Code Design and Quantify

Zilong Wang

A01001507

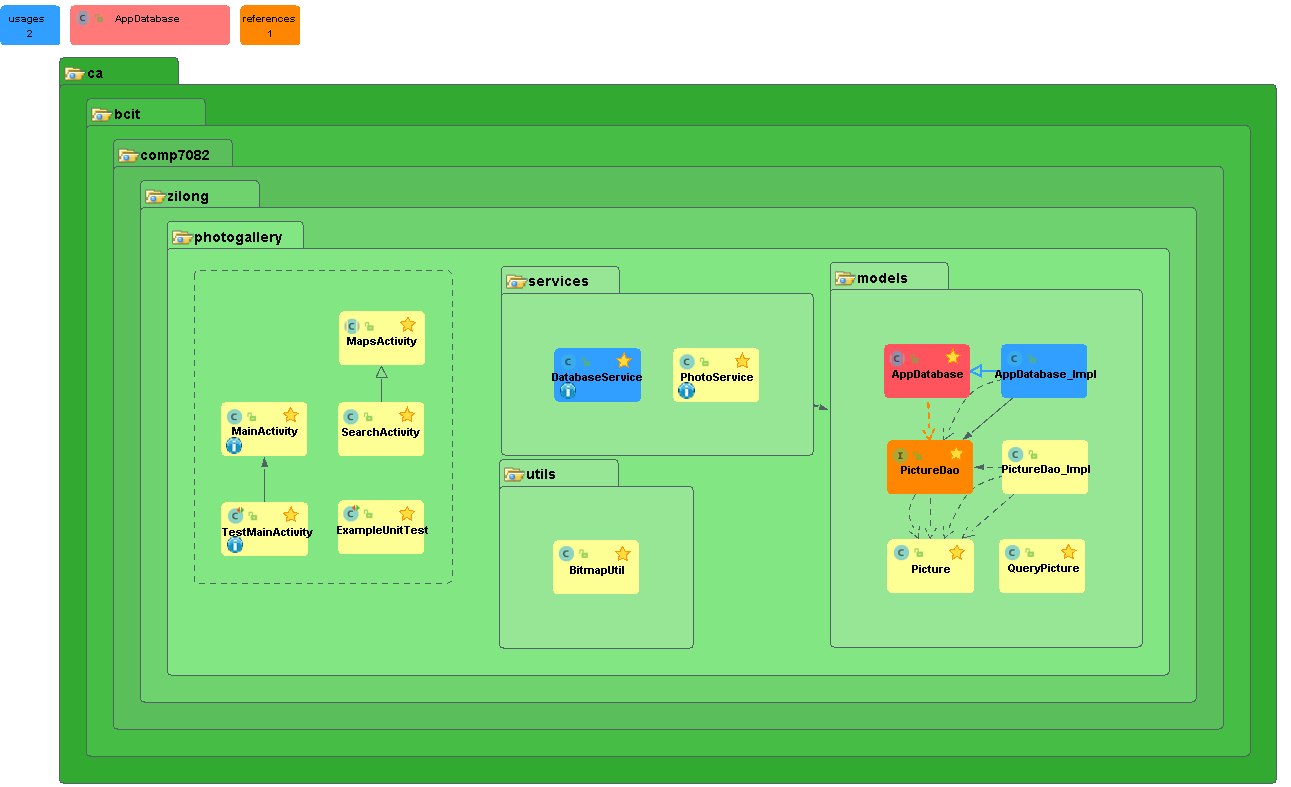
COMP 7082

BCIT

This document describes the code design of Photo Gallery app and the steps to quantify and improve code quality.

# Packages

Figure 1 shows the packages of Photo Gallery app.



