**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

🙤🙧🟍🙥🙦



**PHÁT TRIỂN ỨNG DỤNG TRÊN THIẾT BỊ DI ĐỘNG**

**NT118.J12**

**BÀI TẬP NHÓM SỐ 4**

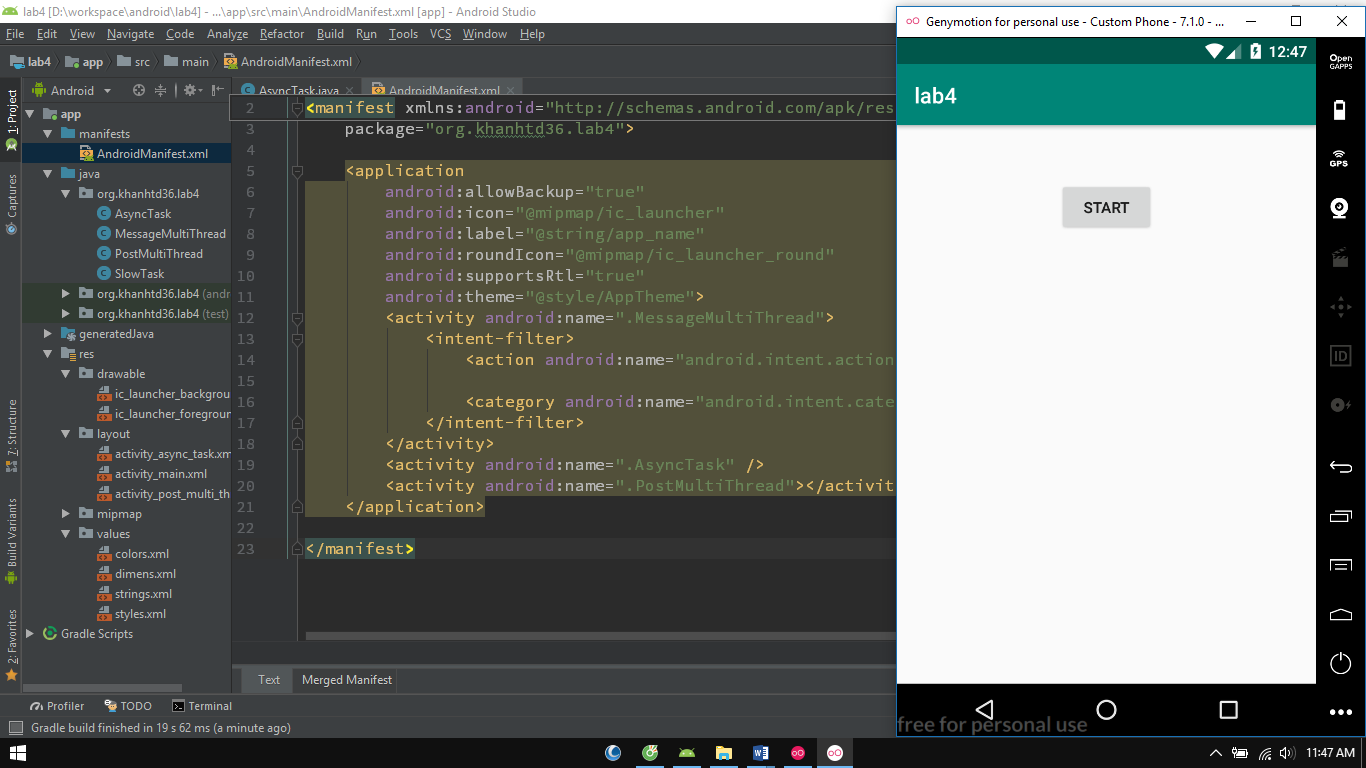
* **GV: Thái Huy Tân**

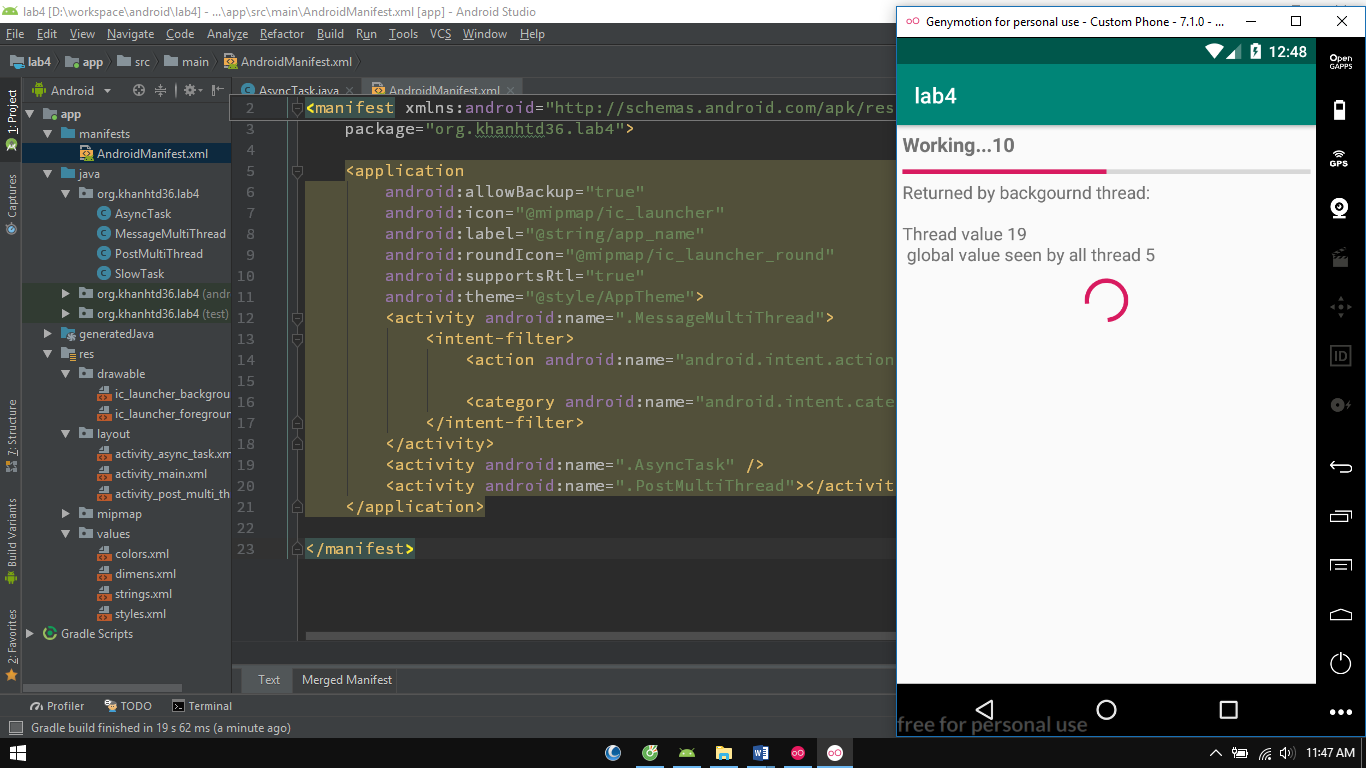
|  |  |  |
| --- | --- | --- |
| Nhóm 11 | | |
| **STT** | **Họ tên** | **MSSV** |
| 1 | Trương Thị Bích Chi | 15520062 |
| 2 | Lê Hồng Ngọc | 15520553 |
| 3 | Đỗ Trọng Nhân | 15520566 |

