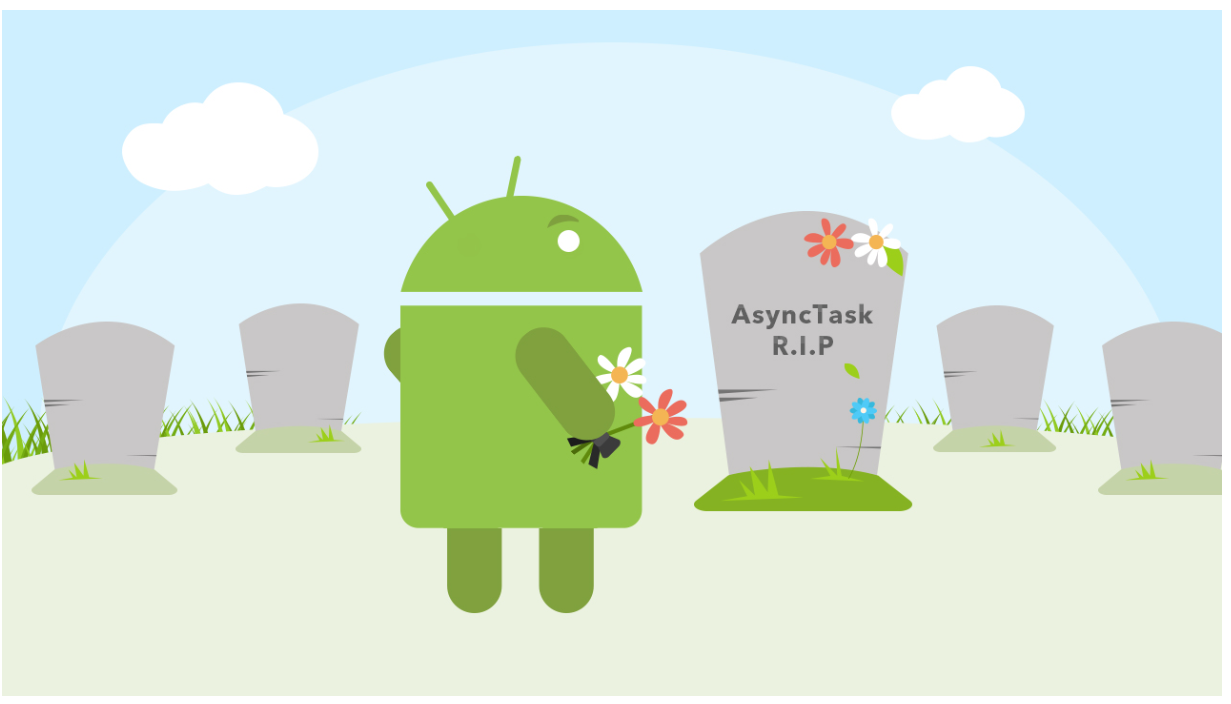
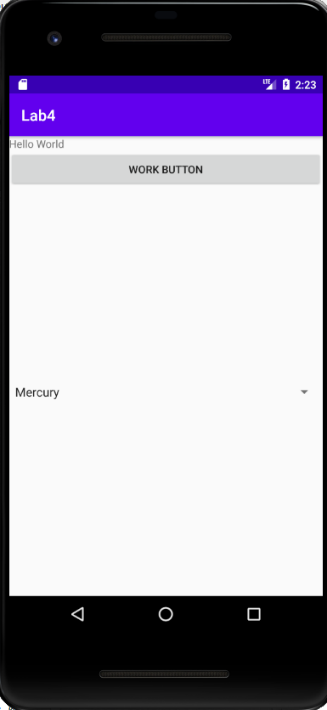
**CSE2MAD Lab 4 Part B WorkManager and Running tasks separate to the UI Thread**



Aim:

1. To build an application that uses WorkManager to execute a long running task and have the main UI thread serviceable (you will continue to be able to use the spinner)
2. To send data to the Worker from MainActivity
3. To receive data from the Work on completion of its task

Step 1: Create a new Android Studio Project and choose ‘Empty Activity’ as the project type.



Step 2: Create a UI as per the layout enclosed.

*<?***xml version="1.0" encoding="utf-8"***?>*<**androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".MainActivity"**>  
  
 <**LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"**>  
  
 <**TextView  
 android:id="@+id/textView"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Hello World"** />  
  
 <**LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal"**>  
  
 <**Button  
 android:id="@+id/WorkButton"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:text="Work Button"** />  
  
 </**LinearLayout**>  
  
 <**Spinner  
 android:id="@+id/spinner"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"** />  
 </**LinearLayout**>  
</**androidx.constraintlayout.widget.ConstraintLayout**>

Step 3: Lets store the values for the spinner. Got to the ‘res/values’ folder and open strings.xml, enter the following.

