Homework 1 Write-Up

We have selected two features of the Glide library: Image Display and Image Processing. Both are high-level features that can be implemented at scale, depending on the different types of images that get displayed or processed. The Glide project provides a sample svg module, which demos the two features on images with .svg file type. To understand the specifics of how the two features work, we based our write-up on the examination of these implementations mostly within the svg module sample.

Feature 1: Image Display (e.g. svg)

Because Glide provides an image loader library on Android, one of the main features is to display images into ImageView. To examine how the feature is implemented, we took a look at the svg sample module to look into a specific case of image display from just image resources with the svg file type.

Assuming that a method related to image load or image display is invoked at the onStart() method, we started at the MainActivity class in the /svg/src folder. In the onStart() method, the reload() method is called. Tracing the method, we check what is implemented in reload().

```
79    private void reload() {
80      Log.w(TAG,  msg: "reloading");
81      ((TextView) findViewById(R.id.button))
82          .setText(getString(R.string.scaleType, imageViewRes.getScaleType()));
83      loadRes();
84      loadNet();
85    }
```

We find that the reload() method logs a message, defines the TextView in the layout, and calls two methods, loadRes() and loadNet(). From running the sample on AVD, we discovered that loadRes() loads a resource from an image file directory, while loadNet() loads a resource from a web URL.

We trace the loadRes() and loadNet() methods to find out how images resources actually get displayed.

```
87    private void loadRes() {
88      Uri uri =
89          Uri.parse(
90              ContentResolver.SCHEME_ANDROID_RESOURCE
91                  + "://"
92                  + getPackageName()
93                  + "/"
94                  + R.raw.android_toy_h);
95      requestBuilder.load(uri).into(imageViewRes);
96    }
97
98    private void loadNet() {
99      Uri uri = Uri.parse("http://www.clker.com/cliparts/u/Z/2/b/a/6/android-toy-h.svg");
100     requestBuilder.load(uri).into(imageViewNet);
101   }
```

At first glance, we discovered that the two methods have similar implementations. Both store the resource uri in a variable, and pass the variable to load into imageViewRes and imageViewNet, respectively, through requestBuilder.

We remembered seeing requestBuilder being initially defined at the start of the MainActivity class, in the onCreate() method, so we trace back to onCreate() to check what a requestBuilder does.

In the onCreate() method, we first see that the two ImageViews are defined; one for the image resource from directory, another from a web URL. Then, the method defines requestBuilder, which calls GlideApp.with(this), where this is an activity, to begin a load with Glide that is tied to a given activity's lifecycle. Glide.with() returns RequestManager, which is extended by GlideRequests, which is returned by GlideApp.with().

```
27    @Override
28    protected void onCreate(Bundle savedInstanceState) {
29      super.onCreate(savedInstanceState);
30      setContentView(R.layout.activity_main);
31
32      imageViewRes = (ImageView) findViewById(R.id.svg_image_view1);
33      imageViewNet = (ImageView) findViewById(R.id.svg_image_view2);
34
35      requestBuilder =
36          GlideApp.with( activity: this) GlideRequests
37              .as(PictureDrawable.class) GlideRequest<PictureDrawable>
38              .placeholder(R.drawable.image_loading) GlideRequest<PictureDrawable>
39              .error(R.drawable.image_error) GlideRequest<PictureDrawable>
40              .transition(withCrossFade()) GlideRequest<PictureDrawable>
41              .listener(new SvgSoftwareLayerSetter());
42    }
```

A series of methods are called upon loading the requestBuilder for an image; as(), placeholder(), error(), transition(), and listener().

For a given activity, as() specifies GlideRequest type to be <PictureDrawable>, placeholder() then sets an android resource id for a resource to display while a resource is loading, error() then sets a resource to display when load fails, transition() then sets transition options to use to transition from placeholder when load completes, and lastly the listener() sets a request listener to monitor the resource load.