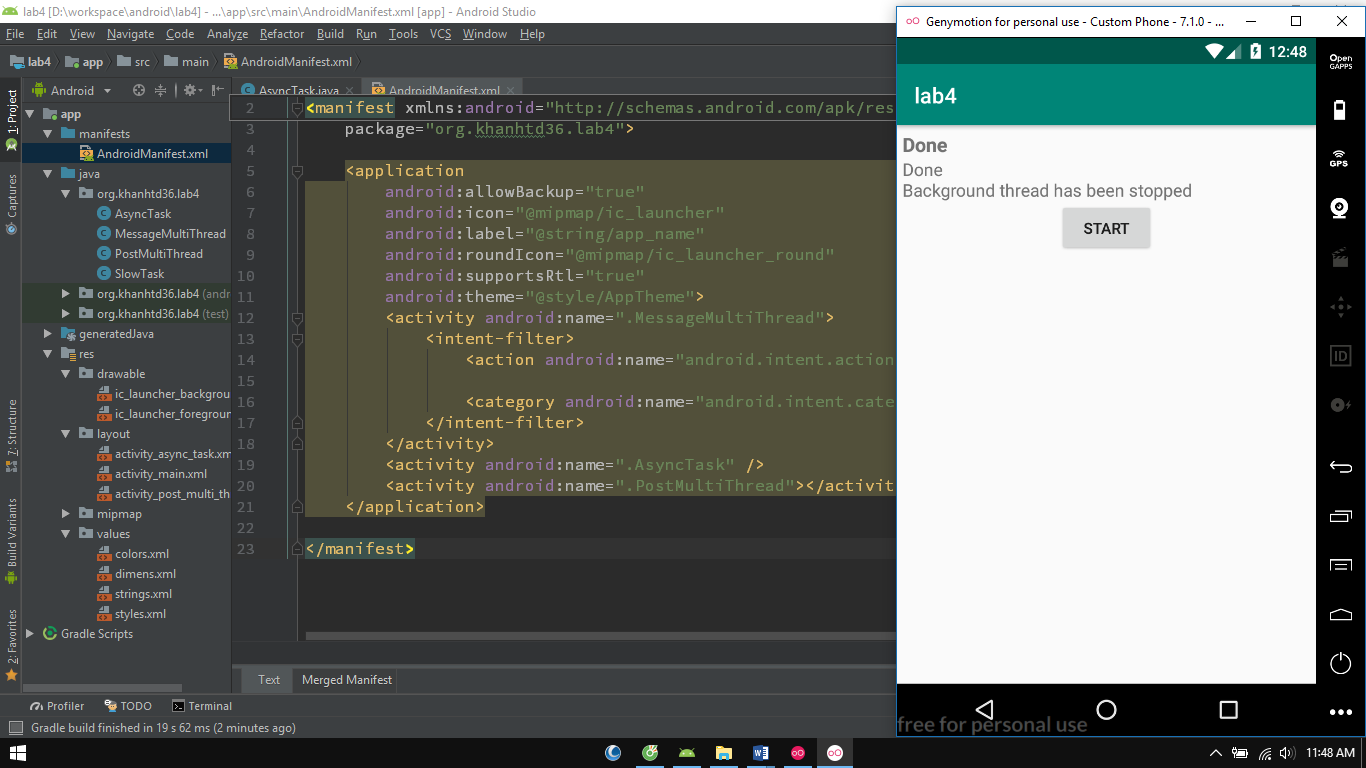
**TP. HỒ CHÍ MINH – 02/11/2018**

# MultiThread sử dụng Message:

## Kết quả màn hình:







## Source:

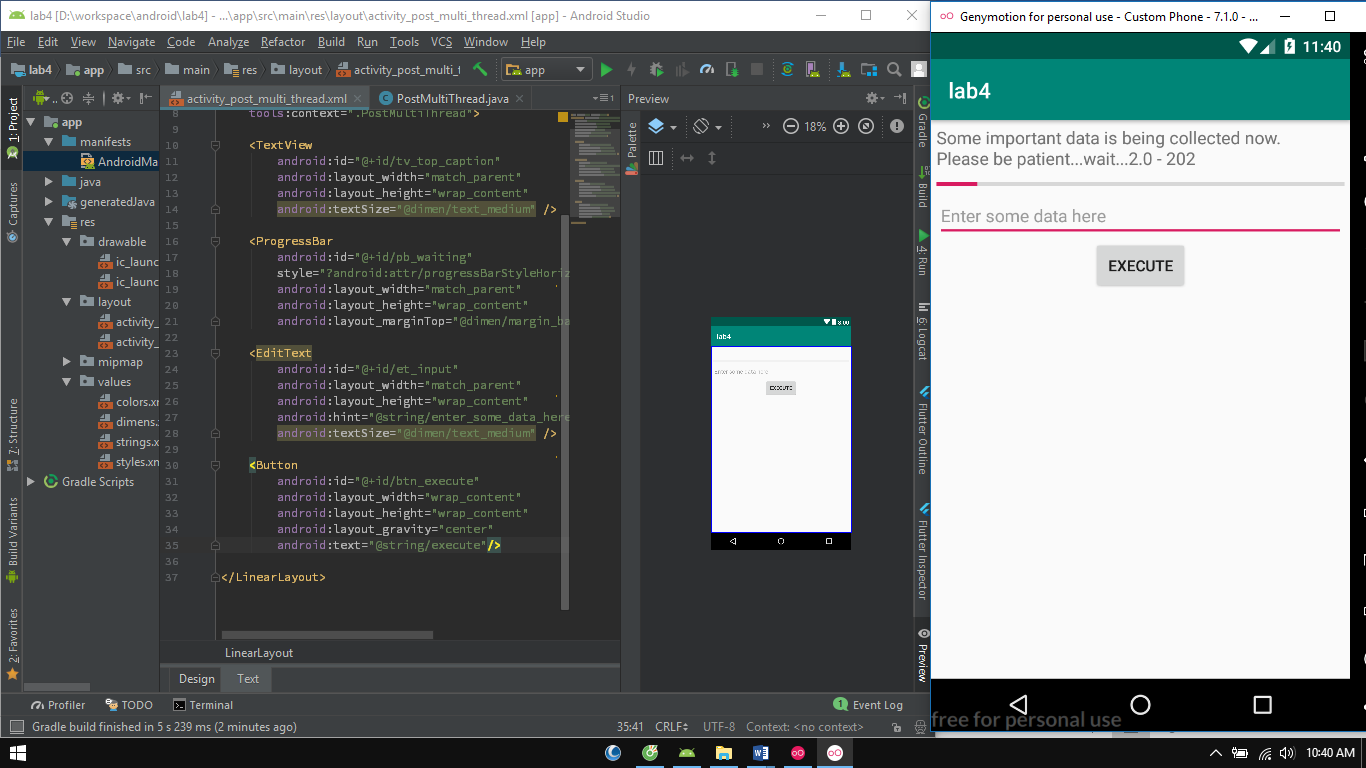
* + Dimens.xml:
* <?xml version="1.0" encoding="utf-8"?>  
  <resources>  
   <dimen name="margin\_base">5dp</dimen>  
   <dimen name="text\_small">14dp</dimen>  
   <dimen name="text\_medium">16dp</dimen>  
   <dimen name="text\_medium\_large">18dp</dimen>  
   <dimen name="text\_large">20dp</dimen>  
  </resources>
  + String.xml:
* <resources>  
   <string name="app\_name">lab4</string>  
   <string name="start">Start</string>  
   <string name="returned\_by\_bg\_thread"> Returned by backgournd thread: \n\n</string>  
   <string name="done\_background\_thread\_has\_been\_stopped"> Done \nBackground thread has been stopped</string>  
   <string name="done">Done</string>  
   <string name="working">Working...</string>  
   <string name="global\_value\_seen">\n global value seen by all thread</string>  
  </resources>
  + activity\_main.xml:
* <?xml version="1.0" encoding="utf-8"?>  
  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
   xmlns:tools="http://schemas.android.com/tools"  
   android:layout\_width="match\_parent"  
   android:layout\_height="match\_parent"  
   android:orientation="vertical"  
   android:padding="@dimen/margin\_base"  
   tools:context=".MainActivity">  
    