<**string-array name="planets\_array"**>  
 <**item** >Mercury</**item** >  
 <**item** >Venus</**item**>  
 <**item** >Earth</**item**>  
 <**item** >Mars</**item** >  
 <**item** >Jupiter</**item** >  
 <**item** >Saturn</**item** >  
 <**item** >Uranus</**item** >  
 <**item** >Neptune</**item** >  
</**string-array** >

In general, if you have static data, it is best to store it in the project resource folder.

Step 4: Fill out the class members of MainActivity.

**private** Button **startWorkBtn** =**null**;  
**private** TextView **txtView** = **null**;  
**private** Spinner **spinner** = **null**;  
  
**private** String[] **urls**;

Step 5: Refer back to the lecture notes in week 4 on WorkManager, we are going to implement it next.

1. Let’s import the code

**import** androidx.annotation.Nullable;  
**import** androidx.appcompat.app.AppCompatActivity;  
**import** androidx.lifecycle.Observer;  
**import** androidx.work.Data;  
**import** androidx.work.OneTimeWorkRequest;  
**import** androidx.work.WorkInfo;  
**import** androidx.work.WorkManager;

1. Implement the onCreate()

**super**.onCreate(savedInstanceState);  
setContentView(R.layout.***activity\_main***);

*// initialise the UI widgets***startWorkBtn** = (Button) findViewById(R.id.***WorkButton***);  
**txtView** = (TextView) findViewById(R.id.***textView***);  
**spinner** = (Spinner) findViewById(R.id.***spinner***);

*// Create an ArrayAdapter using the string array and a default spinner layout*ArrayAdapter<CharSequence> adapter = ArrayAdapter.*createFromResource*(**this** ,  
 R.array.***planets\_array***, android.R.layout.***simple\_spinner\_item***);  
  
*// Specify the layout to use when the list of choices appears*adapter.setDropDownViewResource(android.R.layout.***simple\_spinner\_dropdown\_item***);  
  
*// Apply the adapter to the spinner***spinner**.setAdapter(adapter);

*// initialise the class member with some data*initURLS();

At this stage implement initURLS() as a member of MainActivity

*// this will be the data to pass to the worker  
// init a class member of MainActivity***private void** initURLS(){  
 **try** {  
 **urls** = **new** String[]{  
 **new** String(**"http://www.amazon.com/somefiles.pdf"** ),  
 **new** String(**"http://www.wrox.com/somefiles.pdf"** ),  
 **new** String(**"http://www.google.com/somefiles.pdf"** ),  
 **new** String(**"http://www.learn2develop.net/somefiles.pdf"** ),  
 **new** String(**"http://www.wrox.com/somefiles.pdf"** )};  
 }**catch** (Exception e){  
 e.printStackTrace();  
 }  
}

Go back to onCreate(), all the below code to be added to this method.

*// data to pass to worker*Data data = **new** Data.Builder().putStringArray(BackgroundWorker.***URLS***, **urls**)  
 .build();

*// build the work request (BackgroundWorker.class will do the 'work' for us***final** OneTimeWorkRequest workRequest =  
 **new** OneTimeWorkRequest.Builder(BackgroundWorker.**class**)  
 .setInputData(data)  
 .build();

Here we begin to use WorkManager. First we create a data object (it will contain the String[] urls), note we use a key-value pair (BackgroundWorker.***URLS***, **urls**). The keen-eyed amongst you will have seen that we will be using a user defined class ‘BackgroundWorker’ with a member ***URLS*** as the key. Since we are only interested in a one-off request we use OneTimeWorkRequest in conjunction with BackgroundWorker.class, and we set input data to be our array of URL’s.

Before we implement the class BackgroundWorker, let’s register a listener for the button.

**startWorkBtn**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
 **txtView**.setText(**"Starting Work ....."** + **"\n"**);  
 **startWorkBtn**.setEnabled(**false**);  
 WorkManager.*getInstance*(getApplication()).enqueue(workRequest);  
 }  
});

Note: here in the button to ‘do some work’, we are adding a work request to the queue.

Lastly, we need some code to handle changes that are occurring in the worker class.

