ListActivity in class lab
with and without threaded rows

Create the view that defines each rows appearance in the list (row_layout.xml)

two textviews in a linear layout (horizontal orientation, use wrap_content for both height and width)

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent">
  <TextView
    android:id="@+id/tv_equation"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:text="TextView" />
  <TextView
    android:id="@+id/tv_result"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:text="TextView" />
</LinearLayout>
```

Create Adapter (the brains of the operation). Supplies the Listactivity with 1 row of data at a time via calls to adapters getView(..) method

1. create a new Java class that extends BaseAdapter
Add unimplemented methods(alt-enter)

2. handle some of the easier methods
public int getCount() { //add how many you expect to have
create a const or return the number in your datasource, leave it 0 and you get nothing in the list

3. Add a constructor (hover over class name and hit alt-insert) and pass in a reference to Mainactivity save in a member

```
private final Context ctx;
public Data_Adapter(Context ctx) {  this.ctx = ctx;}
```

4. add an inner Viewholder Class to adapter that points to all fields in **row_layout.xml (used for optimization)**

```
        private static class Viewholder {
                TextView tv_equation;
```

```
TextView tv_result;
int myNumberToDouble;
}
```

5. In Data_Adapter add a class reference to layoutinflator (so you can generate views)

```
private LayoutInflater inflater;
```

6. In GetView()

```
public View getView(int position, View convertView, ViewGroup parent) {
  Viewholder myVH;
  //if cannot recycle, then create a new one, this should only happen
  //with first screen of data (or rows)
  if (convertView == null){
    if (inflater == null)
      inflater =(LayoutInflater)ctx.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
    //get a row
    convertView = inflater.inflate(R.layout.row_layout,null);
    //create a viewholder for effeciency (and for thread usage)
    myVH = new Viewholder();
    //get refs to widgets
    myVH.tv_equation = (TextView)convertView.findViewById(R.id.tv_equation);
    myVH.tv_result = (TextView)convertView.findViewById(R.id.tv_result);
    //marry the viewholder to the convertview row
    convertView.setTag(myVH);
  }
  myVH = (Viewholder)convertView.getTag();
  //this viewholder is going to double the number given in position
  myVH.myNumberToDouble = position;
  String s1 = Integer.toString(myVH.myNumberToDouble);
  //set the first field
  myVH.tv_equation.setText(s1 + " + " + s1 + " = " );
  //set the result (non threaded)
  s1 = Integer.toString(2*myVH.myNumberToDouble);
  myVH.tv_result.setText(s1);
  //set the result (threaded)
  return convertView;
}
```

In MainActivity

1. Extend ListActivity
2. in onCreate(...)

   Get rid of setContentView(...) cause you dont need it anymore

```
// a custom data adapter
myAdapter = new Data_Adapter(this);
setListAdapter(myAdapter);
```

Try it

Want to see which row clicked?

```
@Override
protected void onListItemClick(ListView l, View v, int position, long id) {
    super.onListItemClick(l, v, position, id);
    Toast.makeText(this, "Click ListItem Number " + Integer.toString(position),
Toast.LENGTH_LONG).show();
}
```

Heavy lifting time - Lets do the calculations in a thread and update the result at a later time. Why? Because often rows consists of easy to get data, like the position, and hard to get data, like an image located on another server.

You can't pause the getView dataadapter pipeline while waiting to download image (what would a http timeout do to your apps performance?).

So generate and show all the easy to get stuff and launch a athread to get the harder stuff, when the thread finishes it goes and updates the appropriate view.

Create inner class in adapter :

private class CalculateValueTask **extends** AsyncTask<Integer, Void, Integer>

doInBackground just adds 2 numbers and returns the result

constructor takes a viewholder object

```
private class CalculateValueTask extends AsyncTask<Integer, Void, Integer> {
  private Viewholder myVh;    //ref to a viewholder
  private int origNumbToDouble; //since myVH may be recycled and reused
                //we have to verify that the result we are returning
                //is still what the viewholder wants
  public CalculateValueTask(Viewholder myVh) {
    this.myVh = myVh;
    this.origNumbToDouble = myVh.myNumberToDouble;
  }
  @Override
  protected Integer doInBackground(Integer... integers) {
    try {
      Thread.sleep(25);
    } catch (InterruptedException e) {
      e.printStackTrace();
    }
    return integers[0] + integers[0];
  }
  @Override
  protected void onPostExecute(Integer integer) {
    super.onPostExecute(integer);
    //got a result, if the following are NOT equal
    // then the view has been recycled and is being used by another
    // number DO NOT MODIFY
    if (this.myVh.myNumberToDouble == this.origNumbToDouble){
      //still valid
      //set the result (non threaded)
      String s1 = Integer.toString(2*myVh.myNumberToDouble);
      myVh.tv_result.setText(s1);
```

```
    }
  }
}
```

In Data_Adapters getView(..)

```
//double in another thread
CalculateValueTask myTask = new CalculateValueTask(myVh);
myTask.execute(myVh.myNumbToDouble);
```