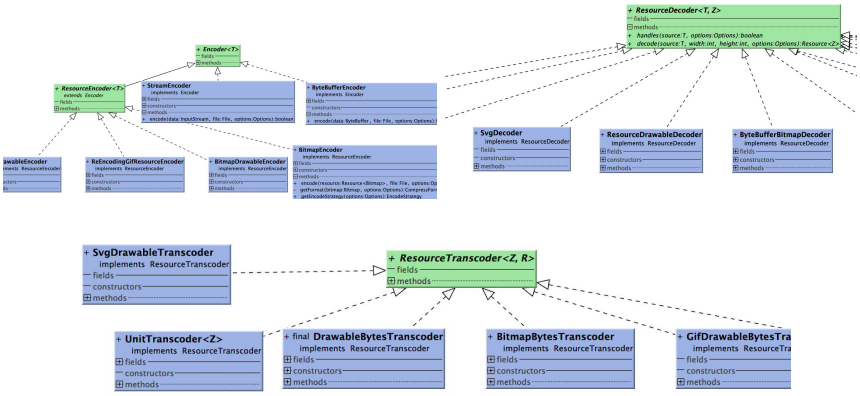
SvgSoftwareLayerSetter class implemented another library interface called RequestListener(). Two methods are overridden: onLoadFailed() and onResourceReady(). This method determines whether an svg source can be rendered onto the target display (screen).

```java
/**
 * Listener which updates the {@link ImageView} to be software rendered, because {@link
 * com.caverock.androidsvg.SVG SVG}/{@link android.graphics.Picture Picture} can't render on a
 * hardware backed {@link android.graphics.Canvas Canvas}.
 */
public class SvgSoftwareLayerSetter implements RequestListener<PictureDrawable> {

  @Override
  public boolean onLoadFailed(
      GlideException e, Object model, Target<PictureDrawable> target, boolean isFirstResource) {
    ImageView view = ((ImageViewTarget<?>) target).getView();
    view.setLayerType(ImageView.LAYER_TYPE_NONE,  paint: null);
    return false;
  }

  @Override
  public boolean onResourceReady(
```

Going back to loadRes() and loadNet(), we can now understand that these methods are called as part of the GlideRequest to implement effects and transitions as an image resource is loading and displaying onto the screen.

Feature 2: Resource File Processing (e.g. svg)

One of the features that Glide possesses is the ability to process different types of image file types. We decided to focus in on the processing of .svg files, which is displayed in a sample program within Glide's source code. To start our search we looked at the class SvgDecoder, within the svg folder. We found that the SvgDecoder implemented an interface called ResourceDecoder. This interface was located in the main library of Glide under their resource folder. We can see from the UML diagram this relationship as well:



SvgDecoder overrides ResourceDecoder's two methods, decode() and handles(). Both of these methods appear to determine whether an input source is valid or not. From this we believe that SvgDecoder specifically considers .svg file input and whether or not it is decodable. This becomes useful when we need to potentially reformat this image file type to something more compatible. This took us to looking into the next related class SvgDrawableTranscoder. Just like the decoder, there is a main library interface, which this class implements called ResourceTranscoder. The method overridden is transcode() and appears to aid in the conversion of .svg files to something that is compatible with the Glide app:

```java
/**
 * Convert the {@link SVG}'s internal representation to an Android-compatible one ({@link Picture}).
 */
public class SvgDrawableTranscoder implements ResourceTranscoder<SVG, PictureDrawable> {
  @Nullable
  @Override
  public Resource<PictureDrawable> transcode(
```

Under the sample program for svg, there are two methods in the class SvgModule: registerComponents() and isManifestParsingEnabled(). The class extends AppGlideModule, which is on the larger scope of the project. However, we looked into the basic implementations of this class and it seems to establish dependencies and specific options to use "when initializing Glide within an application" SvgModule overrides it's two methods, and must be integral in the initial setup of the sample program.

```java
/**
 * Defines a set of dependencies and options to use when initializing Glide within an application.
 *
 * <p>There can be at most one {@link AppGlideModule} in an application. Only Applications can
 * include a {@link AppGlideModule}. Libraries must use {@link LibraryGlideModule}.
 *
 * <p>Classes that extend {@link AppGlideModule} must be annotated with {@link
 * com.bumptech.glide.annotation.GlideModule} to be processed correctly.
 *
 * <p>Classes that extend {@link AppGlideModule} can optionally be annotated with {@link
 * com.bumptech.glide.annotation.Excludes} to optionally exclude one or more {@link
 * LibraryGlideModule} and/or {@link GlideModule} classes.
 *
 * <p>Once an application has migrated itself and all libraries it depends on to use Glide's
 * annotation processor, {@link AppGlideModule} implementations should override {@link
 * #isManifestParsingEnabled()} and return {@code false}.
 */
// Used only in javadoc.
@deprecation@
public abstract class AppGlideModule extends LibraryGlideModule implements AppliesOptions {
```

Overall, from our analysis of svg processing, we have determined that this is just one example of file processing that Glide offers. Through implementing the specified library interfaces, we believe that Glide makes it possible to process other types of image files as well.