*// listen for changes of the worker*WorkManager.*getInstance*(**this**.getApplication()).getWorkInfoByIdLiveData(workRequest.getId())  
 .observe(**this**, **new** Observer<WorkInfo>() {  
  
 @Override  
 **public void** onChanged(@Nullable WorkInfo workInfo) {  
  
 *//Displaying the status into TextView* **txtView**.append(workInfo.getState().name() + **"\n"**);  
  
 *// check to see if the worker has completed* **if**(workInfo != **null** && workInfo.getState().isFinished()) {  
 **txtView**.append(workInfo.getOutputData().getString(BackgroundWorker.***RETURNTEXT***) + **"\n"**);  
 **startWorkBtn**.setEnabled(**true**);  
 }  
 }  
 });

Here we are observing changes using the Observer pattern. There are a few UI changes we wish to make depending on what the worker is doing. First we wish to print the state of the worker. Note it cycles through Enqueued, Running and Succeeded states if all goes well. We simple print this to the textView.

The final conditional block checks to see if the worker is finished, if so, we add the value contained in the worker to the txtView. We access this with a specific key, BackgroundWorker.***RETURNTEXT.*** As would expect, this would appear as a member of the worker class.

Step 6: Implement the BackgroundWorker class

Create a new Java Class in your project and paste the code below.

**package** com.example.lab4;  
  
**import** android.content.Context;  
**import** android.util.Log;  
  
**import** androidx.annotation.NonNull;  
**import** androidx.work.Data;  
**import** androidx.work.Worker;  
**import** androidx.work.WorkerParameters;  
  
**public class** BackgroundWorker **extends** Worker {  
  
 **public static final** String ***URLS*** = **"URLS"**; *// from the MainActivity* **public static final** String ***RETURNTEXT*** = **"receivedWorkerText"**; *// to the MainActivity* **public** BackgroundWorker(  
 @NonNull Context context,  
 @NonNull WorkerParameters params) {  
 **super**(context, params);  
 }  
  
 @Override  
 **public** Result doWork() {  
  
 Context appContext = getApplicationContext();  
  
 **try** {  
 *// do something that may take some time* downloadFile(getInputData().getStringArray(***URLS***));  
  
 *// Indicate the work finished successfully with the Result* Data data = **new** Data.Builder()  
 .putString(***RETURNTEXT***, **"Work Done!!"**)  
 .build();  
  
 **return** Result.*success*(data);  
 } **catch** (Throwable throwable) {  
 Log.*e*(**"ERROR"**, **"doWork() Failed"**, throwable);  
 *// Indicate the work failed with the Result* Data data = **new** Data.Builder()  
 .putString(***RETURNTEXT***, **"Work Failed!!"**)  
 .build();  
 **return** Result.*failure*();  
 }  
 }  
  
  
 **private void** downloadFile(String[] urls) {  
 **try** {  
  
 **for** (String url: urls) {  
 *//---simulate taking some time to download a file---* Thread.*sleep*( 2000);  
 }  
  
 } **catch** (InterruptedException e) {  
 e.printStackTrace();  
 }  
  
 }  
}

The first thing to note is the class extends Worker, it will be used to run a task.

The Class has static ID’s that are used to reference to/from links.

After the constructor, we implement doWork() where will be create long running execute, downloadFile simulates a lengthy task.

At its completion, we build a new data object, insert a string for display and return. This value is printed from MainActivity.

In summary, you have used WorkManager to create a worker that free the main UI from being encumbered with a long running task. You have learnt how to pass data to and from the worker and in doing so, implemented your own class that extends Worker.

**Homework Task (on your own for next week):** Once the worker has completed, click the button on the UI to start it again, note what happens. Look at the code and determine why, once you have done that, come up with a solution. This may stretch you a little and we will discuss it in the next class, sometimes personal investigation and problem solving will enhance your learning instead of wrote directions.

**Appendix.**

**MainActvity.java**

**package** com.example.lab4;  
  
  
**import** androidx.annotation.Nullable;  
**import** androidx.appcompat.app.AppCompatActivity;  
**import** androidx.lifecycle.Observer;  
**import** androidx.work.Data;  
**import** androidx.work.OneTimeWorkRequest;  
**import** androidx.work.WorkInfo;  
**import** androidx.work.WorkManager;  
  
**import** android.os.Bundle;  
**import** android.view.View;  
**import** android.widget.ArrayAdapter;  
**import** android.widget.Button;  
**import** android.widget.Spinner;  
**import** android.widget.TextView;  
  
**public class** MainActivity **extends** AppCompatActivity {  
  
 **private** Button **startWorkBtn** =**null** ;  
 **private** TextView **txtView** = **null** ;  
 **private** Spinner **spinner** = **null** ;  
  
 **private** String[] **urls** ;  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
  
 *// initialise the class member with some data* initURLS();  
  
