ViewModel LiveData Java Thread - In class Lab and Project

```xml
<?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">

    <Button
        android:id="@+id/bStart"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginBottom="8dp"
        android:onClick="doStart"
        android:text="Start"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.254"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.517" />

    <Button
        android:id="@+id/bCancel"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginBottom="8dp"
        android:onClick="doCancel"
        android:text="Cancel"
        android:enabled="false"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.713"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.517" />

    <ProgressBar
        android:id="@+id/progressBar1"
```

```xml
        style="?android:attr/progressBarStyleHorizontal"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:layout_constraintBottom_toTopOf="@+id/textView2"
        app:layout_constraintTop_toBottomOf="@+id/bCancel" />

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="32dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:text="Results will go here"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent" />


</androidx.constraintlayout.widget.ConstraintLayout>
```

Your UI should look like this

ViewModel_LiveData_Thread_DEMO

START   CANCEL

Results will go here
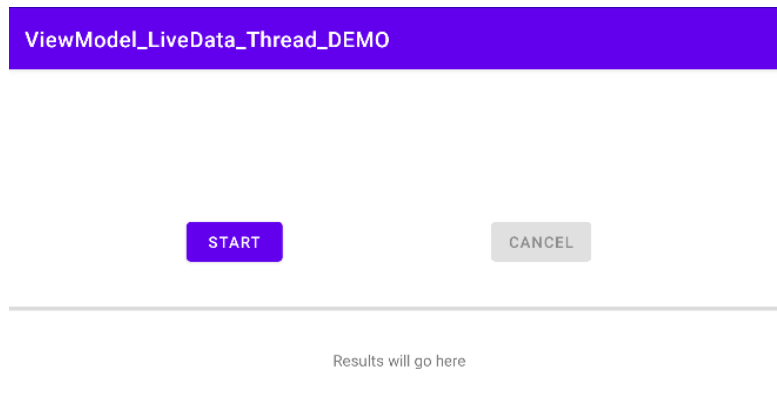
Next hover over doStart and doCancel to add those methods in MainActivity.java

```java
public class DataVM extends ViewModel {
}
```

Add a thread to DataVM, dont make it static, we dont care about implicit references anymore since we are decoupling the thread from the activity
```java
public class MyThread extends Thread {

    //used to ask thread to stop
    public boolean stopRequested=false;
    private int NUMBER_TICKS=100;

    public void run() {
        for (int i=0;i<NUMBER_TICKS;i++){
            //if MainACtivity has asked us to stop then break
            if(stopRequested)
                break;

            //need to notify Mainactivity of progress

            //sleep for 1/2 sec
            try {
                Thread.sleep(50);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }

        //need to notify mainactivity that we are done
        //so it can enable/disable buttons

    }
}
```

also add a constructor and MyThread mt member variable to DataVM
```java
public class DataVM extends ViewModel {

    private MyThread mt;
    public DataVM() {

    }
```

```java
public void startThread(){
    //if I have not started a thread
    //or if I have and its finished
    //then start another
    if(mt==null || !mt.isAlive()) {
        mt = new MyThread();
        mt.start();
    }
}

public void stopThread(){
    //if null we are done
    if(mt==null)
        return;

    //ask for it to stop
    mt.stopRequested=true;

    try {
        //wait until its done
        //this is risky, it may take a while
        //your thread has to be designed to
        //handle this
        mt.join();
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    mt=null; //GC this thread
}
```

```java
private TextView    tv;
private Button      bStart;
private Button      bCancel;
private ProgressBar pBar;
//persists accross config changes
DataVM myVM;
```

```java
tv = findViewById(R.id.textView2);
bStart= findViewById(R.id.bStart);
bCancel = findViewById(R.id.bCancel);

pBar = findViewById(R.id.progressBar1);
```

```java
// Create a ViewModel the first time the system calls an activity's
// onCreate() method.  Re-created activities receive the same
// MyViewModel instance created by the first activity.
myVM = new ViewModelProvider(this).get(DataVM.class);
```

```java
public void doStart(View view) {
    myVM.startThread();
        pBar.setProgress(0);

}

public void doCancel(View view) {
    myVM.stopThread();
}
```

Go over *Why use ViewModel and LiveData* `lecture`
Enter LiveData – An observer pattern where all the data is kept in the ViewModel and the Activity is notified when the data changes, the framework correctly manages all the lifecycle events for you.