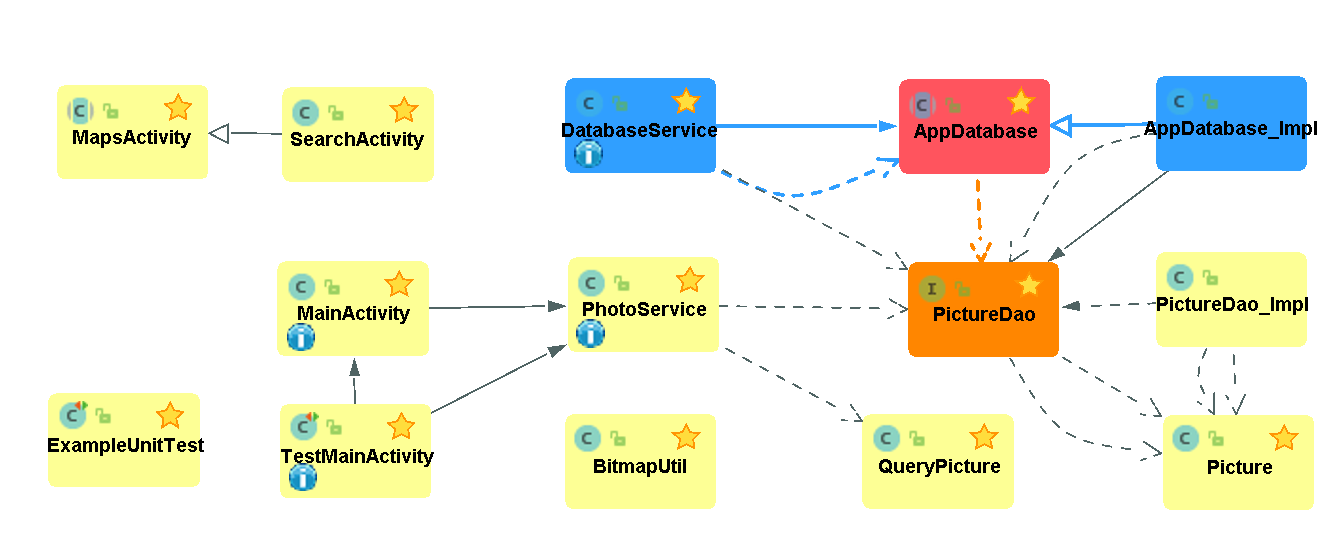
*Figure 1*. Package Diagram of Photo Gallery

The main packages in this app include:

* Default: contains Activities, Fragments and their unit test classes.
* Services: Service classes which provide model, system and other tool function to Activities.
* Models: Model, Store access and Database server classes.
* Uitls: static utility classes.

# Classes

Figure 2 shows the classes of Photo Gallery app.



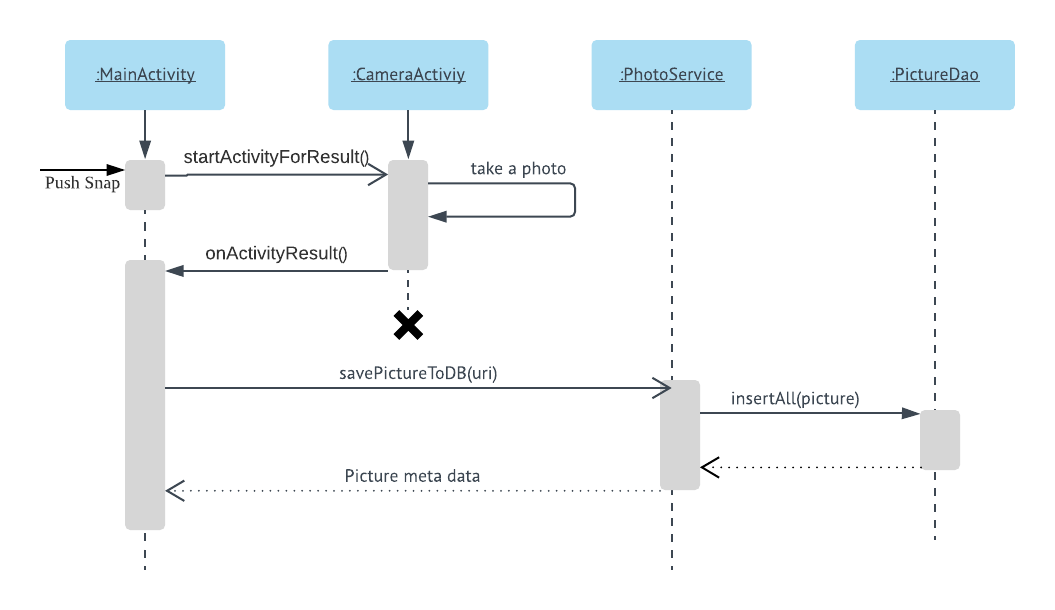
*Figure 2*: Class Diagram of Photo Gallery