 *// initialise the UI widgets* **startWorkBtn** = (Button) findViewById(R.id.***WorkButton***);  
 **txtView** = (TextView) findViewById(R.id.***textView***);  
 **spinner** = (Spinner) findViewById(R.id.***spinner***);  
  
 *// Create an ArrayAdapter using the string array and a default spinner layout* ArrayAdapter<CharSequence> adapter = ArrayAdapter.*createFromResource*(**this** ,  
 R.array.***planets\_array***, android.R.layout.***simple\_spinner\_item***);  
  
 *// Specify the layout to use when the list of choices appears* adapter.setDropDownViewResource(android.R.layout.***simple\_spinner\_dropdown\_item***);  
  
 *// Apply the adapter to the spinner* **spinner**.setAdapter(adapter);  
  
  
 *// data to pass to worker* Data data = **new** Data.Builder().putStringArray(BackgroundWorker.***URLS***, **urls**)  
 .build();  
  
 *// build the work request (BackgroundWorker.class will do the 'work' for us* **final** OneTimeWorkRequest workRequest =  
 **new** OneTimeWorkRequest.Builder(BackgroundWorker.**class**)  
 .setInputData(data)  
 .build();  
  
  
  
  
 **startWorkBtn**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
 **txtView**.setText(**"Starting Work ....."** + **"\n"**);  
 **startWorkBtn**.setEnabled(**false**);  
 WorkManager.*getInstance*(getApplication()).enqueue(workRequest);  
 }  
 });  
  
 *// listen for changes of the worker* WorkManager.*getInstance*(**this**.getApplication()).getWorkInfoByIdLiveData(workRequest.getId())  
 .observe(**this**, **new** Observer<WorkInfo>() {  
  
 @Override  
 **public void** onChanged(@Nullable WorkInfo workInfo) {  
  
 *//Displaying the status into TextView* **txtView**.append(workInfo.getState().name() + **"\n"**);  
  
 *// check to see if the worker has completed* **if**(workInfo != **null** && workInfo.getState().isFinished()) {  
 **txtView**.append(workInfo.getOutputData().getString(BackgroundWorker.***RETURNTEXT***) + **"\n"**);  
 **startWorkBtn**.setEnabled(**true**);  
 }  
 }  
 });  
 }  
  
  
 *// this will be the data to pass to the worker  
 // init a class member of MainActivity* **private void** initURLS(){  
 **try** {  
 **urls** = **new** String[]{  
 **new** String(**"http://www.amazon.com/somefiles.pdf"** ),  
 **new** String(**"http://www.wrox.com/somefiles.pdf"** ),  
 **new** String(**"http://www.google.com/somefiles.pdf"** ),  
 **new** String(**"http://www.learn2develop.net/somefiles.pdf"** ),  
 **new** String(**"http://www.wrox.com/somefiles.pdf"** )};  
 }**catch** (Exception e){  
 e.printStackTrace();  
 }  
 }  
  
  
}

**BackgroundWorker.java**

**package** com.example.lab4;  
  
**import** android.content.Context;  
**import** android.util.Log;  
  
**import** androidx.annotation.NonNull;  
**import** androidx.work.Data;  
**import** androidx.work.Worker;  
**import** androidx.work.WorkerParameters;  
  
**public class** BackgroundWorker **extends** Worker {  
  
 **public static final** String ***URLS*** = **"URLS"**; *// from the MainActivity* **public static final** String ***RETURNTEXT*** = **"receivedWorkerText"**; *// to the MainActivity* **public** BackgroundWorker(  
 @NonNull Context context,  
 @NonNull WorkerParameters params) {  
 **super**(context, params);  
 }  
  
 @Override  
 **public** Result doWork() {  
  
 Context appContext = getApplicationContext();  
  
 **try** {  
 *// do something that may take some time* downloadFile(getInputData().getStringArray(***URLS***));  
  
 *// Indicate the work finished successfully with the Result* Data data = **new** Data.Builder()  
 .putString(***RETURNTEXT***, **"Work Done!!"**)  
 .build();  
  
 **return** Result.*success*(data);  
 } **catch** (Throwable throwable) {  
 Log.*e*(**"ERROR"**, **"doWork() Failed"**, throwable);  
 *// Indicate the work failed with the Result* Data data = **new** Data.Builder()  
 .putString(***RETURNTEXT***, **"Work Failed!!"**)  
 .build();  
 **return** Result.*failure*();  
 }  
 }  
  
  
 **private void** downloadFile(String[] urls) {  
 **try** {  
  
 **for** (String url: urls) {  
 *//---simulate taking some time to download a file---* Thread.*sleep*( 2000);  
 }  
  
 } **catch** (InterruptedException e) {  
 e.printStackTrace();  
 }  
  
 }  
}