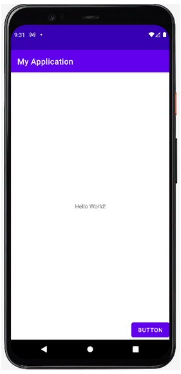
Helpful shortcuts
Alt-insert add constructor
ctrl-o override members

Start with basic activity project
activity.xml
add a button
give the text and the button an ID
add a onClick handler to the button
should look like below



Add a separate class, updatetask that extends from thread
alt-insert to add constructor and override run method

Updatetask.java

```
public class Updatetask extends Thread{
    private MainActivity activity;
   private int cnt=0;   //which thread?


    public Updatetask(MainActivity act,int cnt) {
            this.cnt=cnt;
            attach(act);
    }

    public void attach(MainActivity act){activity=act;}
    public void detach(){activity=null;}

    @Override
    public void run() {
        super.run();


        SystemClock.sleep(5000);

        //now update the textbox in mainactivity
        if (activity != null)
            activity.runOnUiThread(new Runnable() {
                public void run() {
                    TextView tv=activity.findViewById(R.id.tv);
                    tv.setText("Finished Thread "+Integer.toString(cnt));

                }
            });
    }
}
```

in MainActivity.java

```
private int cnt=0;

public void doClick(View view) {
    Updatetask mt=new Updatetask(this,cnt);
    mt.start();
    cnt++;
}
```

But what happens when device is rotated?

add an onStart and onStop method with the intent of detaching the thread from an activity that is being torn down, and reattaching it to the new activity.
But what will hold onto the thread while Activity is being created and destroyed ? You can’t use onSaveInstanceState (a thread is not serializable).

Enter the singleton
Add a class, Mysingleton, move all thread operations over to it.
See week 7 project “In Class Lab using a Java Thread and a simple singleton”