The main class types include:

* Activity Class: the definition of pages and functions in Android app.
* Test Class: unit Test class for Activities
* Service Class: provide model, system and other tool functions to Activities
* Model Class: property definition of models
* Database and Dao Class: provide database service and access functions
* Utility Class: provide static tool functions

# Sequences

Figure 3 shows the sequence of ‘snap’.



*Figure 3*: Sequence Diagram of Snap

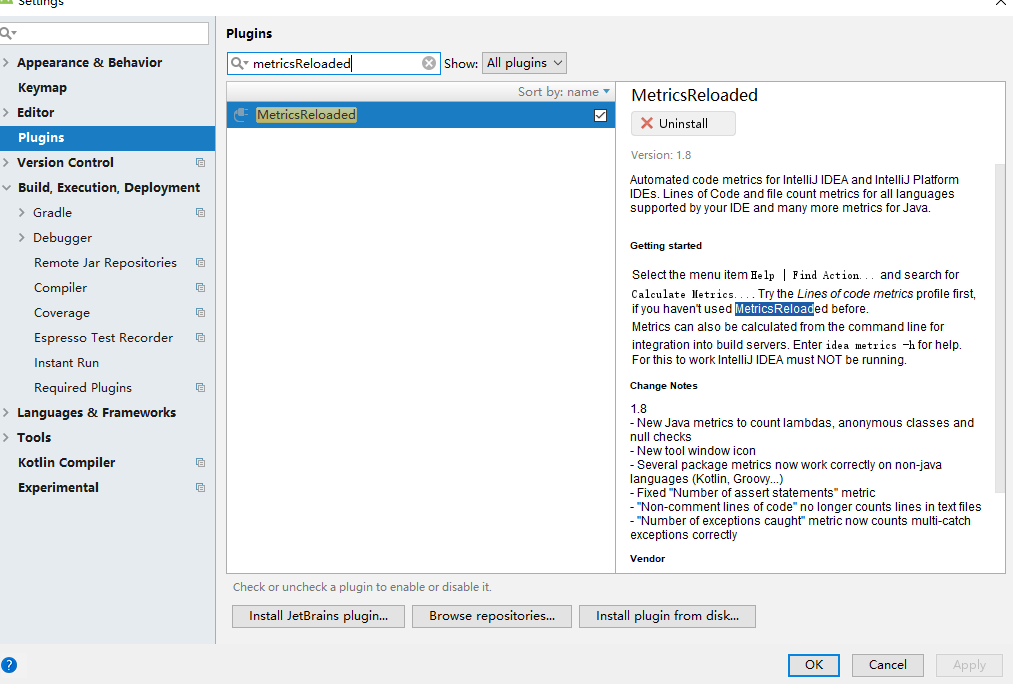
The sequence of taking a photo:

1. User pushes snap button.
2. MainActivity listen the push action, create an Intent and start system camera activity.
3. User takes a photo in system camera activity, camera callback MainActivity and terminate.
4. MainActivity gets the URI of the photo and call PhotoService to save the photo into database.
5. PhotoService creates a Picture model with default values and call insert function in PictureDao.
6. After PictureDao saves the model data, PhotoService return the model to MainAcitity to display the photo and information.

# Quantify Code

## Install Metrics Reloaded Plugin

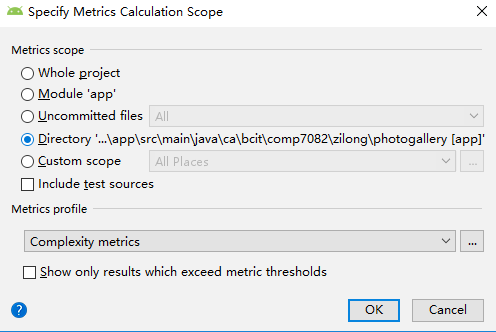
1. Go to Files -> Settings -> Plugins in Android Studio. You should see windows as Figure 4.
2. Input "MetricsReloaded" in the search bar and click link “search in repositories”. You should find the MetricsReloaded plugin.
3. Install the plugin and restart the Android Studio.



