Android Assignment

1. Create a worker class which will run the tasks asynchronously. You can queue multiple tasks in this worker class but only one task should be run at a time.

```
//Task interface
public interface Task<T> {
   T onExecuteTask();
   void onTaskComplete();
}

ServiceWorker serviceWorker = new ServiceWorker("service_worker");
serviceWorker.addTask(new Task<String>() {
   @Override
   public String onExecuteTask() {
    //This callback should run on background thread
    //Do the background work here
    //TASK1
    //returns String based  on what is defined in the implementation
    Return result;
   }

   @Override
   public void onTaskComplete(String result) {
    //This should run on main thread
    //The result should be same as object(in this case a String) which is returned in
    //onExecuteTask method
     textView.setText(result);
   }
});

serviceWorker.addTask(new Task<String>() {
   @Override
   public String onExecuteTask() {
     //TASK2
   }

   @Override
   public void onTaskComplete(String result) {
      textView.setText(result);
   }
});
```

2. In the implementation above **ServiceWorker** class has a method called **addTask** through which we enqueue tasks, added two tasks into the service worker, *TASK1* runs first then *TASK2*

runs and so on. Please make note that **onExecuteTask** should run on background thread and **onTaskComplete** should run on Main thread for updating any UI as shown in the code.

3. If you create two worker objects, both should have separate worker queue and separate threads where tasks run on.

```
ServiceWorker serviceWorker1 = new ServiceWorker("service_worker_1");
ServiceWorker serviceWorker2 = new ServiceWorker("service_worker_2");
Task taskForWorker1 = new Task<String>() {
   //method implementations
}
Task taskForWorker2 = new Task<String>() {
   //method implementations
}
serviceWorker1.addTask(taskForWorker1);
serviceWorker2.addTask(taskForWorker2);
```

Here in the above code we created two service workers and two tasks, tasks run different threads, **taskForWorker1** runs on **serviceWorker1** and **taskForWorker2** runs on **serviceWorker2**.

4. After creating ServiceWorker class, add two Buttons and two ImageViews in an activtiy as shown in the code below. Create two service worker objects, worker1 and worker2, when **button1** is pressed call **fetchImage1AndSet** method where we use **serviceWorker1** to fetch the image, when **button2** is pressed call **fetchImage2AndSet** where **serviceWorker2** is used.

```
ServiceWorker serviceWorker1 = new ServiceWorker("service_worker_1");
ServiceWorker serviceWorker2 = new ServiceWorker("service_worker_2");
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    button1.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
        fetchImage1AndSet()
      }
    });
    button2.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
        fetchImage2AndSet()
      }
    });
}
```

```
Public static final String IMAGE_1= “”;
Public static final String IMAGE_2= “”;

private void fetchImage1AndSet() {
   serviceWorker1.addTask(new Task<Bitmap>() {
       @Override
       public Bitmap onExecuteTask() {
            //Fetching image1 through okhttp
            Request request = new Request.Builder().url(IMAGE_1).build();
            Response response = okHttpClient.newCall(request).execute();
            final Bitmap bitmap = BitmapFactory.decodeStream(response.body().byteStream());
            return bitmap;
       }

       @Override
       public void onTaskComplete(Bitmap result) {
          //Set bitmap to imageview
           imageView1.setBitmap(result);
       }
   });

}

private void fetchImage2AndSet() {
   serviceWorker2.addTask(new Task<Bitmap>() {
       @Override
       public Bitmap onExecuteTask() {
           //Fetching image1 through okhttp
           Request request = new Request.Builder().url(IMAGE_2).build();
           Response response = okHttpClient.newCall(request).execute();
           final Bitmap bitmap = BitmapFactory.decodeStream(response.body().byteStream());
           return bitmap;
           return result;
       }

       @Override
       public void onTaskComplete(Bitmap result) {
           //Set bitmap to image 1
           imageView2.setBitmap(result);
       }
   });

}
```

5. You can use ThreadPools, Threads, HandlerThreads etc..

6. Please upload the code to github and share the repo link.