In ViewModel (DataVM)

Add MutableLiveData listeners and initialize in DataVM's constructor, anytime we want to change the data we use setValue (from main thread) or postValue from a new thread, DataVM now looks like this with new bits in `light gray`

```java
public class DataVM extends ViewModel {
    //lets add some livedata
    private MutableLiveData<Integer> progress ;
    public MutableLiveData<Integer> getCurrentProgress() {
        return progress;
    }

    //if any observer wants to react to a running thread
    // (for instance if an activity wants to configure its start and stop buttons)
    private MutableLiveData<Boolean> isThreadRunning;
    public MutableLiveData<Boolean> getThreadState() {
        return isThreadRunning;
    }

    :
    public DataVM() {
        super();

        //initialize as appropriate
        progress = new MutableLiveData<Integer>();
        progress.setValue(0);    //initialize

        //a tough one is the thread running or not
        isThreadRunning = new MutableLiveData<Boolean>();
        isThreadRunning.setValue(false);
    }

    public void startThread(){
        //if I have not started a thread
        //or if I have and its finished
```

```java
        //then start another
        if(mt==null || !mt.isAlive()) {
            mt = new MyThread();
            mt.start();
            isThreadRunning.setValue(true);
        }
    }

    public void stopThread(){
        //if null we are done
        if(mt==null)
            return;

        //ask for it to stop
        mt.stopRequested=true;

        try {
            //wait until its done
            //this is risky, it may take a while
            //your thread has to be designed to
            //handle this
            mt.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        mt=null;  //GC this thread

        isThreadRunning.setValue(false);
    }
```

```java
public class MyThread extends Thread {

    //used to ask thread to stop
    public boolean stopRequested=false;
    private int NUMBER_TICKS=100;

    public void run() {
        for (int i=0;i<NUMBER_TICKS;i++){
            //if MainACtivity has asked us to stop then break
            if(stopRequested)
                break;

            //need to notify Mainactivity of progress
            //can only postValue from background thread cannot use setValue
            progress.postValue(i);
            //sleep for 1/2 sec
            try {
                Thread.sleep(50);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }

        //need to notify mainactivity that we are done
        //so it can enable/disable buttons
        //if we are done then say so
        isThreadRunning.postValue(false);
    }
}
```

```java
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    tv = findViewById(R.id.textView2);
    bStart= findViewById(R.id.bStart);
    bCancel = findViewById(R.id.bCancel);
    pBar = findViewById(R.id.progressBar1);

    myVM = new ViewModelProvider(this).get(DataVM.class);

    // Create the observer which updates the UI.
    final Observer<Integer> cntrObserver = new Observer<Integer>() {
        @Override
        public void onChanged(@Nullable final Integer newInt) {
            // Update the UI, in this case, a TextView.
            tv.setText("The new cnt=" +Integer.toString(newInt));
            pBar.setProgress(newInt);
        }
    };
    //now observe
    myVM.getCurrentProgress().observe(this,cntrObserver);

    final Observer<Boolean> isThreadRunningObserver = new Observer<Boolean>() {
        @Override
        public void onChanged(Boolean aBoolean) {
            bStart.setEnabled(!aBoolean);
            bCancel.setEnabled(aBoolean);
            pBar.setProgress(0);
        }
    };
    //now observe
    myVM.getThreadState().observe(this,isThreadRunningObserver);
}
```

You are done. See ViewModel_LiveData_Thread project.