*Figure 4*: Plugin Manager in Android Studio.

## Analyze Code with Metrics Reloaded

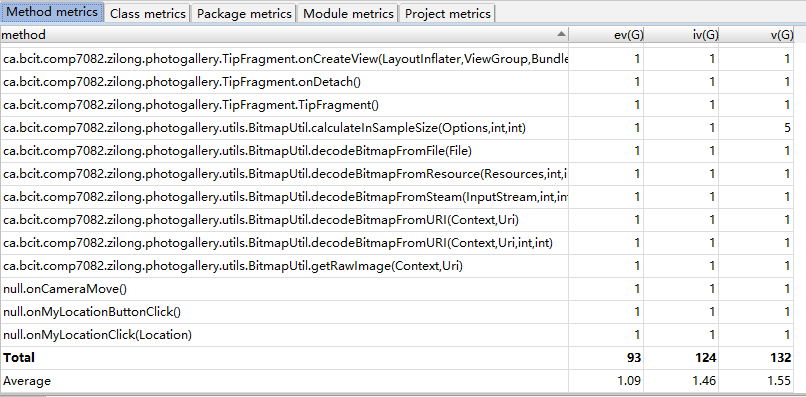
1. Select the package which you are going to analyze in the project window.
2. Go to Analyze -> Calculate Metrics, You should see windows as Figure 4.
3. Select metrics scope as "current file" and metrics profile as "Complexity Metrics" and Click on "Ok".



*Figure 5*: Analyze Code with Metrics Reloaded

## Analysis Results

Figure 6 shows the code complexity in method level.



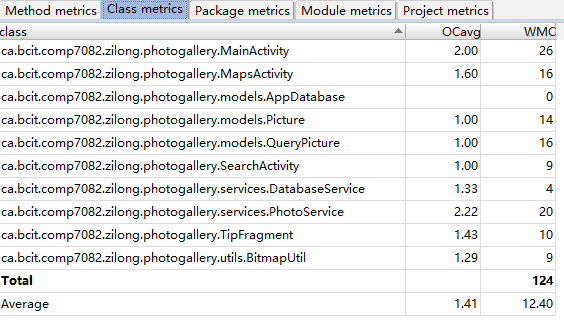
*Figure 6*: Code Complexity in Method Level

Columns meaning:

* ev(G): essential cyclometric complexity
* iv(G): module design complexity
* v : cyclometric complexity

The less values of these results mean the better code complexity. The minimums are 1.

Figure 7 shows the analysis results in class level

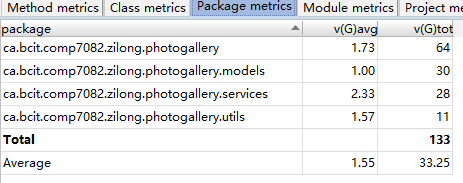


*Figure 7*: Analysis Results in Class Level

Columns meaning:

* OCavg: average Operation Complexity
* WMC: Weighted Methods Complexity

Figure 8 shows the analysis results in package level



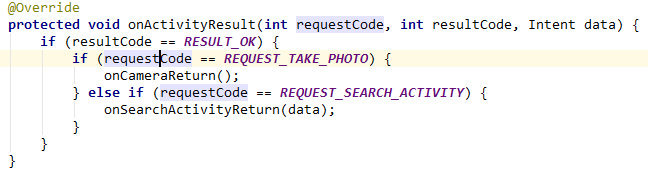
*Figure 8* Analysis Results in Package Level

# Improving Quality Cases

This section will show some example of how to change code complexity in method level.