Templates

Feature 1: Image Display (svg)

Folder	File	Method	Relevant?	Relevant how?	Confidence	Notes
/svg/src	MainActivity	reload()	yes	Is called in onStart() method	high	Calls loadRes() and loadNet()
/svg/src	MainActivity	loadRes() loadNet()	yes	Is called in reload() method	high	Generates request using requestBuilder; loadRes() loads resource into imageViewRes, loadNet() loads web resource into imageViewNet
/svg/src	MainActivity	requestBuilder.load(uri)	yes		high	Here is how to load an SVG image into Image View, and after we are going to take a look at how requestBuilder is created and defined in onCreate() method.
/svg/src	MainActivity	onCreate()	yes	Defines field ImageView for different resource types and generates template for requestBuilder	High	Determines where the images fetched from the resource get displayed on the screen
/build/svg/src	GlideApp	with(this)	yes	Is called when create requestBuilder in MainActivity	high	Called Glide.with() and returns GlideRequests which extends RequestManager
/library/src/.../glide	Glide	with(activity)	Yes	Is called in GlideApp with() method	high	Begin a load with Glide that will be tied to a given activity's lifecycle then returns RequestManager which is extended by GlideRequests
/build/svg/src	GlideRequests	as(resourceClass) placeholder() error() transition() listener()	Yes	Is called when create requestBuilder in MainActivity	high	

/library/src/.../glide	RequestBuilder	RequestBuilder()	yes	Is called when GlideRequest as() method is called	high	Specify GlideRequest type to be <PictureDrawable>
/library/src/.../glide/request	BaseRequestOptions	placeholder()	yes	Is called when GlideRequest placeholder() method is called	high	Set an android resource id for a resource to display while a resource is loading.
/library/src/.../glide/request	BaseRequestOptions	error()	yes	Is called when GlideRequest error() method is called	high	Set a resource to display when load fails.
/library/src/.../glide	RequestBuilder	transition	yes	Is called when GlideRequest transition() method is called	high	Set transition options to use to transition from placeholder when load completes.
/library/src/.../glide	RequestBuilder	listener	yes	Is called when GlideRequest listener() method is called	high	Set a request listener to monitor the resource load
/svg	SvgSoftwareLayerSetter		yes	This class is instantiated when RequestBuilder listener() is called	high	Listener listens to onLoadFailed() and onResourceReady().

Feature 2: File Processing (svg)

Folder	File	Method	Relevant?	Relevant how?	Confidence	Notes
/svg	SvgDecoder	handles()	yes	Returns boolean for decodable or not decodable	high	Always returns true, so need to find where SvgDecoder is used

/svg	SvgDecoder	decode()	yes	If handles returns true, returns the source of the svg file	high	
/src/load/resource/transcode	ResourceDecoder	decode()	yes	Image file processing	high	Help decode different image file types
/svg	SvgDrawableTranscoder	transcode()	yes	Convert svg file to an android-compatible version <PictureDrawable>	high	File Conversion method. Implements SuperClass ResourceTranscoder
/src/load/resource/transcode	ResourceTranscoder	transcode()	yes	It is implemented by SvgDrawableTranscode	high	
/svg	SvgModule	registerComponents()	yes	Registers components of svg file to app Glide	high	Extends AppGlideModule and overrides the methods to establish dependencies and requirements of functions for the particular module
/svg	SvgModule	isManifestParsingEnabled()	yes	Complementary method to ensure no duplicates of the svg module sample	high	
/library/request	RequestListener	onLoadFailed() and onResourceReady()	yes	SvgSoftwareLayerSetter implements this interface	high	Listens to updates to the image link to be software rendered
/svg	SvgSoftwareLayerSetter	onLoadFailed() and onResourceReady()	yes	Implements RequestListener	high	Determines whether an svg source can or cannot be rendered onto target. Monitors status of these requests while images load
/src/load/resource/	Encoder	encode()	yes	Encoder interface is an important part in Files Processing procedure	high	An interface for writing data to some persistent data store(i.e. A local File cache)