    
   <TextView  
   android:id="@+id/tv\_working"  
   android:layout\_width="match\_parent"  
   android:layout\_height="wrap\_content"  
   android:textSize="@dimen/text\_medium\_large"  
   android:textStyle="bold" />  
    
   <ProgressBar  
   android:id="@+id/pb\_first"  
   style="?android:attr/progressBarStyleHorizontal"  
   android:layout\_width="match\_parent"  
   android:layout\_height="wrap\_content"  
   android:layout\_marginTop="@dimen/margin\_base" />  
    
   <TextView  
   android:id="@+id/tv\_return"  
   android:layout\_width="match\_parent"  
   android:layout\_height="wrap\_content"  
   android:textSize="@dimen/text\_medium" />  
    
   <FrameLayout  
   android:layout\_width="match\_parent"  
   android:layout\_height="wrap\_content">  
    
   <Button  
   android:id="@+id/btn\_start"  
   android:layout\_width="wrap\_content"  
   android:layout\_height="wrap\_content"  
   android:layout\_gravity="center"  
   android:text="@string/start" />  
    
   <ProgressBar  
   android:id="@+id/pb\_second"  
   android:layout\_width="match\_parent"  
   android:layout\_height="wrap\_content"  
   android:layout\_gravity="center"  
   android:layout\_marginTop="@dimen/margin\_base" />  
   </FrameLayout>  
    
  </LinearLayout>
  + MainActivity.java:
* package org.khanhtd36.lab4;  
    
  import android.os.Handler;  
  import android.os.Message;  
  import android.support.v7.app.AppCompatActivity;  
  import android.os.Bundle;  
  import android.view.View;  
  import android.widget.Button;  
  import android.widget.ProgressBar;  
  import android.widget.TextView;  
    
  import java.util.Random;  
    
  public class MainActivity extends AppCompatActivity {  
    
   private ProgressBar pbFirst, pbSecond;  
   private TextView tvMsgWorking, tvMsgReturned;  
   private boolean isRunning;  
   private int MAX\_SEC;  
   private int intTest;  
   private Thread bgThread;  
   private Handler handler;  
   private Button btnStart;  
    
   @Override  
   protected void onCreate(Bundle savedInstanceState) {  
   super.onCreate(savedInstanceState);  
   setContentView(R.layout.*activity\_main*);  
    
   findViewByIds();  
   initVariables();  
    
   // Handle clickListenner  
   btnStart.setOnClickListener(new View.OnClickListener() {  
   @Override  
   public void onClick(View v) {  
   isRunning = true;  
   pbFirst.setVisibility(View.*VISIBLE*);  
   pbSecond.setVisibility(View.*VISIBLE*);  
   btnStart.setVisibility(View.*GONE*);  
   bgThread.start();  
   }  
   });  
   }  
    
   @Override  
   protected void onStart() {  
   super.onStart();  
    
   initBgThread();  
   }  
    
   @Override  
   protected void onStop() {  
   super.onStop();  
   isRunning=false;  
   }  
    
   private void findViewByIds(){  
   pbFirst = findViewById(R.id.*pb\_first*);  
   pbSecond = findViewById(R.id.*pb\_second*);  
   tvMsgWorking = findViewById(R.id.*tv\_working*);  
   tvMsgReturned = findViewById(R.id.*tv\_return*);  
   btnStart = findViewById(R.id.*btn\_start*);  
   }  
    
   private void initVariables (){  
   isRunning = false;  
   MAX\_SEC = 20;  
   intTest = 1;  
   pbFirst.setMax(MAX\_SEC);  
   pbFirst.setProgress(0);  
    
   //Init Views  
   pbFirst.setVisibility(View.*GONE*);  
   pbSecond.setVisibility(View.*GONE*);  
    
   handler = new Handler(){  
   @Override  
   public void handleMessage(Message msg) {  
   super.handleMessage(msg);  
    
   String returnedValue = (String) msg.obj;  
   //Do something with the value sent by background thread here...  
   tvMsgReturned.setText(getString(R.string.*returned\_by\_bg\_thread*)+returnedValue);  
   pbFirst.incrementProgressBy(2);  
    
   if(pbFirst.getProgress()==MAX\_SEC){  
   tvMsgReturned.setText(getString(R.string.*done\_background\_thread\_has\_been\_stopped*));  
   tvMsgWorking.setText(getString(R.string.*done*));  
   pbFirst.setVisibility(View.*GONE*);  
   pbSecond.setVisibility(View.*GONE*);  
   btnStart.setVisibility(View.*VISIBLE*);  
   isRunning = false;  
   }  
   else {  
   tvMsgWorking.setText(getString(R.string.*working*)+pbFirst.getProgress());  
    
   }  
   }  
   };  
   }  
    
   private void initBgThread(){  
   bgThread = new Thread(new Runnable() {  
   @Override  
   public void run() {  
   try{  
   for (int i = 0; i<MAX\_SEC && isRunning; i++){  
   //Sleep one second  
   Thread.*sleep*(1000);  
    
   Random rnd = new Random();  
   //This is a locally generated value  
   String data = "Thread value " +(int) rnd.nextInt(101);  
    
   // We can see change (global) class variables  
   data += getString(R.string.*global\_value\_seen*) + " "+ intTest;  
   intTest++;  
    
   //If thread is still alive send the message  
   if(isRunning){  
   // Resquest a message token and put some data in it  
   Message msg = handler.obtainMessage(1, (String) data);  
   handler.sendMessage(msg);  
   }  
    
   }  
   }  
   catch (Throwable t){  
    
   }  
   }  
   });  
   }  
  }

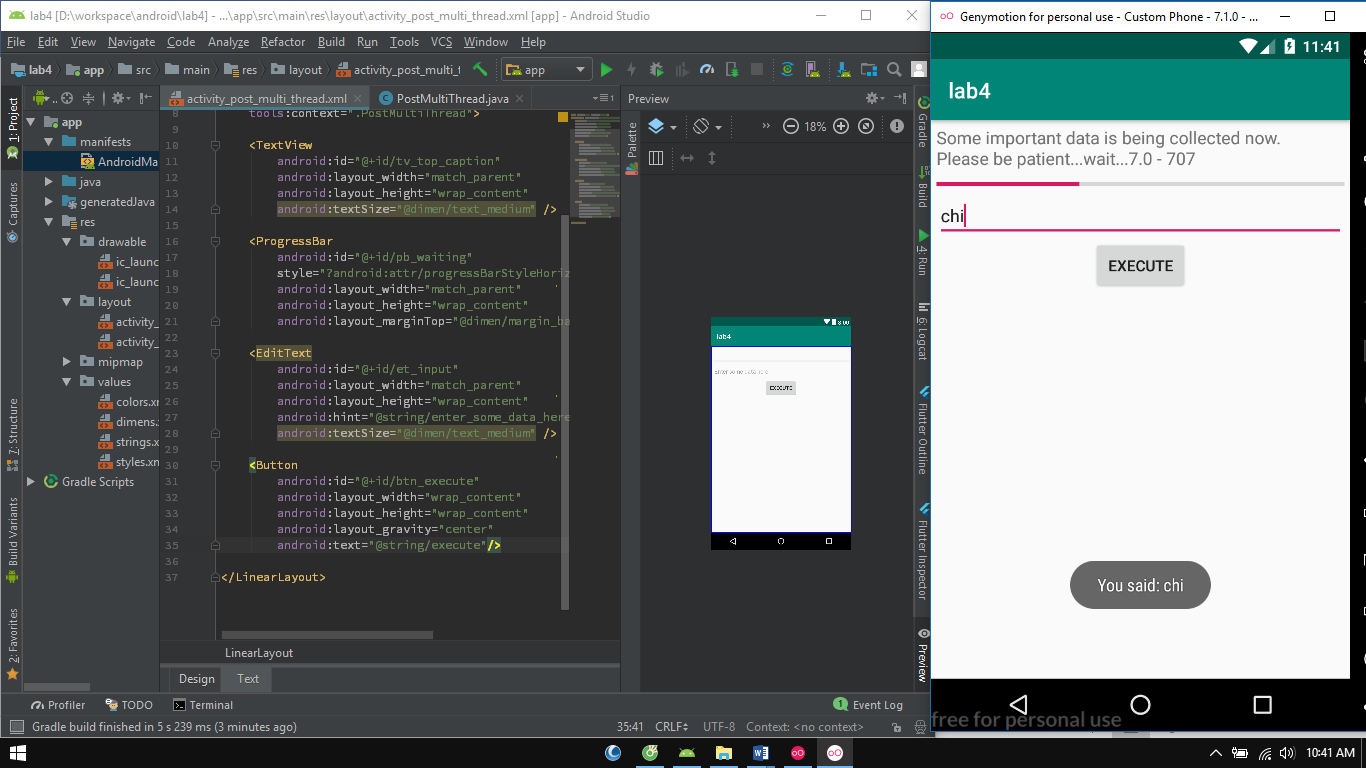
## Giải thích tổng quát:

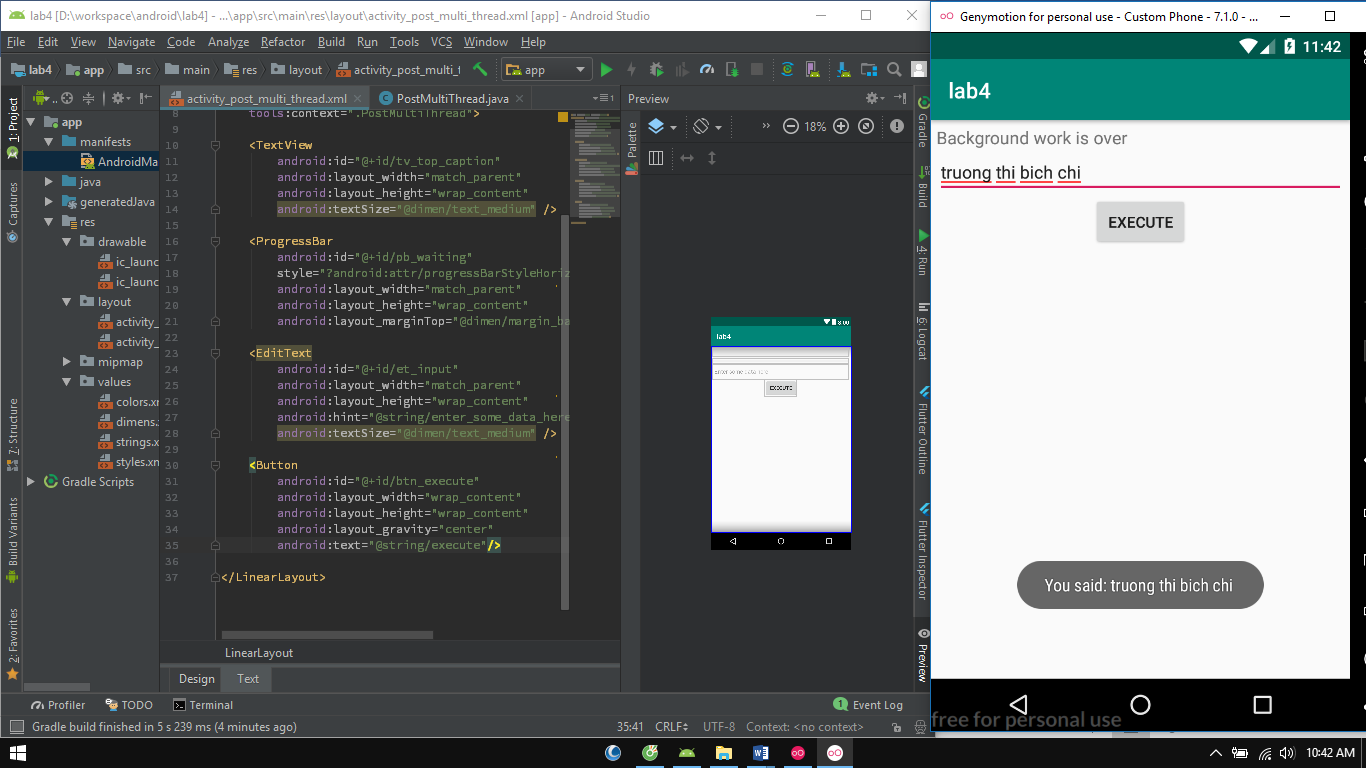
* + File message\_multi\_thread.xml:
    - TextField working: Dùng để hiển thị số giây khi thread đang chạy.
    - ProgressBar: Style là chiều ngang Horizontal để hiển thị tiến trình của thread.
    - TextField return: Dùng để hiển thị message nhận được thông qua handler.
    - FrameLayout: Button start để khởi chạy thread.và ProgressBar để hiển thị tiến trình thread.
  + File MessageMultiThread.java:
    - Tạo biến toàn cục boolean isRunning (giá trị ban đầu là false, sau khi thread được khởi tạo thì cập nhật giá trị true), int MAX\_SEC (thời gian tối đa để chạy thread), int intTest (hiển thị số lần thay đổi giá trị random), Thread bgThread và Handler handler.
    - OnCreate: gán giá trị cho MAX\_SEC là 20 giây, sau đó định nghĩa **handler** theo phương thức override để sử lý hiển thị message nhận được: Object trong message nhận được sẽ hiển thị lên TextField tvMsgReturned, sau đó tăng progress của ProgressBar đầu tiên lên 2 đơn vị. Tăng đến khi progress = MAX\_SEC (20 giây) thì thông báo thread chạy xong lên UI.
    - OnStart: Khởi tạo **Thread**, định nghĩa trong hàm run: Tạo vòng lập, nếu thời gian để chạy thread < MAX\_SEC và isRunning = true thì tạo biến random, sau đó tạo biến data có chứa giá trị random cùng với số lần giá trị thay đổi rồi gán data đó vào 1 message, dùng phương thức Handler.sendMessage để gửi dữ liệu từ thread thông qua handler.
    - Cài đặt sự kiện click vào button Start: bgThread (được định nghĩa ở OnStart) bắt đầu thực thi, mỗi khi handler gửi message nào từ thread thì handler lại được khởi tạo biến mới (được định nghĩa ở onCreate), sau đó lấy dữ liệu trong message ra và hiển thị lên UI cho tới khi hết thời gian thực thi (20s).

# MultiThread sử dụng Post:

## Kết quả màn hình:







## Source:

* + Dimens.xml:
* <?xml version="1.0" encoding="utf-8"?>  
  <resources>  
   <dimen name="margin\_base">5dp</dimen>  
   <dimen name="text\_small">14dp</dimen>  
   <dimen name="text\_medium">16dp</dimen>  
   <dimen name="text\_medium\_large">18dp</dimen>  
   <dimen name="text\_large">20dp</dimen>  
  </resources>
  + String.xml:
* <resources>  
   <string name="app\_name">lab4</string>  
   <string name="start">Start</string>  
   <string name="returned\_by\_bg\_thread"> Returned by backgournd thread: \n\n</string>  
   <string name="done\_background\_thread\_has\_been\_stopped"> Done \nBackground thread has been stopped</string>  
   <string name="done">Done</string>  
   <string name="working">Working...</string>  
   <string name="global\_value\_seen">\n global value seen by all thread</string>  
   <string name="bg\_work\_is\_over">Background work is over</string>  
   <string name="execute">Execute</string>  
   <string name="enter\_some\_data\_here">Enter some data here</string>  
  </resources>
  + Activity\_post\_multi\_thread.xml:
* <?xml version="1.0" encoding="utf-8"?>  
  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
   xmlns:tools="http://schemas.android.com/tools"  
   android:layout\_width="match\_parent"  
   android:layout\_height="match\_parent"  
   android:orientation="vertical"  
   android:padding="@dimen/margin\_base"  
   tools:context=".PostMultiThread">  
    
   <TextView  
   android:id="@+id/tv\_top\_caption"  
   android:layout\_width="match\_parent"  
   android:layout\_height="wrap\_content"  
   android:textSize="@dimen/text\_medium" />  
    
   <ProgressBar  
   android:id="@+id/pb\_waiting"  
   style="?android:attr/progressBarStyleHorizontal"  
   android:layout\_width="match\_parent"  
   android:layout\_height="wrap\_content"  
   android:layout\_marginTop="@dimen/margin\_base" />  
    
   <EditText  
   android:id="@+id/et\_input"  
   android:layout\_width="match\_parent"  
   android:layout\_height="wrap\_content"  
   android:hint="@string/enter\_some\_data\_here"  
   android:textSize="@dimen/text\_medium" />  
    
   <Button  
   android:id="@+id/btn\_execute"  
   android:layout\_width="wrap\_content"  
   android:layout\_height="wrap\_content"  
   android:layout\_gravity="center"  
   android:text="@string/execute"/>  
    
  </LinearLayout>
  + PostMultiThread.java:

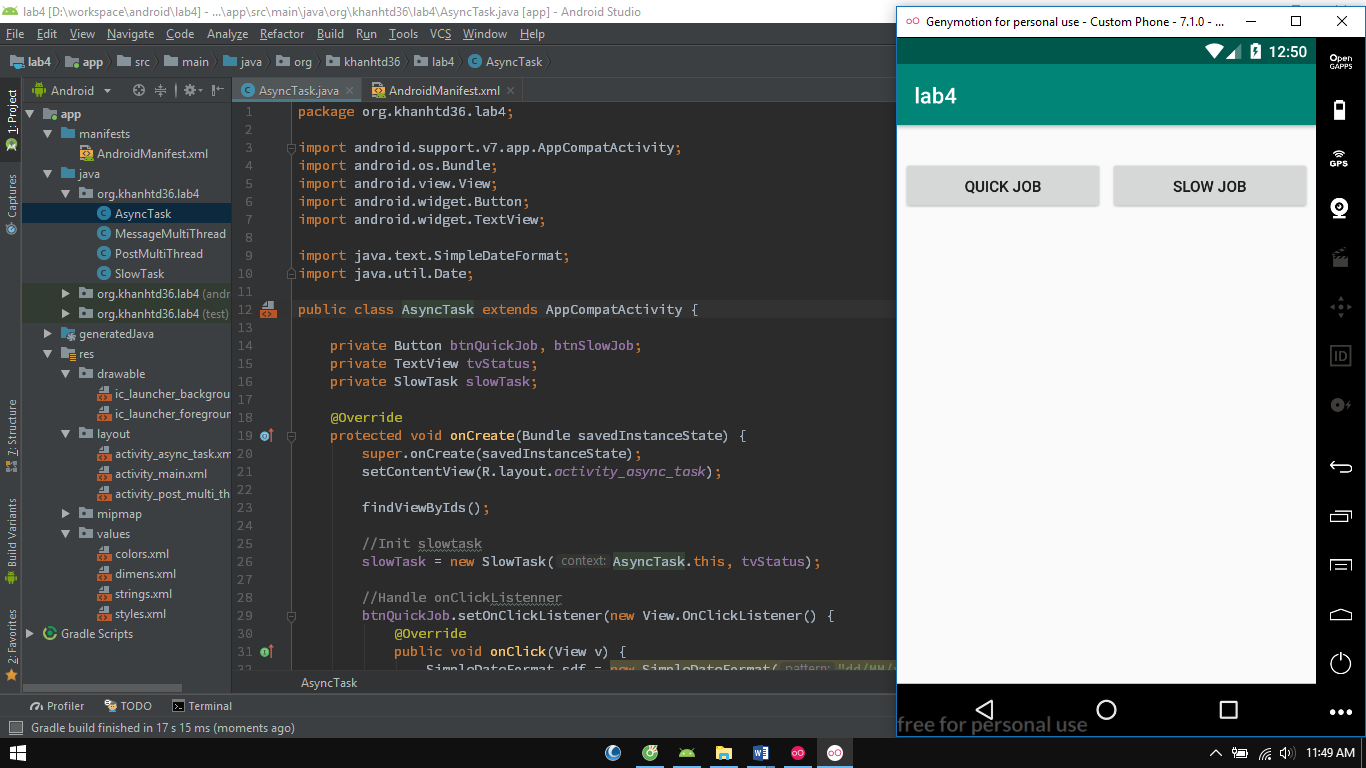
package org.khanhtd36.lab4;  
  
import android.os.Handler;  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
import android.util.Log;  
import android.view.View;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.ProgressBar;  
import android.widget.TextView;  
import android.widget.Toast;  
  
public class PostMultiThread extends AppCompatActivity {  
  
 private ProgressBar pbWaiting;  
 private TextView tvTopCaption;  
 private EditText etInput;  
 private Button btnExecute;  
 private int globalValue, accum;  
 private long startTime;  
 private final String PATIENCE = "Some important data is being collected now.\nPlease be patient...wait...";  
 private Handler handler;  
 private Runnable fgRunnable, bgRunnable;  
 private Thread testThread;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_post\_multi\_thread*);  
  
 findViewByIds();  
 initVariables();  
  
 //Handle onClickListenner  
  
 btnExecute.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 String input =etInput.getText().toString();  
 Toast.*makeText*(PostMultiThread.this, "You said: "+input, Toast.*LENGTH\_SHORT*).show();  
 }  
 });  
  
 //start thread  
 testThread.start();  
 pbWaiting.incrementProgressBy(0);  
 }  
  
 private void findViewByIds(){  
 tvTopCaption = (TextView) findViewById(R.id.*tv\_top\_caption*);  
 pbWaiting = (ProgressBar) findViewById(R.id.*pb\_waiting*);  
 etInput = (EditText) findViewById(R.id.*et\_input*);  
 btnExecute = (Button) findViewById(R.id.*btn\_execute*);  
  
 }  
  
 private void initVariables(){  
 globalValue = 0;  
 accum = 0;  
 startTime = System.*currentTimeMillis*();  
 handler = new Handler();  
  
 fgRunnable = new Runnable() {  
 @Override  
 public void run() {  
 try{  
 //Calculate new value  
 int progressStep = 5;  
 double totalTime = (System.*currentTimeMillis*()-startTime)/1000;  
 synchronized (this){  
 globalValue+=100;  
 }  
 //update UI  
 tvTopCaption.setText(PATIENCE + totalTime + " - "+globalValue);  
 pbWaiting.incrementProgressBy(progressStep);  
 accum += progressStep;  
  
 //check to stop  
 if (accum >= pbWaiting.getMax()){  
 tvTopCaption.setText(getString(R.string.*bg\_work\_is\_over*));  
 pbWaiting.setVisibility(View.*GONE*);  
 }  
 }  
 catch (Exception e){  
 Log.*e*("fgRunnable", e.getMessage());  
 }  
 }  
 };  
  
 bgRunnable = new Runnable() {  
 @Override  
 public void run() {  
 try{  
 for(int i=0; i<20; i++){  
 //sleep 1 second  
 Thread.*sleep*(1000);  
  
 //Nov talk to main thread  
 //Optinally change some global variable such as : globalValue  
  
  
  
 synchronized (this){  
 globalValue +=1;  
 }  
  
 handler.post(fgRunnable);  
 }  
 }  
 catch (Exception e){  
 Log.*e*("bgRunnable", e.getMessage());  
 }  
 }  
  
 };  
  
 testThread = new Thread(bgRunnable);  
 }  
}

## Giải thích tổng quát:

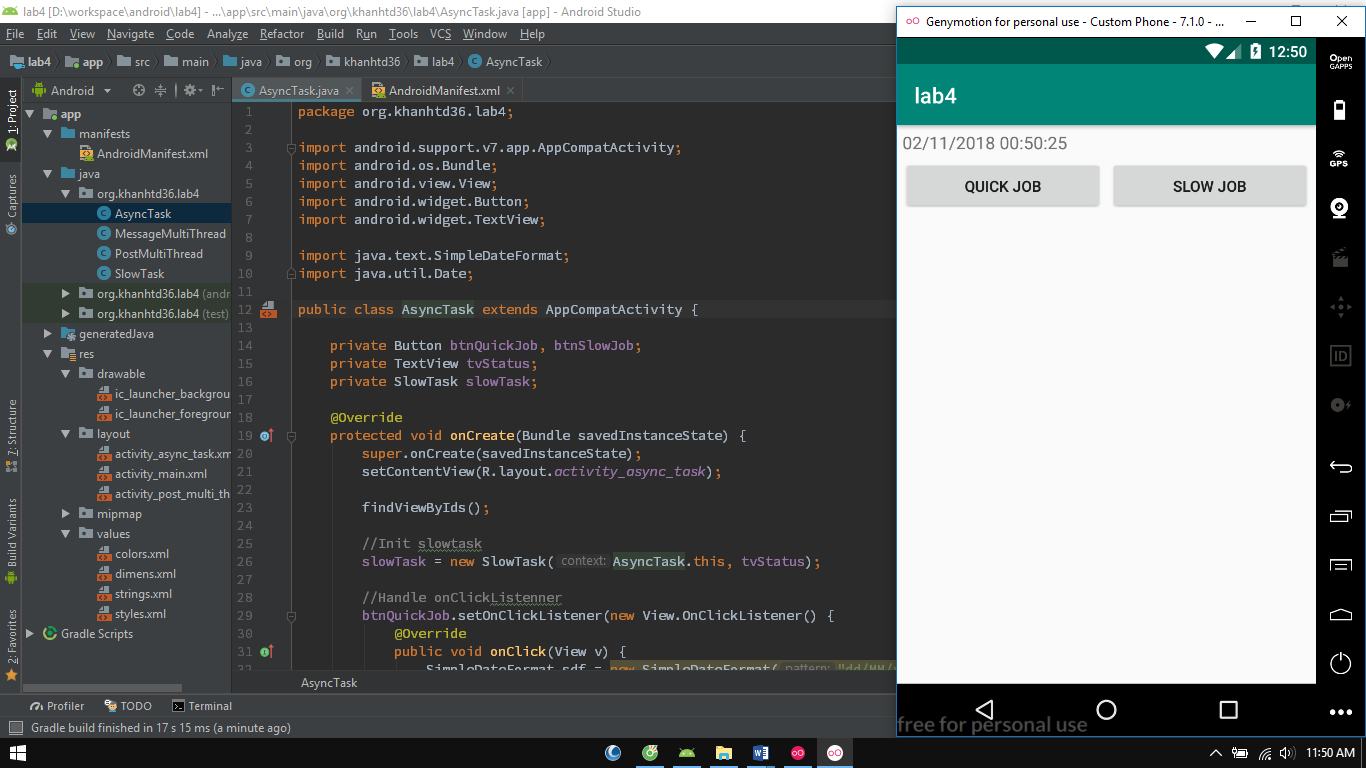
* + Activity\_post\_multi\_thread.xml:
    - TextField: Hiển thị thông tin.
    - ProgressBar: Style Horizontal biểu diễn progress đang chạy.
    - EditText: Nhập dữ liệu từ người dùng.
    - Button: khi click vào sẽ hiển thị dữ liệu trên editText dưới dạng Toask.
  + PostMultiThread.java:
    - Tạo các biến toàn cục int globalValue, accum, Handler handler, Runnable fgRunnable (foreground Runnable) và bgRunnable (background Runnable), Thread testThread.
    - Tại hàm onCreate: Khởi tạo các giá trị: globalValue=0, accum=0, startTime là số mili giây của giờ hiện tại, khởi tạo handler. Định nghĩa 2 runnable:
      * Foreground Runnable: tạo biến totalTime với giá trị là lấy số mili giây của giờ hiện tại (lúc fgRunnable được gọi để thực thi) – startTime và chia cho 1000. Sau đó dùng lệnh synchronized(this) để khóa tất cả các tiến trình khác lại không cho thực thi truy xuất dữ liệu lên biến globalValue để tăng globalValue thêm 100 đơn vị. Tiếp theo tăng tiến độ của progressBar lên 5% (ban đầu mặc định là 100%, do đó có thể dự đoán được globalValue sẽ thay đổi giá trị 20 lần).
      * Background Runnable: Tạo 1 vòng lặp (i=0 đến i<20), cho thread sleep trong vòng 1 giây, sau đó dùng lệnh synchronized(this) để khóa tất cả các tiến trình khác lại không có truy xuất dữ liệu lên biến globalValue để tăng globalValue thêm 1 đơn vị. Sau đó post fgRunnable vào handler để thực thi foreground Runnable.
      * Sau khi định nghĩa 2 runnable xong, testThread được khởi tạo thông qua bgRunnable và startThread này.
      * Thiết lập sự kiện click vào button execute: dữ liệu được nhập trong editField sẽ được hiển thị lên UI dưới dạng Toask.
* Luồng chạy chương trình: khi start testThread, background runnable được chạy. Do vòng lặp 20 lần, globalValue (ban đầu = 0 ) sẽ được tăng lên 1. Sau khi tăng xong, foreground runnable được gọi thông qua handler (câu lệnh handler.post(fgRunnable). Do đó, globalValue lại tiếp tục tăng lên 100 đơn vị thành 101. Tiếp tục vòng lặp thứ 2, globalValue tăng lên thành 2 đơn vị, gọi tiếp foreground runnable thì biến này tăng lên 202. Cứ thế tiếp tục đến hết 20 vòng lặp thì global sẽ tăng lên 2020.

# Asynctask:

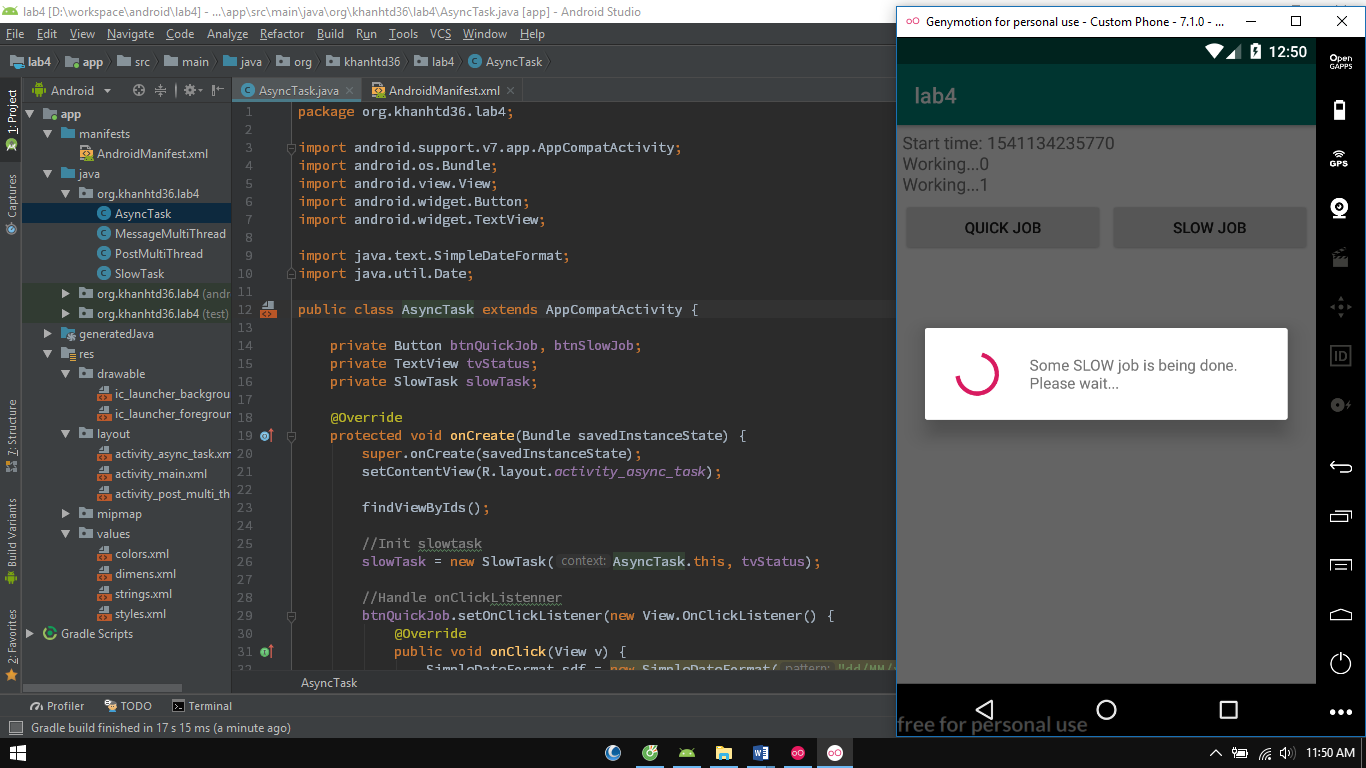
## Kết quả màn hình:

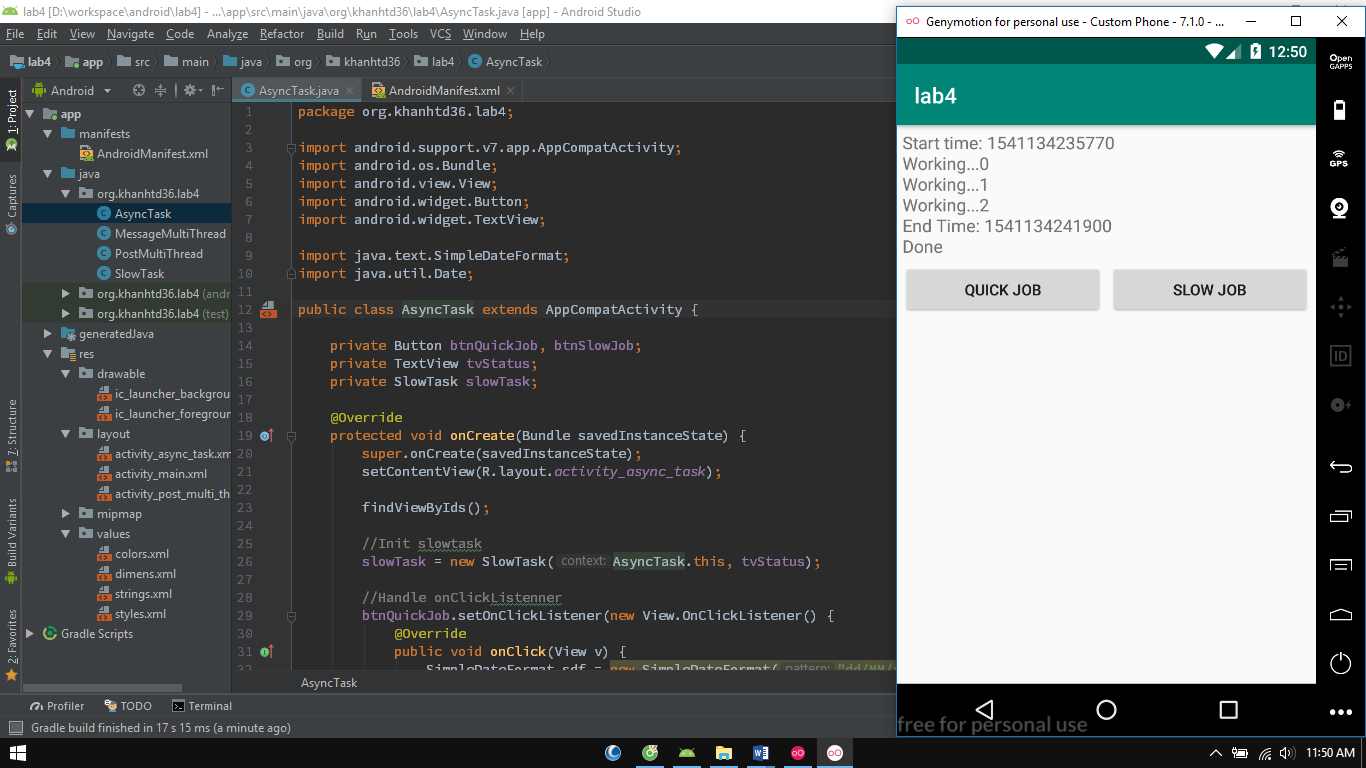


* + Click button Quick Job:



* + Click button Slow Job:





## Source:

* + String.xml:
* <resources>  
   <string name="app\_name">lab4</string>  
   <string name="start">Start</string>  
   <string name="returned\_by\_bg\_thread"> Returned by backgournd thread: \n\n</string>  
   <string name="done\_background\_thread\_has\_been\_stopped"> Done \nBackground thread has been stopped</string>  
   <string name="done">Done</string>  
   <string name="working">Working...</string>  
   <string name="global\_value\_seen">\n global value seen by all thread</string>  
   <string name="bg\_work\_is\_over">Background work is over</string>  
   <string name="execute">Execute</string>  
   <string name="enter\_some\_data\_here">Enter some data here</string>  
   <string name="quick\_job">Quick Job</string>  
   <string name="slow\_job">Slow Job</string>  
   <string name="please\_wait"> Some SLOW job is being done. Please wait...</string>  
  </resources>
  + Activity\_async\_task.xml:
* <?xml version="1.0" encoding="utf-8"?>  
  <LinearLayout 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"  
   android:orientation="vertical"  
   android:padding="@dimen/margin\_base"  
   tools:context=".AsyncTaskActivity">  
    
    
   <TextView  
   android:id="@+id/tv\_status"  
   android:layout\_width="match\_parent"  
   android:layout\_height="wrap\_content"  
   android:textSize="@dimen/text\_medium" />  
    
   <LinearLayout  
   android:layout\_width="match\_parent"  
   android:layout\_height="wrap\_content"  
   android:layout\_marginTop="@dimen/margin\_base"  
   android:orientation="horizontal">  
    
   <Button  
   android:id="@+id/btn\_quick\_job"  
   android:layout\_width="0dp"  
   android:layout\_height="wrap\_content"  
   android:layout\_weight="1"  
   android:text="@string/quick\_job" />  
    
   <Button  
   android:id="@+id/btn\_slow\_job"  
   android:layout\_width="0dp"  
   android:layout\_height="wrap\_content"  
   android:layout\_weight="1"  
   android:text="@string/slow\_job"  
   android:layout\_marginLeft="@dimen/margin\_base"/>  
   </LinearLayout>  
    
  </LinearLayout>
  + AsyncTaskActivity.java:
* package org.khanhtd36.lab4;  
    
  import android.support.v7.app.AppCompatActivity;  
  import android.os.Bundle;  
  import android.view.View;  
  import android.widget.Button;  
  import android.widget.TextView;  
    
  import java.text.SimpleDateFormat;  
  import java.util.Date;  
    
  public class AsyncTaskActivity extends AppCompatActivity {  
    
   private Button btnQuickJob, btnSlowJob;  
   private TextView tvStatus;  
   private SlowTask slowTask;  
    
   @Override  
   protected void onCreate(Bundle savedInstanceState) {  
   super.onCreate(savedInstanceState);  
   setContentView(R.layout.*activity\_async\_task*);  
    
   findViewByIds();  
    
   //Handle onClickListenner  
   btnQuickJob.setOnClickListener(new View.OnClickListener() {  
   @Override  
   public void onClick(View v) {  
   SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy HH:mm:ss");  
   tvStatus.setText(sdf.format(new Date()));  
   }  
   });  
    
    
   btnSlowJob.setOnClickListener(new View.OnClickListener() {  
   @Override  
   public void onClick(View v) {  
   //Init slowtask  
   slowTask = new SlowTask(AsyncTaskActivity.this, tvStatus);  
   slowTask.execute();  
   }  
   });  
   }  
    
   private void findViewByIds() {  
   btnQuickJob = (Button) findViewById(R.id.*btn\_quick\_job*);  
   btnSlowJob = (Button) findViewById(R.id.*btn\_slow\_job*);  
   tvStatus = (TextView) findViewById(R.id.*tv\_status*);  
    
   }  
    
  }

## Giải thích tổng quát:

* + Activity\_async\_task.xml:
    - TextField status để hiển thị thông tin khi thực thi các task.
    - Button QuickJob để gọi lệnh thực thi hiển thị thời gian hiện tại.
    - Button SlowJob để gọi lệnh thực thi AsyncTask.
  + Tạo class SlowTask được kế thừa từ class AsyncTask hỗ trợ từ android, các thông số truyền vào là Params, Progress và Result. Trong hàm doInBackground: Giả lập các task cần nhiều thời gian bằng cách cho thread sleep 3 lần, mỗi lần 2 giây. Sau đó dòng hàm PublishProgress để thông báo cập nhật giá trị biến i (ban đầu = 0, sau mỗi lần tăng lên 1 đơn vị). Lúc này chương trình chạy vào hàm onProgressUpdate, hiển thị giá trị i lên TextField. Sau khi doInBackground chạy xong, hàm onPostExcute sẽ được chạy, hiển thị thông báo thread chạy xong lên TextField.
  + AsyncTaskActivity.java: Tạo biến toàn cục SlowTask slowTask, TextField tvStatus. Trong hàm onCreate thiết lập sự kiện cho 2 button:
    - QuickJob: Hiển thị ngày giờ tại thời điểm click.
    - SlowJob: Lưu ý điểm khác code của thầy: slowTask được tạo mới mỗi khi click vào button, thay vì được tạo trong hàm onCreate. Lí do: Khi click vào button SlowJob lần đầu, task sau khi thực hiện xong sẽ ở tình trạng đã được thực thi, nếu click thêm 1 lần nữa sẽ hiển thị lỗi :



Lúc khởi tạo slowTask thì truyền vào context là AsyncTaskActivity.this và TextField tvStatus để slowTask hiển thị thông tin lên UI. Gọi hàm execute() để AsyncTask được bắt đầu thực thi.

# **Source Code:** [**https://github.com/truongthibichchi/android\_lab4**](https://github.com/truongthibichchi/android_lab4)