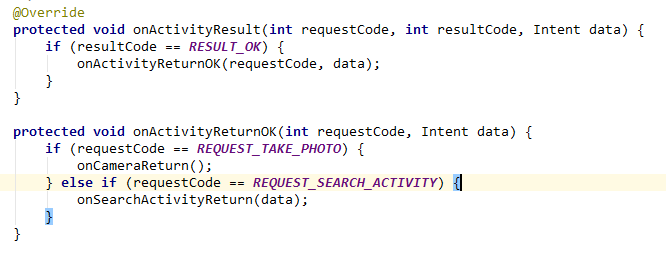
## Split Methods

Before modification:





After modification:





## Combine condition

Before modification:





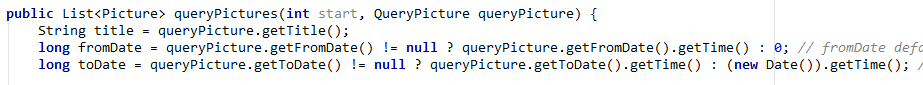
After modification:





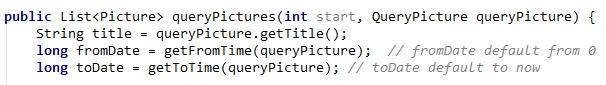
## Extract expression to method

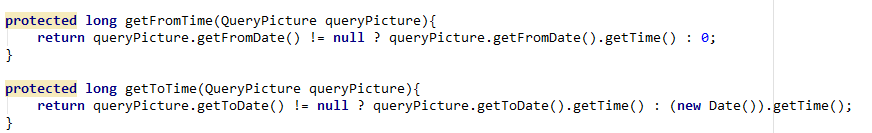
Before modification:

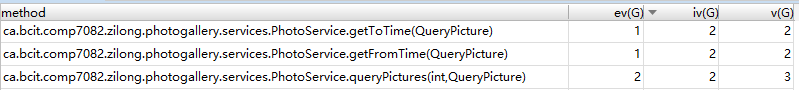




After modification:

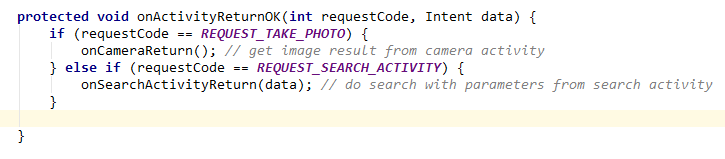


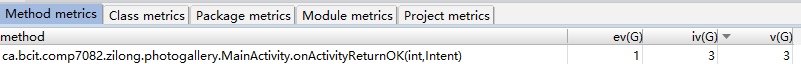




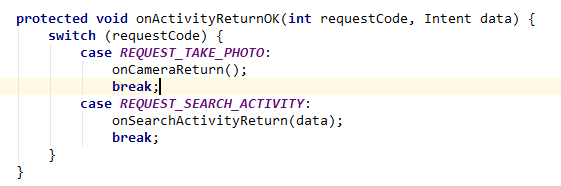
## Change *if* to *switch*

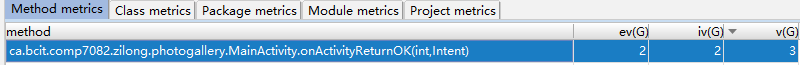
Before modification:





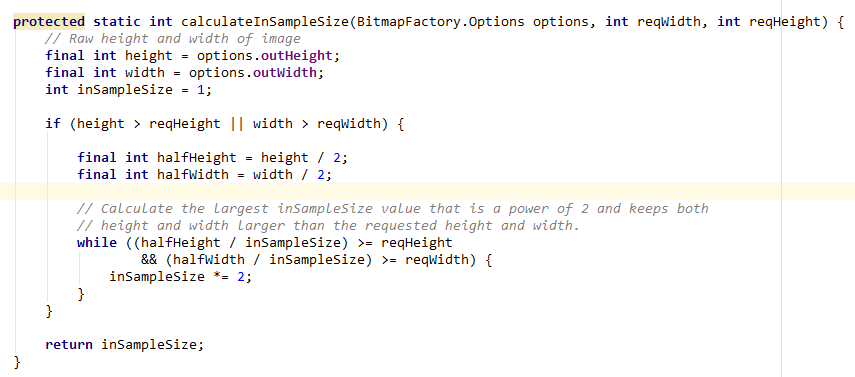
After modification:





## Simplify the expression of the algorithm

Before modification:



After modification:

