Trường Đại học Bách Khoa Hà Nội
Hanoi University of Science and Technology

Chapter 4. Graphical User Interfaces

Hanoi University of Science and Technology

Nguyen Hong Quang
19/8/2012

Nội dung

- Screen rotation
- Saving your state

Screen rotation

- Switching from portrait to landscape
- Change in the phone configuration:
  - Extending or hiding a physical keyboard, for devices that have such sliding keyboards
  - Putting the device in a car or desk dock, or removing it from a dock
  - Changing the locale, and thereby changing the preferred language

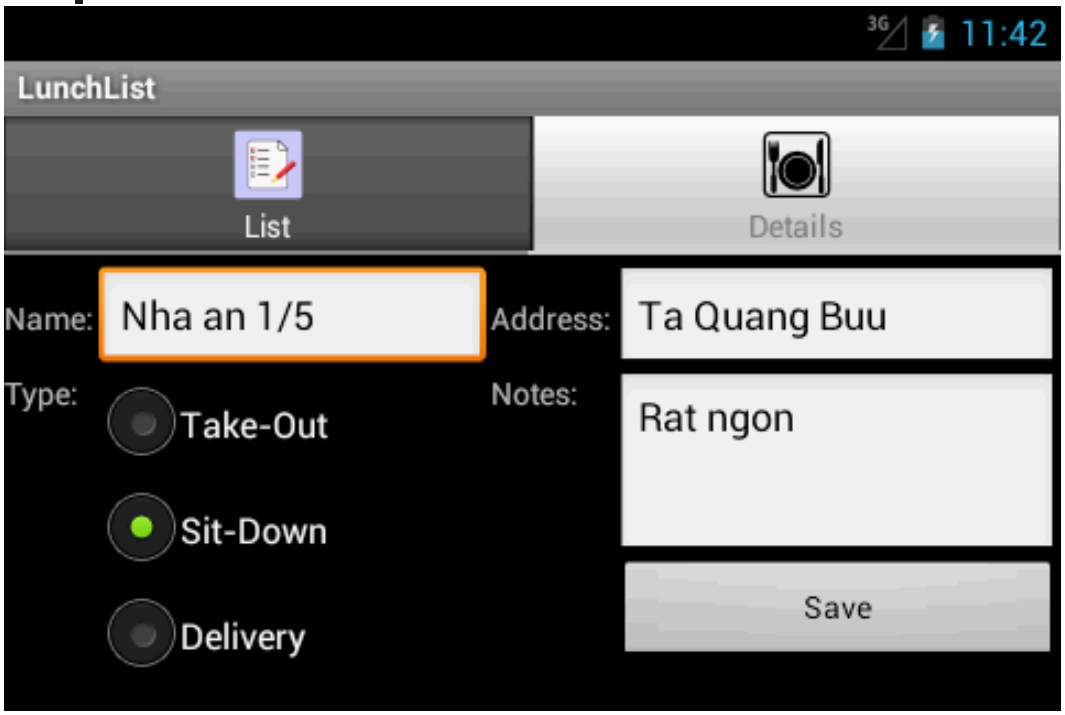
Change in the phone configuration

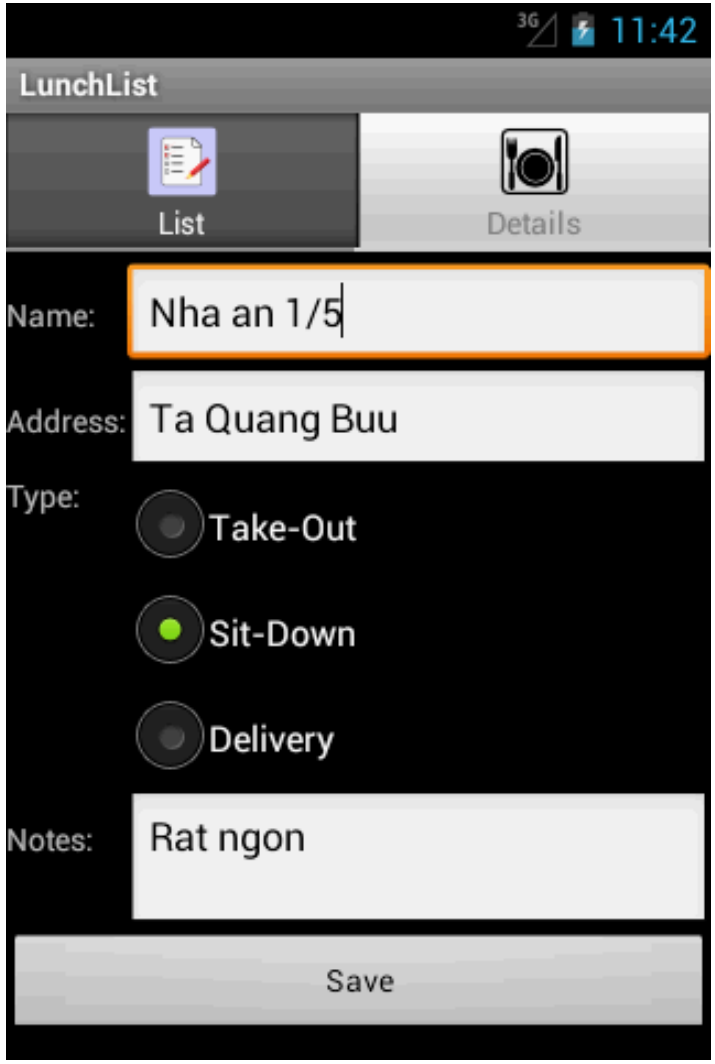
- Android will destroy and re-create any running or paused activities the next time they are to be viewed.

Default behavior

- Android will automatically update widgets
- What Android cannot automatically help you with is anything held outside the widgets.

