user inputted metadata. The images returned as a result of this query are returned using the default Android Image Display Object – the Horizontal Image Thumbnail preview Screen.

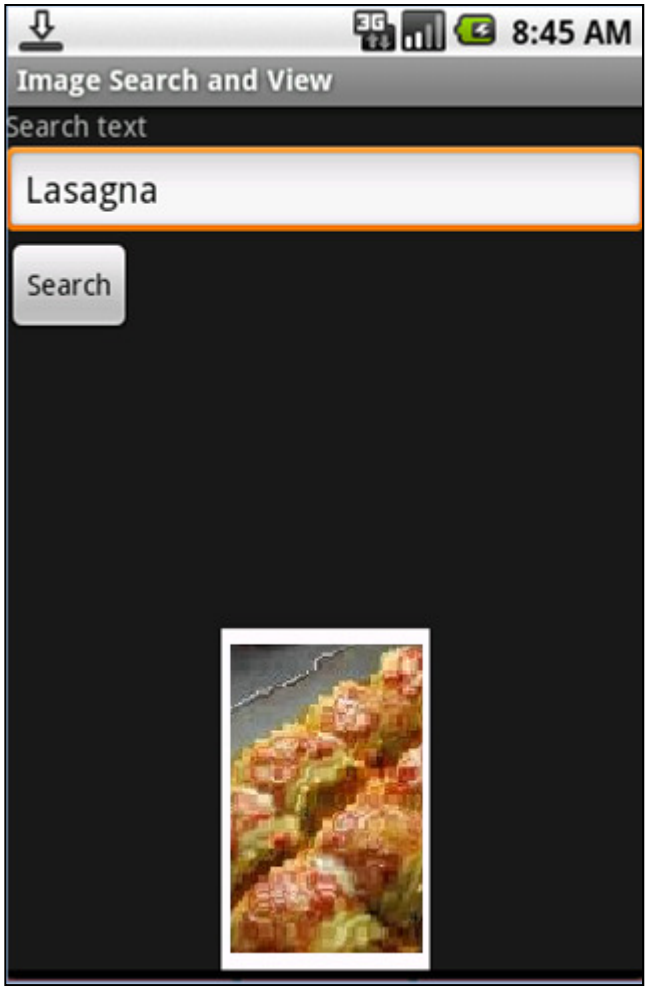


Figure 15: Search Image Collection by Metadata

Interface: Exceptional Scenario

If no image is returned in the query the following error message is displayed. The ok button would exit the Exceptional Scenario Screen and return the user to the Image Search and View Interface.

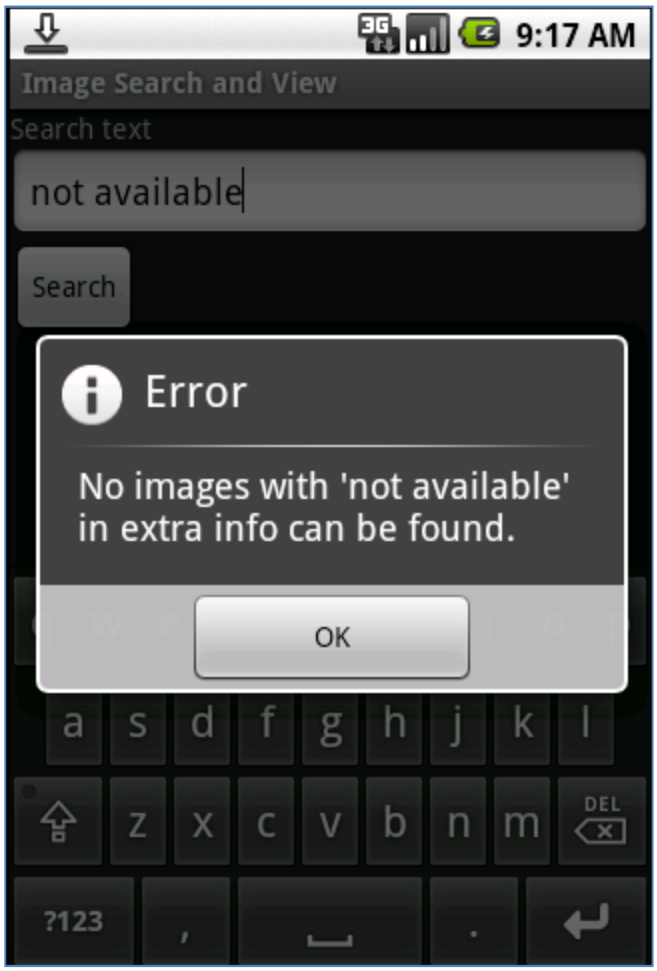


Figure 16. Exceptional Scenario

GPSImageTagger - Live Screenshots

Android: Installation & Program Menu

Installation of the Program is implicitly done by installing the .APK files on the Android device. For debugging purposes installation is done directly through the Eclipse API interface resulting in the following two icons being inserted into the Android default menu screen.

Below is a snapshot of a live installation on the Android Emulator.

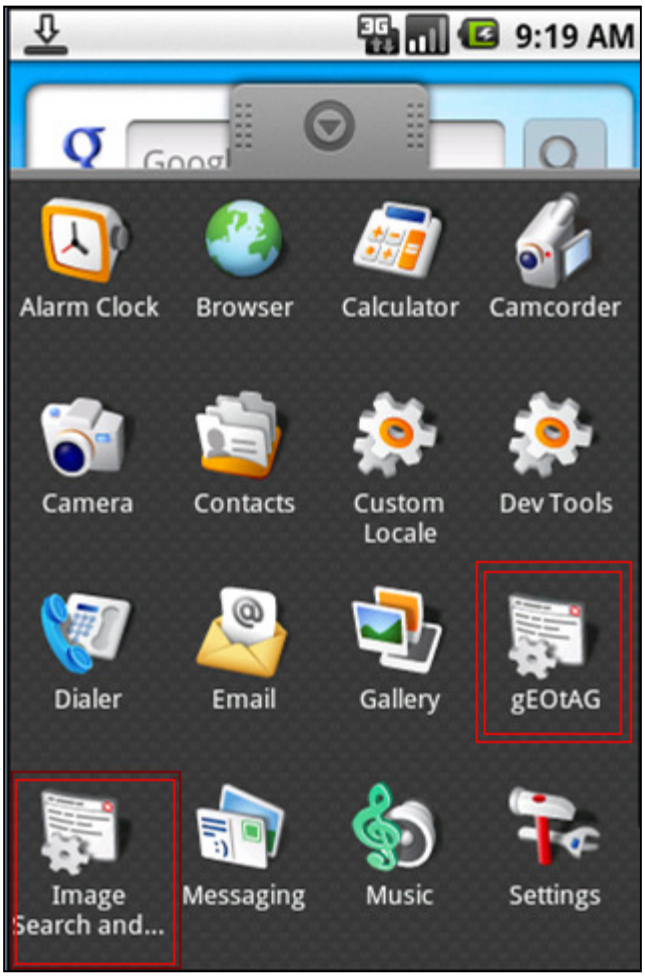


Figure 17: Installation and Program Menu Shortcut Positioning

Android: GeoImageView

The following is a live snapshot on the Android emulator of the program being asked to Query all images on the SD Card that were tagged with the word ‘Lasagna’ where the word must have been a part of the User Input after the picture was taken.

