Lab 10.1 – Asynchronous Tasks

Overview:

In this lab you will turn a long running process on the UI thread into a separate asynchronous tasks that will still be able to communicate with the UI thread.

Step	Description
1.	Add a new menu item to the main menu called “Refresh” which will get the events from the EventService and display them in the main activity.

a.	Add the new menu item to “res/menu/main_menu.xml” .
	
	`<item`
	`android:id="@+id/mi_refresh"`
	`android:title="@string/mi_refresh"/>`
	
	Also add the new string to “res/values/strings.xml”
	
	`<string name="mi_refresh">Refresh</string>`

b.	In “MainActivity” handle the new menu selection and create a new method called “showRefresh” which will be called when the user selects the “Refresh” menu item.
	
	Add the following code to the “onOptionsItemSelected” method
	
	`case R.id.mi_refresh:`
	`showRefresh();`
	`return true;`
	
	In the “showRefresh” method, call “displayListView”
	
	`private void showRefresh() {`
	`Log.v("ShowEventActivity", "Running showRefresh method.");`
	`displayListView();`
	`}`
	
	Note: the “displayListView” method calls “EventService.getAllEvents()” which returns the list of events.
	
	Run the application to verify that the events are being re-displayed when the “Refresh” menu option is selected.

2.	Increase the execution time of the EventService method “addEvent” until it forces the UI activity to be stopped by the Android OS

a.	Add the following code to the top of the “getAllEvents” method in the “EventService” class:
	
	`try {`
	`Log.v("EventService", "Thread sleep");`
	`Thread.sleep(10000);`
	`Log.v("EventService", "Thread awake");`
	`} catch (InterruptedException e) {`
	`e.printStackTrace();`

	} This will cause the UI thread to sleep for 10 seconds. It is simulating the effect of making a very long call to the server. The UI will either now crash or hang. This is a problem since the UI should be very responsive. Notice that the UI freezes after the user presses the "Refresh" button on the menu.
3.	Create the asynchronous task which will be run from the main UI thread.

a.	Create a new class in the "src/com.garagze.service" package called "GetAllEventsTask" and make "AsyncTask" the super class.
	
	`public class GetAllEventsTask`
	`  extends AsyncTask<Context, Void, ArrayList<Event>>`
	
	The first parameter is passed to the "execute" method
	
	The second parameter is passed to the "onProgress" method
	
	The third parameter is returned from the "doInBackground" method and then passed to "onPostExecution"
	
	Create a constructor to pass the context (which should be the MainActivity itself)
	
	`private MainActivity mainActivity;`
	
	`public GetAllEventsTask(MainActivity mainActivity) {`
	`  this.mainActivity = mainActivity;`
	`}`

b.	Declare the method "doInBackground" which runs on a separate thread. The long running process will be executed in this method, outside of the UI thread.
	
	`@Override`
	`protected ArrayList<Event> doInBackground(Context... context) {`
	`  return EventService.getAllEvents(context);`
	`}`

c.	Declare the method which runs after the background method completes. This method runs on the UI thread of MainActivity.
	
	`@Override`
	`protected void onPostExecute(ArrayList<Event> events) {`
	`  mainActivity.buildAdapter(events);`
	`}`
	
	You will get an error in the above code because the method "buildAdapter" does not exist in "MainActivity". You will create it in the next step.

d.	Create a new method "buildAdapter" in "MainActivity.java". The method should take an "ArrayList<Event" object as a parameter. The purpose of this method is to remove the building of the adapter from "displayListView" so that it can be called from the AsyncTask without causing the list to re-display with new data. If you take this approach, make your code look like the following:
	
	`private void displayListView() {`
	`  final List<Event> events = EventService.getAllEvents();`
	`  final ArrayAdapter<Event> arrayAdapter = buildAdapter(events);`
	`  ListView listView = (ListView) findViewById(R.id.eventlistview);`
	`  listView.setAdapter(arrayAdapter);`

	}
	
	public ArrayAdapter<Event> buildAdapter(final List<Event> events) {
	final ArrayAdapter<Event> arrayAdapter =
	new EventArrayAdapter(this, R.layout.event_list_item, events);
	return arrayAdapter;
	}
	
	Another technique would be to change “displayListView” to take an event arrray list and allow the list to re-display as soon as the new data is acquired. This would require simply making the “displayListView” method public.

4	Change the call to “getAllEvents” into an asynchronous task that is executed in a separate thread from the activity.
a.	Replace the call to “getAllEvents” in “displayListView” with a call to the new asynchronous method as follows: GetAllEventsTask task = new GetAllEventsTask(this); task.execute(this);
5.	The user should be notified that data is being retrieved.

a.	In the “preExecute” method in "GetAllEventsTask.java", start a progress bar.
	
	@Override
	protected void onPreExecute() {
	mainActivity.clearList();
	pd = new ProgressDialog(mainActivity);
	pd.setMessage("Loading Events ...");
	pd.show();
	}
	
	Note: Create a new method in “MainActivity” called “clearList” which will empty the list view.
	
	Make "pd" an instance variable of "GetAllEventsTask"

b.	In the “postExecute” method in "GetAllEventsTask.java", stop the progres bar.
	
	@Override
	protected void onPostExecute(ArrayList<Event> events) {
	mainActivity.setEvents(events);
	mainActivity.showList();
	pd.hide();
	}
	
	Note: Create a new method in “MainActivity” called “setEvents” which will populate an event instance variable. Also create a new method called “showList” which will create a new adapter from whatever the current events are and add that adapter to the list view.

c.	In the “doInBackground” method, get the data.
	
	@Override
	protected ArrayList<Event> doInBackground(Context... params) {
	return (ArrayList<Event>) EventService.getAllEvents();
	}

d.	Spoiler Alert! Following is the code you can use to implement the new methods in “MainActivity”. You may

want to try this yourself before looking at the code below.

```
ArrayList<Event> events = new ArrayList<Event>();
ArrayAdapter<Event> arrayAdapter = null;
ListView listView;

public void clearList() {
      if (listView != null) {
            listView.setAdapter(null);
      }
}

public void showList() {
      // Build the adapter with whatever the current events are
      arrayAdapter = buildAdapter();
      listView = (ListView) findViewById(R.id.eventlistview);
      listView.setAdapter(arrayAdapter);
}

public void setEvents(ArrayList<Event> events) {
      this.events = events;
}

private ArrayAdapter<Event> buildAdapter() {
      final ArrayAdapter<Event> arrayAdapter =
         new EventArrayAdapter(this, R.layout.event_list_item, events);
      return arrayAdapter;
}

private void refreshList() {
      GetAllEventsTask task = new GetAllEventsTask(this);
      task.execute(this);
}
```

Note: Remove the “displayListView” method and replace calls to it with “refreshList().