uml

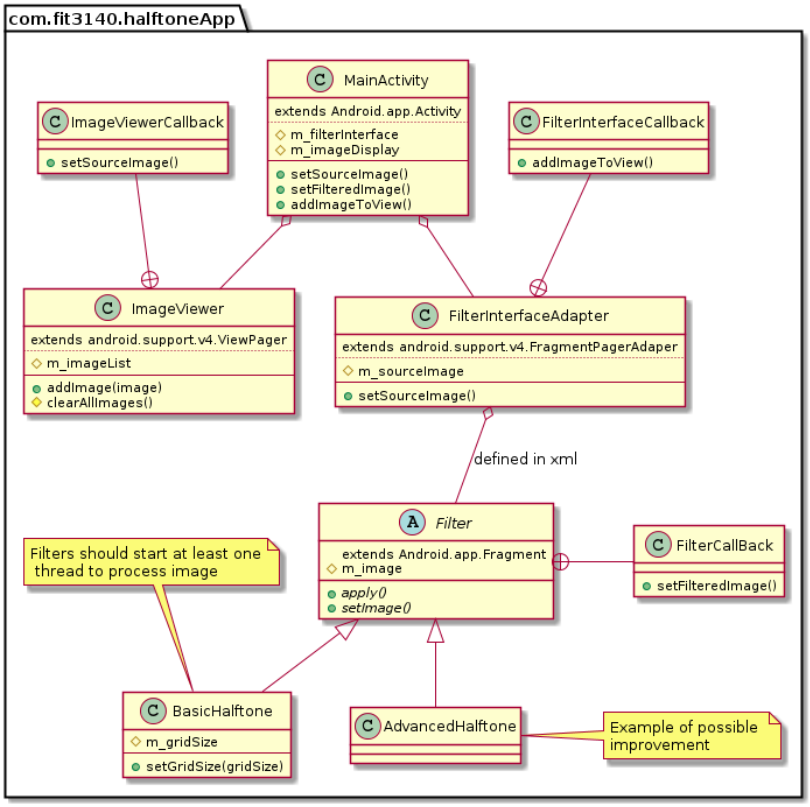
Jack Hosemans

April 30, 2014

Contents

1 UML diagram

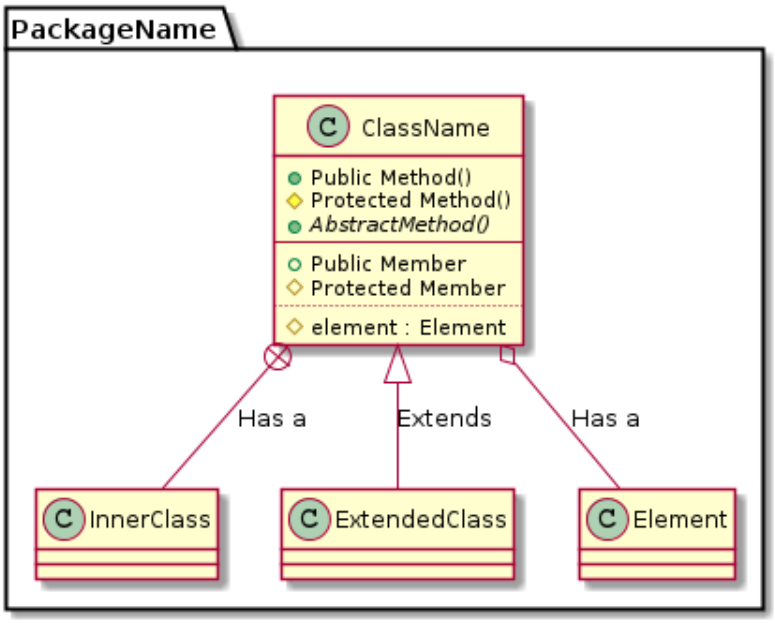
Note that android boilerplate code has been omitted for brevity, instead inherited classes from the android api have been used.



2 Dictionary

- Fragment: An graphical view independent from any other element.
- Adapter: A class designed to adapt the interface of one class with the interface of another.
- Thread: a process that runs within the application. Takes messages from the current program execution and acts upon it.
- Callback: A method that passed on to other code to be executed at some later time.
- Extends: A class inherits the methods and data values of another class.
- Implements: A class inherits the methods of an interface.

- Interface: A class that implements no functionality, but defines a signature for it's methods that other classes can use.



3 Descriptions of classes

3.1 MainActivity

This class should be the avenue that the filters get/add images to the user interface

3.1.1 setOriginalImage

This should only be called by the ImageViewer when the source image changes.

It calls the `setSourceImage` method in the `FilterInterfaceAdapter` class to change the image filtered when the user presses apply.

3.1.2 addImageToView

This should call the `addImageToView` method in the `ImageViewer` class in order to allow the user to view the results of the previous filter.

3.2 ImageViewer

This class is to show the image results to the user.

3.2.1 getCurrentImage

Sets the current `Bitmap` being viewed by the user by calling the `setSourceImage` method in the parent `MainActivity` via the callback class `FilterInterfaceCallback`.

3.2.2 setOriginalImage

Clears all current images then sets the new source image to the `Bitmap` passed in.

3.3 FilterInterfaceAdapter

This allows for swapping between the

Should implement a `ViewPager` so that the user can swipe between the halftoning modes that are implemented. Hopefully there will be more than one otherwise this will be wasted.

Each element in the `ViewPager` should be a fragment that implements its own interface to the user.

3.3.1 setImage

Sets the reference image to be passed on to the halftone implementation when the user presses "Apply"

3.4 Filter

This is an abstract class that forces any filter to implement an user interface via the `Fragment` api from android while adding it's own methods to do filtering on an image.

3.4.1 setImage

Sets the source image that the filtering should be applied to the image once the user presses apply.

3.4.2 apply

Applies the given filter to the set image.

3.5 BasicHalftone

This is an example of a basic filter, does basic halftoning to a given image with a given grid size.

Note that this current takes quite a while to apply.

3.5.1 Proposal for improvement

Halftoning usually takes a long time (>10s for large images.) A solution should be made for this.

- multiple threads could be used to work around the slowness of the application.
  - This would give the advantage of not slowing down the GUI
  - However, it would be harder to code. Maybe another spike is in order for implementation?
- A hard limit on the grid size would also be sufficient
  - "worse" than the threaded way because the application would appear to freeze when being halftoned. This is VERY bad.
  - Worse output due to each dot being bigger than any other way.
  - Not using the full power of the processor (We need all we can get)

3.6 AdvancedHalftone

This is currently just a placeholder in case we extend the application to do more than basic halftoning.

Implementations:

- Different colors
- Different angles
- Adding text
- Background colors
- Sepia
- dithering