Figure 18: Image Search and View

GPSImageTagger - Software Tools Used

Eclipse

Eclipse is a multi-language software development environment comprising an IDE and a plug-in system to extend it. It is written primarily in Java and can be used to develop applications in Java and, by means of the various plug-ins, in other languages as well, including C, C++, COBOL, Python, Perl, PHP, and others.

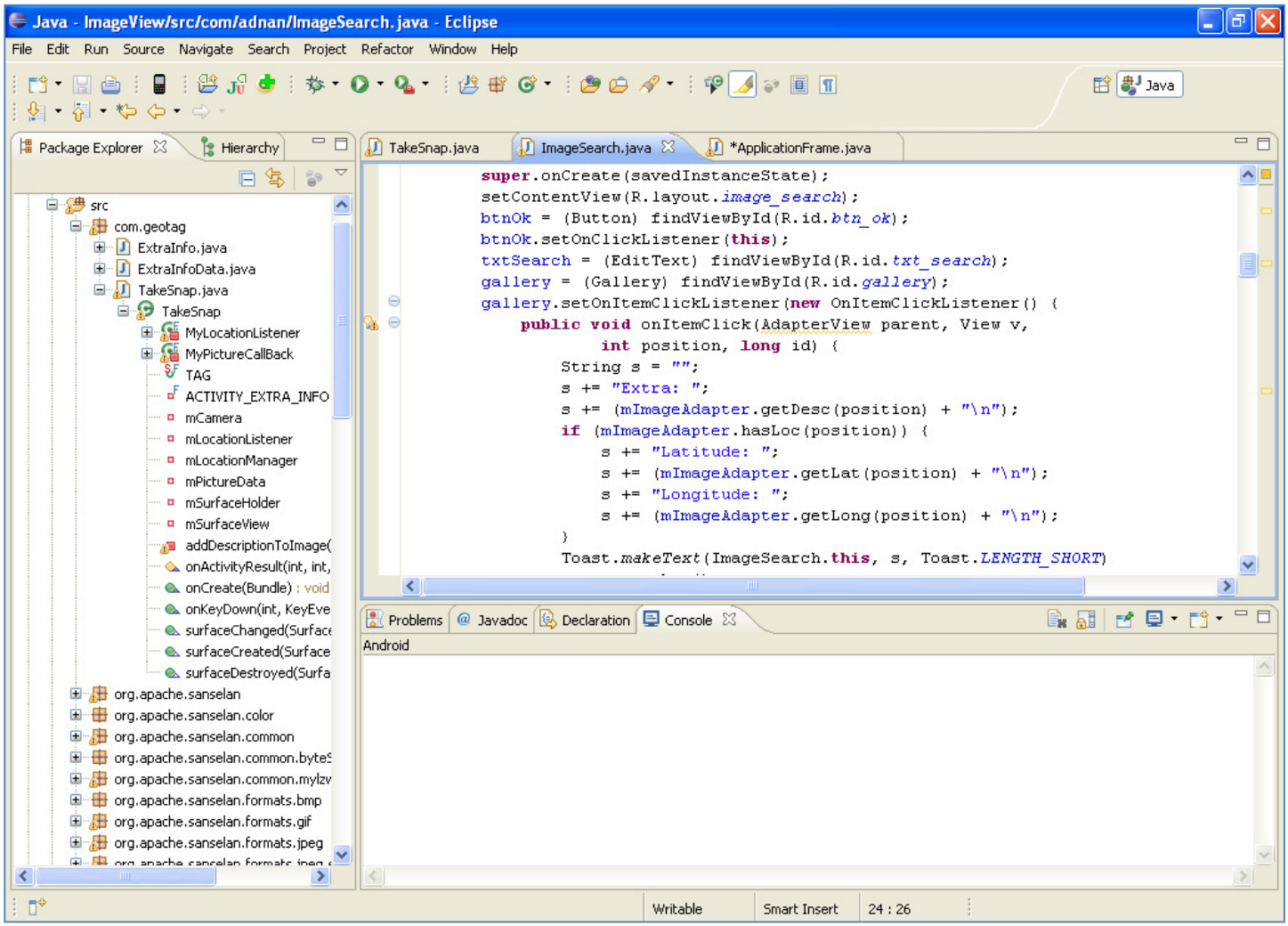


Figure 19: Eclipse Live Development Snapshot

DIA

Dia is free and open source general-purpose diagramming software, developed as part of the GNOME project's office suite and was originally created by Alexander Larsson. Dia uses a controlled single document interface (CSDI).

Dia has a modular design with several shape packages available for different needs: flowchart, network diagrams, circuit diagrams, and more. It does not restrict symbols and connectors from various categories from being placed together.

Dia can be used to draw many different kinds of diagrams. It has special objects to help draw entity-relationship models (tedia2sql can be used to create the SQL DDL), Unified Modeling Language (UML) diagrams, flowcharts, network diagrams, and simple electrical circuits. It is also possible to add support for new shapes by writing simple XML files, using a subset of Scalable Vector Graphics (SVG) to draw the shape.

Dia loads and saves diagrams to a custom XML format.

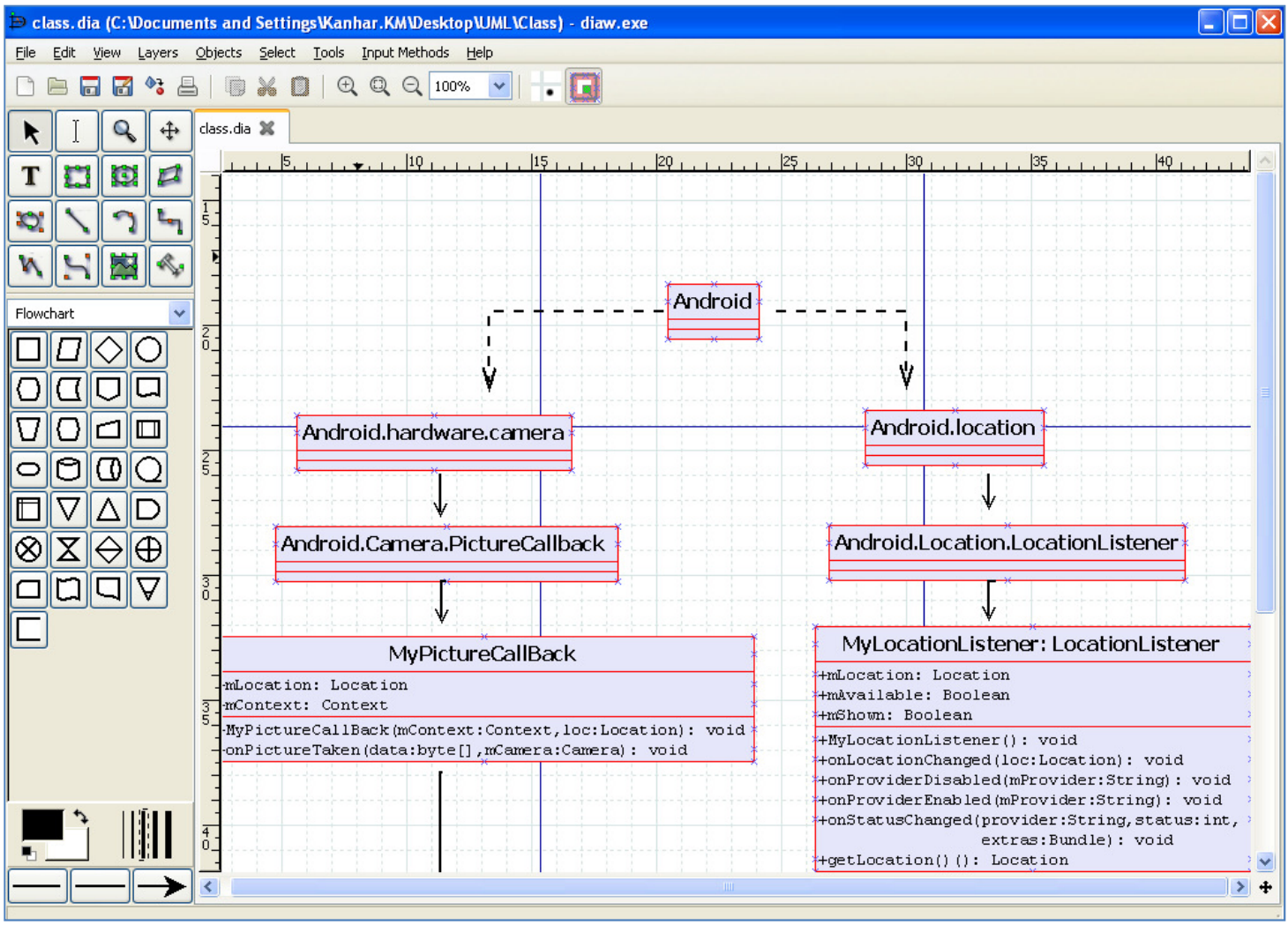


Figure 20: Dia Class Diagram Development Environment

Android Emulator

Android is a mobile operating system running on the Linux kernel. It was initially developed by Android Inc., a firm later purchased by Google, and lately by the Open Handset Alliance. It allows developers to write managed code in the Java language, controlling the device via Google-developed Java libraries.

The unveiling of the Android distribution on 5 November 2007 was announced with the founding of the Open Handset Alliance, a consortium of 47 hardware, software, and telecom companies devoted to advancing open standards for mobile devices. Google released most of the Android code under the Apache License, a free software and open source license.

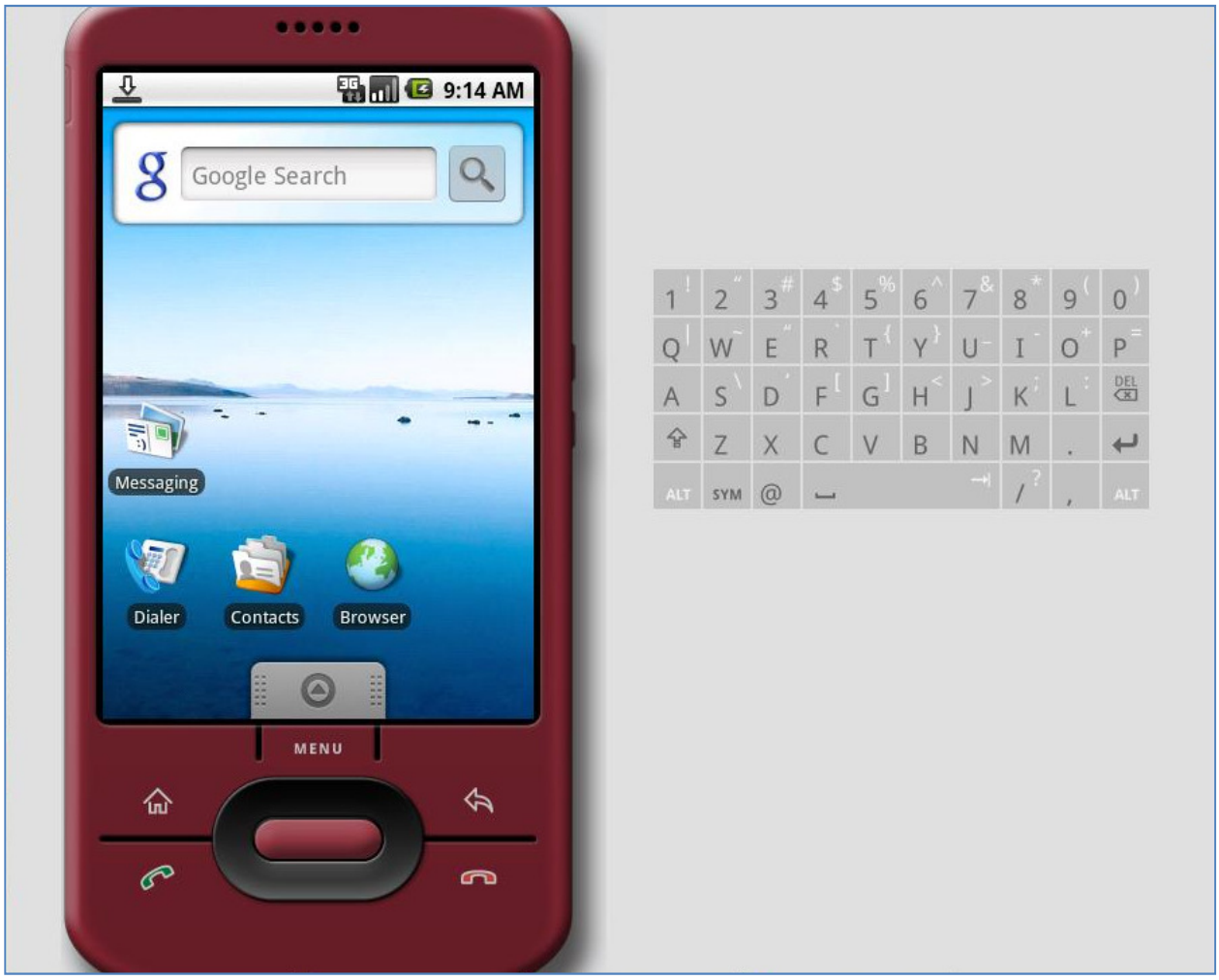


Figure 21: Development Environment Snapshot of Emulator

GPSImageTagger Application Authors

Kanhar Munshi and Asher Snyder are in their final year as Computer Science students at the City College of the City University of New York (CCNY).

They were mentored in this project by Professor **Izidor Gertner**, a professor at the Computer Science Department at CCNY over the course of two semesters to assist with initial project research concerning Real Time Operating Systems (RTOS) specifically with respect to Android and Moblin, and who guided is in choosing the right design paradigm, operating system and software language for our project, and who constantly mentored us with respect to application documentation, specifications and implementation.

Further background information on the authors is presented and categorized by author below:-

Kanhar Munshi

A software consultant for the New York City Department of Health and Environmental Surveillance building, deploying and maintaining ASP.NET web applications, Kanhar was Project Manager in deploying New York City's version of the Environment Public Health Tracking Network in collaboration with the Center of Disease Control (CDC) National EPHT Network.

The website is live and viewable at https://gis.nyc.gov/doh/track/

Kanhar is currently involved in the building of a Large Scale implementation of a Normalized Data Store, Data Warehouse for the Department of Health in better analyzing environmental indicators, global warming and other primary data plotted against health related indicators.

Asher Snyder

A programmer since the age of 11 who started programming professionally by the time he was 14, Asher is one of the inventors and driving personality behind the NOLOH Application Development Platform (http://www.noloh.com/). A "technological polymath," Asher has experience working with a vast number of different programming languages and development methodologies from the desktop out to the Internet, to maintaining servers and designing, implementing, and maintaining advanced databases.

Prior projects include the Automated Boiler Inspection system for Controlled Combustion of New York, and the OASIS system for the State of Virginia.

Asher is known for his confidence in his beliefs about the right way to develop, which serves as a major drive for his creativity.

Appendix A: Code Listing GeoTag

Class: TakeSnap

```
package com.geotag;

import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;

import org.apache.sanselan.ImageReadException;
import org.apache.sanselan.ImageWriteException;
import org.apache.sanselan.Sanselan;
import org.apache.sanselan.common.IImageMetadata;
import org.apache.sanselan.formats.jpeg.JpegImageMetadata;
import org.apache.sanselan.formats.jpeg.exifRewrite.ExifRewriter;
import org.apache.sanselan.formats.tiff.TiffImageMetadata;
import org.apache.sanselan.formats.tiff.constants.ExifTagConstants;
import org.apache.sanselan.formats.tiff.constants.TiffConstants;
import org.apache.sanselan.formats.tiff.constants.TiffFieldTypeConstants;
import org.apache.sanselan.formats.tiff.write.TiffOutputDirectory;
import org.apache.sanselan.formats.tiff.write.TiffOutputField;
import org.apache.sanselan.formats.tiff.write.TiffOutputSet;

import android.app.Activity;
import android.content.ContentValues;
import android.content.Context;
import android.content.Intent;
import android.database.Cursor;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.PixelFormat;
import android.hardware.Camera;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.location.LocationProvider;
import android.net.Uri;
import android.os.Bundle;
import android.provider.MediaStore;
import android.util.Log;
import android.view.KeyEvent;
import android.view.SurfaceHolder;
import android.view.SurfaceView;

public class TakeSnap extends Activity implements SurfaceHolder.Callback {
      public static final String TAG = "TAKE_SNAP";

      private final int ACTIVITY_EXTRA_INFO = 0;

      private Camera mCamera;
      private SurfaceHolder mSurfaceHolder;
      private SurfaceView mSurfaceView;
      private LocationManager mLocationManager;
      private MyLocationListener mLocationListener;
      private byte[] mPictureData;
```

```
/** Called when the activity is first created. */
//@Override
public void onCreate(Bundle savedInstanceState) {
	super.onCreate(savedInstanceState);
	Log.i(TAG, "onCreate");

	// For the moment I am only interested in GPS updates
	mLocationManager = (LocationManager)
	getSystemService(Context.LOCATION_SERVICE);
	mLocationListener = new MyLocationListener();
	try {
		mLocationManager.requestLocationUpdates(
				LocationManager.GPS_PROVIDER, 1000, 0, mLocationListener);
	} catch (IllegalArgumentException e) {
		// TODO: show a dialog that GPS is not availabe and exit
		System.exit(1);
	}

	getWindow().setFormat(PixelFormat.JPEG);
	setContentView(R.layout.take_snap);
	mSurfaceView = (SurfaceView) findViewById(R.id.review_surface);

	mSurfaceHolder = mSurfaceView.getHolder();
	mSurfaceHolder.addCallback(this);
	mSurfaceHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
	// setContentView(R.layout.main);
}

//@Override
public boolean onKeyDown(int keyCode, KeyEvent event) {
	if (keyCode == KeyEvent.KEYCODE_DPAD_CENTER) {
		Camera.Parameters p = mCamera.getParameters();
		p.remove("gps-latitude");
		p.remove("gps-longitude");
		p.remove("gps-altitude");
		p.remove("gps-timestamp");

		Location loc = mLocationListener.getLocation();
		if (loc != null) {
			p.set("gps-latitude", "" + loc.getLatitude());
			p.set("gps-longitude", "" + loc.getLongitude());
			if (loc.hasAltitude()) {
				p.set("gps-altitude", String.valueOf(loc.getAltitude()));
			} else {
				// for NETWORK_PROVIDER location provider, we may have
				// no altitude information, but the driver needs it, so
				// we fake one.
				p.set("gps-altitude", "0");
			}
			if (loc.getTime() != 0) {
				// Location.getTime() is UTC in milliseconds.
				// gps-timestamp is UTC in seconds.
				long utcTimeSeconds = loc.getTime() / 1000;
				p.set("gps-timestamp", String.valueOf(utcTimeSeconds));
			}
			mCamera.setParameters(p);
		} else
			Log.d(TAG, "location is null");
		mCamera.takePicture(null, null, new MyPictureCallBack(loc, this));
	}
	return super.onKeyDown(keyCode, event);
}
```

```
//@Override
protected void onActivityResult(int requestCode, int resultCode,
            Intent intent) {
      Bundle extras = intent.getExtras();

      String description = "";
      switch (requestCode) {
      case ACTIVITY_EXTRA_INFO:
            Log.d(TAG, "ACTIVITY_EXTRA_INFO returned");

            if (resultCode != RESULT_CANCELED) {
                  description = extras.getString(ExtraInfo.KEY_DESCRIPTION);
                  Log.d(TAG, "result != RESULT_CANCELED");
            }
            if (description == null)
                  Log.d(TAG, "description is null");
            if (description.equals(""))
                  Log.d(TAG, "descritpion is empty");
            ContentValues values = new ContentValues(3);
            values.put(MediaStore.Images.Media.DISPLAY_NAME, "GeoTag Picture");
            values.put(MediaStore.Images.Media.DESCRIPTION, description);
            values.put(MediaStore.Images.Media.MIME_TYPE, "image/jpeg");
            Uri uri = getContentResolver().insert(
            MediaStore.Images.Media.EXTERNAL_CONTENT_URI, values);

            Bitmap bitmap = BitmapFactory.decodeByteArray(mPictureData, 0,
                        mPictureData.length);

            try {
                  OutputStream outputStream = getContentResolver()
                              .openOutputStream(uri);
                  bitmap.compress(Bitmap.CompressFormat.JPEG, 85, outputStream);
                  outputStream.flush();
                  outputStream.close();

                  try {
                        addDescriptionToImage(uri, description);
                  } catch (Exception e) {
                        // TODO Auto-generated catch block
                        Log.d(TAG,"Exception in addDescriptionToImage()");
                        if(e.getMessage()!=null){
                              Log.d(TAG,e.getMessage());
                        }
                  }

                  setResult(RESULT_OK);
            } catch (FileNotFoundException e) {
                  Log.e(TAG, "Can not open file to write image");
                  setResult(RESULT_FIRST_USER);
            } catch (IOException e) {
                  Log.e(TAG, "Error closing ouput stream");
                  setResult(RESULT_FIRST_USER);
            }

            finish();
            break;
      }

}

private void addDescriptionToImage(Uri uri, String description) {
      File tempFile = null;
```

```
String fileName = null;
String[] projection = { MediaStore.Images.ImageColumns.DATA };

Cursor c = managedQuery(uri, projection, null, null, null);
if (c != null && c.moveToFirst())
      fileName = c.getString(0);
Log.d(TAG, "File Name= " + fileName);

File file = new File(fileName);

IImageMetadata metadata = null;
JpegImageMetadata jpegMetadata = null;
TiffImageMetadata exif = null;
OutputStream os = null;
TiffOutputSet outputSet = new TiffOutputSet();

try {
      metadata = Sanselan.getMetadata(file);
} catch (ImageReadException ire) {
      Log.d(TAG, "ImageReadException");
} catch (IOException ioe) {
      Log.d(TAG, "IOException");
}
// establish jpegMedatadata
if (metadata != null) {
      jpegMetadata = (JpegImageMetadata) metadata;
}

// establish exif
if (jpegMetadata != null) {
      exif = jpegMetadata.getExif();
}

// establish outputSet
if (exif != null) {
      try {
            outputSet = exif.getOutputSet();
      } catch (ImageWriteException e) {
            Log.d(TAG, "ImageWriteException on getOutputSet");
      }
}
if (outputSet != null) {
      // check if field already EXISTS - if so remove
      TiffOutputField imageDescription = outputSet
                  .findField(TiffConstants.EXIF_TAG_IMAGE_DESCRIPTION);
      if (imageDescription != null) {
outputSet.removeField(TiffConstants.EXIF_TAG_IMAGE_DESCRIPTION);
      }

      // add field
      try {
            imageDescription = new TiffOutputField(
                        ExifTagConstants.EXIF_TAG_IMAGE_DESCRIPTION,

      TiffFieldTypeConstants.FIELD_TYPE_ASCII, description
                                    .length(), description.getBytes());
            TiffOutputDirectory exifDirectory = outputSet
                        .getOrCreateExifDirectory();
            exifDirectory.add(imageDescription);
            Location loc = mLocationListener.getLocation();
            if (loc != null) {
                  outputSet.setGPSInDegrees(loc.getLongitude(),
            loc.getLatitude());
```

```
                }

            } catch (ImageWriteException e) {
                Log.d(TAG, "ImageWriteException on new TiffOutputField()");
            }

        }

        // create stream using temp file for dst
        try {
            tempFile = File.createTempFile(
                        "temp-" + System.currentTimeMillis(), ".jpeg");
            os = new FileOutputStream(tempFile);
            os = new BufferedOutputStream(os);
        } catch (IOException e) {
            Log.d(TAG, "IOException creating temp file");
        }

        // write/update EXIF metadata to output stream
        try {
            new ExifRewriter().updateExifMetadataLossless(file, os, outputSet);
        } catch (ImageReadException e) {
            Log.d(TAG, "ImageReadException - " + e.getMessage());
        } catch (ImageWriteException e) {
            Log.d(TAG, "ImageWriteException - " + e.getMessage());
        } catch (IOException e) {
            Log.d(TAG, "IOException - " + e.getMessage());
        } finally {
            if (os != null) {
                try {
                    os.close();
                } catch (IOException e) {
                }
            }
        }

        // copy temp file over original file
        String s = file.getAbsolutePath();
        file.delete();
        tempFile.renameTo(file);

    }

    //@Override
    public void surfaceChanged(SurfaceHolder holder, int format, int width,
            int height) {
        try {
            mCamera.stopPreview();
        } catch (Exception e1) {
            Log.d(TAG, "stopPreview exception");
        }
        Camera.Parameters p = mCamera.getParameters();
        p.setPreviewSize(width, height);
        mCamera.setParameters(p);
        try {
            mCamera.setPreviewDisplay(holder);
        } catch (IOException ioe) {
            Log.d(TAG, "setPreviewDisplayFailed");
            Log.d(TAG, ioe.getMessage());
        }
        mCamera.startPreview();
    }
```

```
//@Override
public void surfaceCreated(SurfaceHolder holder) {
	mCamera = Camera.open();
}

//@Override
public void surfaceDestroyed(SurfaceHolder holder) {
	mCamera.stopPreview();
	mCamera.release();

}

private final class MyPictureCallBack implements Camera.PictureCallback {
	private Location mLocation;
	private Context mContext;

	public MyPictureCallBack(Location loc, Context c) {
		mLocation = loc;
		mContext = c;
	}

	//@Override
	public void onPictureTaken(byte[] data, Camera camera) {
		if (data != null) {
			Log.d(TAG, "data is not null, length=" + data.length);

			// save picture data and start second activity to get extra info
			mPictureData = data;

			startActivityForResult(new Intent(mContext, ExtraInfo.class),
					ACTIVITY_EXTRA_INFO);
		} else
			Log.d(TAG, "data is null");
	}

}

private final class MyLocationListener implements LocationListener {
	Location mLocation;
	boolean mAvailabe;
	boolean shown;

	public MyLocationListener() {
		mLocation = new Location(LocationManager.GPS_PROVIDER);
		mAvailabe = false;
		shown = false;
	}

	//@Override
	public void onLocationChanged(Location newLocation) {
		mLocation.set(newLocation);
	Log.d(TAG, "new location lat="+newLocation.getLatitude()+"
	long="+newLocation.getLongitude());
	}

	//@Override
	public void onProviderDisabled(String provider) {
		mAvailabe = false;
	}

	//@Override
	public void onProviderEnabled(String provider) {
		mAvailabe = true;
```

```
            }

            //@Override
                  public void onStatusChanged(String provider, int status, Bundle extras)
                  {
                  switch (status) {
                  case LocationProvider.OUT_OF_SERVICE:
                  case LocationProvider.TEMPORARILY_UNAVAILABLE:
                        mAvailabe = false;
                  case LocationProvider.AVAILABLE:
                        mAvailabe = true;
                  }
            }

            Location getLocation() {
                  return mAvailabe ? mLocation : null;
            }

      }
}
```

Class: ExtraInfo

```
package com.geotag;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

public class ExtraInfo extends Activity implements View.OnClickListener{
      // public class Class1{
//    }
      public static final String KEY_DESCRIPTION = "KEY_DESCRIPTION";
      ExtraInfoData data = new ExtraInfoData("ExtraInfo");

      //@Override
      protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            Log.d(data.TAG,"onCreate");
            setContentView(R.layout.extra_info);
            Button btnSave=(Button)findViewById(R.id.btn_save);
            Button btnCancel=(Button)findViewById(R.id.btn_cancel);
            data.mTxtDescription = (EditText)findViewById(R.id.txt_extra_info);
            btnSave.setOnClickListener(this);
            btnCancel.setOnClickListener(this);
      }

      //@Override
      public void onClick(View v) {
            // TODO Auto-generated method stub
            switch(v.getId()){
            case R.id.btn_save:
                  Intent intent=new Intent();
                  intent.putExtra(KEY_DESCRIPTION,
                  data.mTxtDescription.getText().toString());
                  Log.d(TakeSnap.TAG,data.mTxtDescription.getText().toString());
                  setResult(RESULT_OK, intent);
```

```
                  finish();
                  break;

            case R.id.btn_cancel:
            default:
                  setResult(RESULT_CANCELED);
                  finish();
                  break;
      }

      }


}
```

Class: ExtraInfoData

```
package com.geotag;

import android.widget.EditText;

public class ExtraInfoData {
      public String TAG;
      public EditText mTxtDescription;

      public ExtraInfoData(String tAG) {
            TAG = tAG;
      }
}
```

Appendix B: Code Listing GeoImageView

Class: ImageSearch

```
package com.ad;

import java.io.File;
import java.io.IOException;
import java.util.ArrayList;

import org.apache.sanselan.ImageReadException;
import org.apache.sanselan.Sanselan;
import org.apache.sanselan.common.IImageMetadata;
import org.apache.sanselan.formats.jpeg.JpegImageMetadata;
import org.apache.sanselan.formats.tiff.TiffField;
import org.apache.sanselan.formats.tiff.TiffImageMetadata;
import org.apache.sanselan.formats.tiff.constants.TiffConstants;

import com.adnan.R;

import android.app.Activity;
import android.app.AlertDialog;
import android.content.Context;
import android.content.Intent;
import android.content.res.TypedArray;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.os.Bundle;
```

```
import android.util.Log;
import android.view.View;
import android.view.ViewGroup;
import android.view.View.OnClickListener;
import android.widget.AdapterView;
import android.widget.BaseAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Gallery;
import android.widget.ImageView;
import android.widget.Toast;
import android.widget.AdapterView.OnItemClickListener;

public class ImageSearch extends Activity implements OnClickListener {
	String TAG = "IMGSEARCH";

	Button btnOk;
	EditText txtSearch;
	Gallery gallery;

	ImageAdapter mImageAdapter;
	AlertDialog.Builder mDialogbuilder;

	/** Called when the activity is first created. */
	//@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.image_search);
		btnOk = (Button) findViewById(R.id.btn_ok);
		btnOk.setOnClickListener(this);
		txtSearch = (EditText) findViewById(R.id.txt_search);
		gallery = (Gallery) findViewById(R.id.gallery);
		gallery.setOnItemClickListener(new OnItemClickListener() {
			public void onItemClick(AdapterView parent, View v,
					int position, long id) {
				String s = "";
				s += "Extra: ";
				s += (mImageAdapter.getDesc(position) + "\n");
				if (mImageAdapter.hasLoc(position)) {
					s += "Latitude: ";
					s += (mImageAdapter.getLat(position) + "\n");
					s += "Longitude: ";
					s += (mImageAdapter.getLong(position) + "\n");
				}
				Toast.makeText(ImageSearch.this, s, Toast.LENGTH_SHORT)
						.show();
			}
		});
		mDialogbuilder = new AlertDialog.Builder(this);
		mDialogbuilder.setTitle("Error");
		mDialogbuilder.setIcon(android.R.drawable.ic_dialog_info);
		mDialogbuilder.setPositiveButton("OK", null);
	}

	@Override
	public void onClick(View v) {
		String strSearch = txtSearch.getText().toString().trim();
		ArrayList<String> imgList = new ArrayList<String>();
		if (strSearch.equals("")) {
			mDialogbuilder.setMessage("Please enter search criteria");
			return;
		}
```

```
        File dir = new File("/sdcard/DCIM/Camera");

        for (File file : dir.listFiles()) {
              if (file.isDirectory())
                    continue;

              IImageMetadata metadata = null;
              JpegImageMetadata jpegMetadata = null;
              TiffImageMetadata exif = null;

              try {
                    metadata = Sanselan.getMetadata(file);
              } catch (ImageReadException ire) {
                    Log.d(TAG, "ImageReadException file=" + file.getName());
              } catch (IOException ioe) {
                    Log.d(TAG, "IOException file=" + file.getName());
              }
              // establish jpegMedatadata
              if (metadata != null) {
                    jpegMetadata = (JpegImageMetadata) metadata;
              }

              // establish exif
              if (jpegMetadata != null) {
                    exif = jpegMetadata.getExif();
              }

              if (exif != null) {
                    try {
                          TiffField desc = exif
                    .findField(TiffConstants.EXIF_TAG_IMAGE_DESCRIPTION);

                          if(desc!=null &&
                    desc.getStringValue().contains(strSearch)){
                                imgList.add(file.getAbsolutePath());
                          }

                    } catch (ImageReadException e) {
                          e.printStackTrace();
                    }
              }

        }
        if(imgList.size()==0){
              mDialogbuilder.setMessage("No images with '"+strSearch+"' in extra info
              can be found.");
              mDialogbuilder.show();
              return;
        }
        //mDialogbuilder.setMessage(msg);
        //mDialogbuilder.show();
        Log.d(TAG, "starting viewer list size="+imgList.size());
        mImageAdapter = new ImageAdapter(this, imgList.toArray());
        gallery.setAdapter(mImageAdapter);
    }
```

Class: ImageAdapter

```
class ImageAdapter extends BaseAdapter {
        int mGalleryItemBackground;
        private Context mContext;
        private Object[] mFileNames;
```

```
private String[] mDesc;
private boolean[] mHasLoc;
private double[] mLatitude;
private double[] mLongitude;
private int nothing;

public ImageAdapter(Context c, Object[] fileNames) {
	mContext = c;
	mFileNames = fileNames;
	mDesc = new String[mFileNames.length];
	mHasLoc = new boolean[mFileNames.length];
	mLatitude = new double[mFileNames.length];
	mLongitude = new double[mFileNames.length];

	for (int i = 0; i < fileNames.length; i++) {
		File file = new File(fileNames[i].toString());

		IImageMetadata metadata = null;
		JpegImageMetadata jpegMetadata = null;
		TiffImageMetadata exif = null;

		try {
			metadata = Sanselan.getMetadata(file);
		} catch (ImageReadException ire) {
			Log.d(TAG, "ImageReadException file=" + file.getName());
		} catch (IOException ioe) {
			Log.d(TAG, "IOException file=" + file.getName());
		}
		// establish jpegMedatadata
		if (metadata != null) {
			jpegMetadata = (JpegImageMetadata) metadata;
		}

		// establish exif
		if (jpegMetadata != null) {
			exif = jpegMetadata.getExif();
		}

		if (exif != null) {
			try {
				TiffImageMetadata.GPSInfo gpsInfo = exif.getGPS();
				TiffField desc = exif
	.findField(TiffConstants.EXIF_TAG_IMAGE_DESCRIPTION);

				mDesc[i] = desc.getStringValue();
				if (gpsInfo != null) {
					mHasLoc[i] = true;

				mLongitude[i] = gpsInfo.getLongitudeAsDegreesEast();
					mLatitude[i] =
				gpsInfo.getLatitudeAsDegreesNorth();

				}

			} catch (ImageReadException e) {
				e.printStackTrace();
			}
		}

	}

	TypedArray a = obtainStyledAttributes(R.styleable.Theme);
	mGalleryItemBackground = a.getResourceId(
```

```
                        R.styleable.Theme_android_galleryItemBackground, 0);
            a.recycle();
        }

        public String getDesc(int position) {
            return mDesc[position];
        }

        public boolean hasLoc(int position) {
            return mHasLoc[position];
        }

        public double getLat(int position) {
            return mLatitude[position];
        }

        public double getLong(int position) {
            return mLongitude[position];
        }

        public int getCount() {
            return mFileNames.length;
        }

        public Object getItem(int position) {
            return position;
        }

        public long getItemId(int position) {
            return position;
        }

        public View getView(int position, View convertView, ViewGroup parent) {
            try {
                ImageView i = new ImageView(mContext);
                Bitmap bm = BitmapFactory.decodeFile(mFileNames[position]
                            .toString());
                if (bm != null)
                    Log.d(TAG, "successful decdode file="
                                + mFileNames[position]);
                else
                    Log.d(TAG, "un-successful decdode file="
                                + mFileNames[position]);
                i.setImageBitmap(bm);
                i.setLayoutParams(new Gallery.LayoutParams(150, 170));
                i.setScaleType(ImageView.ScaleType.FIT_CENTER);
                Log.d(TAG, "returning ImageView");
                return i;
            } catch (Exception e) {
                Log.d(TAG, "error in getView()");
                if (e != null && e.getMessage() != null)
                    Log.d(TAG, e.getMessage());
                return null;
            }
        }

        public float getScale(boolean focused, int offset) {
            Log.d(TAG, "getScale()");
            /* Formula: 1 / (2 ^ offset) */
            return Math.max(0, 1.0f / (float) Math.pow(2, Math.abs(offset)));
        }
    }
```

Appendix C: Code Listing ImageSearch

Class: ApplicationFrame

```
import java.awt.Color;
import java.awt.Image;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.io.File;
import java.io.FilenameFilter;
import java.io.IOException;
import java.util.ArrayList;

import javax.swing.BorderFactory;
import javax.swing.ImageIcon;
import javax.swing.JFileChooser;
import javax.swing.JLabel;
import javax.swing.JOptionPane;

import org.apache.sanselan.ImageReadException;
import org.apache.sanselan.Sanselan;
import org.apache.sanselan.common.IImageMetadata;
import org.apache.sanselan.formats.jpeg.JpegImageMetadata;
import org.apache.sanselan.formats.tiff.TiffField;
import org.apache.sanselan.formats.tiff.TiffImageMetadata;
import org.apache.sanselan.formats.tiff.constants.TiffConstants;

import sun.rmi.runtime.Log;

public class ApplicationFrame extends javax.swing.JFrame {
	File mDir;
	File[] mImageFiles;
	String strImageFilePath;
	MouseAdapter mMouseAdapter;

	// ArrayList<String> imageList;

	/** Creates new form ApplicationFrame */
	public ApplicationFrame() {
		mMouseAdapter = new MyMouseAdapter();
		initComponents();
	}

	/**
	 * This method is called from within the constructor to initialize the form.
	 * WARNING: Do NOT modify this code. The content of this method is always
	 * regenerated by the Form Editor.
	 */
	private void initComponents() {

		jToolBar1 = new javax.swing.JToolBar();
		btnOpen = new javax.swing.JButton();
		jSeparator1 = new javax.swing.JToolBar.Separator();
		txtSearch = new javax.swing.JTextField();
		btnSearch = new javax.swing.JButton();
		jSplitPane1 = new javax.swing.JSplitPane();
		jScrollPane1 = new javax.swing.JScrollPane();
		pnlImagesList = new javax.swing.JPanel();
		lblImage = new javax.swing.JLabel();
		lblImage.addMouseListener(mMouseAdapter);
```

```
        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        jToolBar1.setFloatable(false);

        btnOpen.setText("Open");
        btnOpen.setFocusable(false);
        btnOpen.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
        btnOpen.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
        btnOpen.addActionListener(new java.awt.event.ActionListener() {
              public void actionPerformed(java.awt.event.ActionEvent evt) {
                    btnOpenActionPerformed(evt);
              }
        });
        jToolBar1.add(btnOpen);
        jToolBar1.add(jSeparator1);

        txtSearch.setColumns(15);
        txtSearch.setToolTipText("Enter your search criteria and the press Search");
        jToolBar1.add(txtSearch);

        btnSearch.setText("Search");
        btnSearch.setFocusable(false);
        btnSearch.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
        btnSearch.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
        btnSearch.addActionListener(new java.awt.event.ActionListener() {
              public void actionPerformed(java.awt.event.ActionEvent evt) {
                    btnSearchActionPerformed(evt);
              }
        });
        jToolBar1.add(btnSearch);

        jSplitPane1.setDividerLocation(120);

        pnlImagesList.setLayout(new java.awt.GridLayout(0, 1, 0, 2));
        jScrollPane1.setViewportView(pnlImagesList);

        jSplitPane1.setLeftComponent(jScrollPane1);

        lblImage.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jSplitPane1.setRightComponent(lblImage);

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(
                    getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(layout.createParallelGroup(
                    javax.swing.GroupLayout.Alignment.LEADING).addComponent(
                    jToolBar1, javax.swing.GroupLayout.DEFAULT_SIZE, 400,
                    Short.MAX_VALUE).addComponent(jSplitPane1,
                    javax.swing.GroupLayout.DEFAULT_SIZE, 400, Short.MAX_VALUE));
        layout
                    .setVerticalGroup(layout
                                .createParallelGroup(
                    javax.swing.GroupLayout.Alignment.LEADING)
                                .addGroup(
layout
.createSequentialGroup()
.addComponent(
jToolBar1,
javax.swing.GroupLayout.PREFERRED_SIZE, 25,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addPreferredGap(
javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addComponent(
```

```
jSplitPane1,
javax.swing.GroupLayout.DEFAULT_SIZE,269, Short.MAX_VALUE)));

      pack();
}// </editor-fold>

private void btnOpenActionPerformed(java.awt.event.ActionEvent evt) {
      final JFileChooser fc = new JFileChooser();
      // make sure user only selects directories and not files
      fc.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
      // if user didnt cancel then get the directory he selected and load
      // images from it
      if (fc.showOpenDialog(this) == JFileChooser.APPROVE_OPTION) {
            mDir = fc.getSelectedFile();
            // we hav the directory, get images and show them in the scroll

            // get only image files and ignore the rest
            mImageFiles = mDir.listFiles(new ImageFileFilter());

            // show them in the left pane
            showThumbNails(mImageFiles);

      }
}

private void showThumbNails(File[] images) {
      // clear the left pane of all previous thumb nails
      pnlImagesList.removeAll();
      pnlImagesList.setVisible(false);
      for (File image : images) {

            ImageIcon icon = createImageIcon(image.getAbsolutePath(), 100);
            if (icon != null) {
                  JLabel lbl = new JLabel(icon);
                  lbl.setBorder(BorderFactory.createLineBorder(Color.BLACK));
                  // we are using ImageName property to store the file path so we
                  can later use it when
                  // displaying bigger image or when we need to get extrainfo
                  // from file
                  lbl.setName(image.getAbsolutePath());
                  lbl.addMouseListener(mMouseAdapter);
                  pnlImagesList.add(lbl);
            } else
                  System.err.println("Image Icon null for "
                              + image.getAbsolutePath());
      }
      pnlImagesList.setVisible(true);
      this.repaint();
}

private void btnSearchActionPerformed(java.awt.event.ActionEvent evt) {
      // TODO add image searching code here:
      String strSearch = txtSearch.getText().trim();
      ArrayList<File> imgList = new ArrayList<File>();
      if (strSearch.equals("")) {
            JOptionPane.showMessageDialog(this,
            "Please enter search criteria");
            return;
      }

      // File dir = new File("/sdcard/DCIM/Camera");

      for (File file : mImageFiles) {
```

```
			if (file.isDirectory())
				continue;

			IImageMetadata metadata = null;
			JpegImageMetadata jpegMetadata = null;
			TiffImageMetadata exif = null;

			try
			{
				metadata = Sanselan.getMetadata(file);
			} catch (ImageReadException ire) {
				System.err.println("ImageReadException file=" + file.getName());
			} catch (IOException ioe) {
				System.err.println("IOException file=" + file.getName());
			}
			// establish jpegMedatadata
			if (metadata != null) {
				jpegMetadata = (JpegImageMetadata) metadata;
			}

			// establish exif
			if (jpegMetadata != null) {
				exif = jpegMetadata.getExif();
			}

			if (exif != null) {
				try {
					TiffField desc = exif
		.findField(TiffConstants.EXIF_TAG_IMAGE_DESCRIPTION);

					if (desc != null
							&& desc.getStringValue().contains(strSearch)) {
						imgList.add(file);
					}

				} catch (ImageReadException e) {
					e.printStackTrace();
				}
			}

		}
		if (imgList.size() == 0) {
			JOptionPane.showMessageDialog(this, "No images with '" + strSearch
					+ "' in extra info can be found.");
			return;
		}

		showThumbNails((File[]) imgList.toArray());
	}

	/** Returns an ImageIcon, or null if the path was invalid. */
	protected static ImageIcon createImageIcon(String path, double scale) {
		ImageIcon i = new ImageIcon(path);
		Image img = i.getImage();
		int width = img.getWidth(null);
		int height = img.getHeight(null);
		int newWidth = (int) ((width > height) ? scale
				: (width * scale / height));
		int newHeight = (int) ((width > height) ? (height * scale / width)
				: scale);
		i.setImage(img.getScaledInstance(newWidth, newHeight,
				Image.SCALE_SMOOTH));
		return i;
```

```
	}
```

Class: ImageFilter

```
class ImageFileFilter implements FilenameFilter {
		public boolean accept(File dir, String name) {
			String[] extensions = { "jpg", "JPG", "jpeg", "JPEG", "gif",
"GIF","png", "PNG" };
			for (String ext : extensions)
				if (name.endsWith(ext))
					return true;
			return false;
		}
	}
```

Class: MyMouseAdapter

```
class MyMouseAdapter extends MouseAdapter {
		@Override
		public void mouseClicked(MouseEvent e) {
			super.mouseClicked(e);
			JLabel lbl = (JLabel) e.getSource();
			String info = "No exif meta data in the image";
			if (lbl != lblImage) {
				strImageFilePath = lbl.getName();
				Image img = ((ImageIcon) lbl.getIcon()).getImage();
				double scale = (img.getWidth(null) > img.getHeight(null)) ?
				lblImage
							.getWidth()
							: lblImage.getHeight();
				lblImage.setIcon(createImageIcon(strImageFilePath, scale));
			} else if (e.getButton() == MouseEvent.BUTTON3) {
				//Display extra info and location info here
				File file = new File(strImageFilePath);

				IImageMetadata metadata = null;
				JpegImageMetadata jpegMetadata = null;
				TiffImageMetadata exif = null;

				try {
					metadata = Sanselan.getMetadata(file);
				} catch (ImageReadException ire) {
					System.err.println("ImageReadException file="
								+ file.getName());
				} catch (IOException ioe) {
					System.err.println("IOException file=" + file.getName());
				}
				// establish jpegMedatadata
				if (metadata != null) {
					jpegMetadata = (JpegImageMetadata) metadata;
				}

				// establish exif
				if (jpegMetadata != null) {
					exif = jpegMetadata.getExif();
				}

				if (exif != null) {
					try {
```

```
                        TiffImageMetadata.GPSInfo gpsInfo = exif.getGPS();
                        TiffField desc =
                  exif.findField(TiffConstants.EXIF_TAG_IMAGE_DESCRIPTION);
                        info="";
                        if (desc != null)
                              info += ("Extra Info: " + desc.getStringValue()
                  + "\n");
                        else
                              info += ("No extra info in the image.\n");

                        if (gpsInfo != null) {
                              info += ("Longitude: "
                                          +
                  gpsInfo.getLongitudeAsDegreesEast() + "\n");
                              info += ("Latitude: "
                                          +
                  gpsInfo.getLatitudeAsDegreesNorth() + "\n");
                        } else
                              info += "No location info in file.\n";

                  } catch (ImageReadException ire) {
                        System.err.println(ire.getMessage());
                  }
            }
            JOptionPane.showMessageDialog(ApplicationFrame.this, info);
      }
   }
}
```

Class: Main

```
public class Main {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        new ApplicationFrame().setVisible(true);
    }

}
```