Default behavior

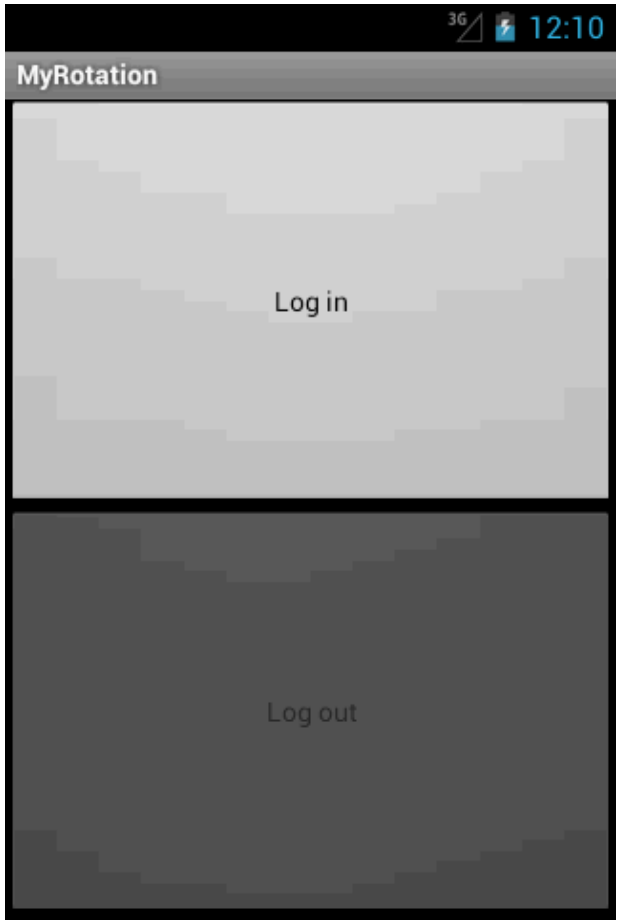


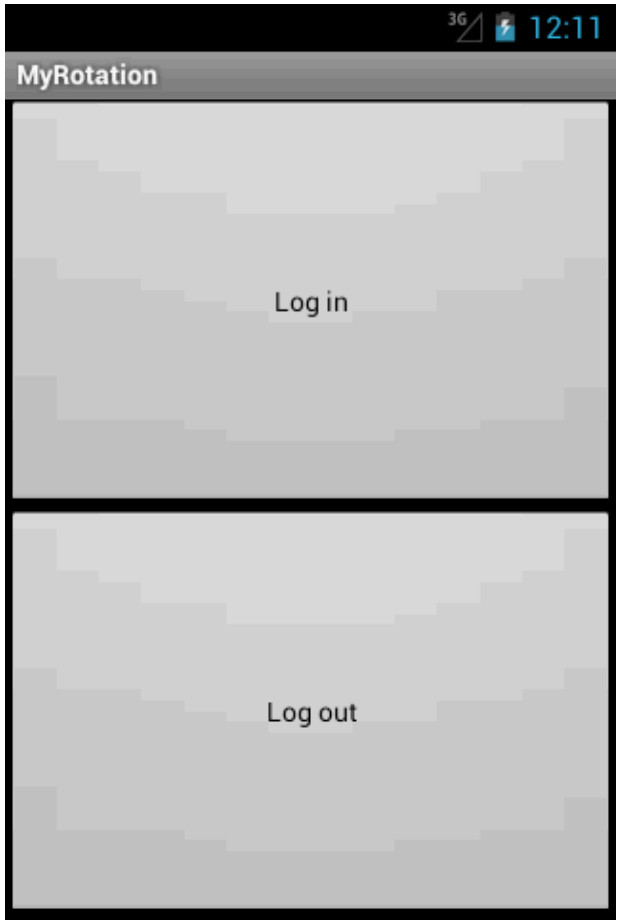


XML Layout

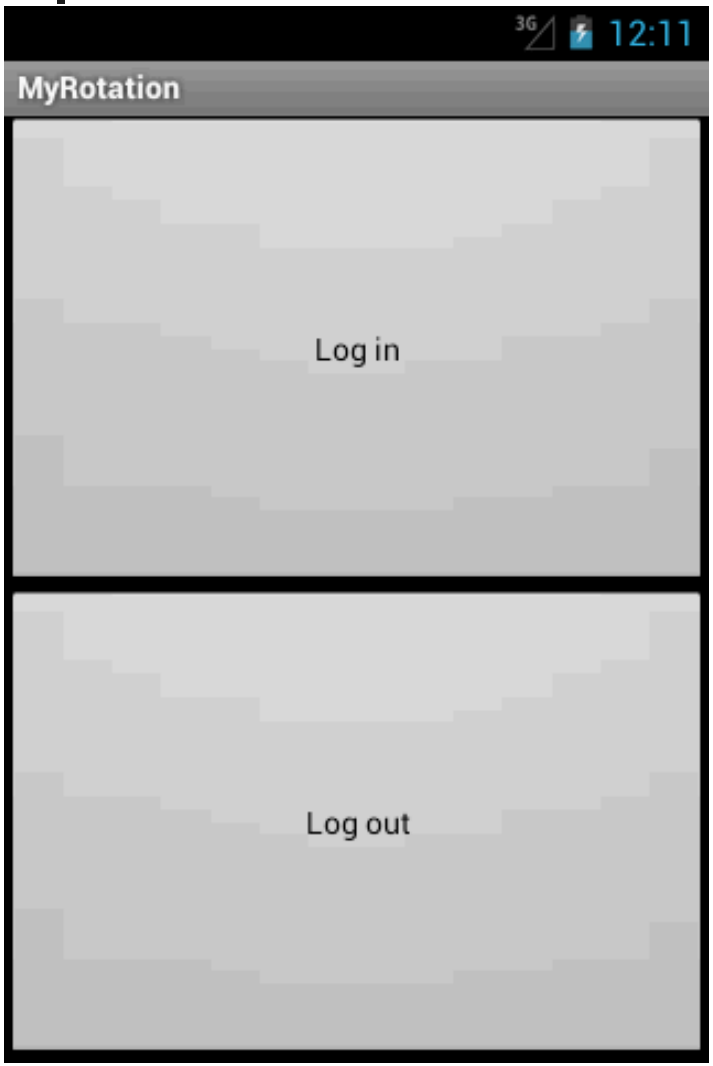
- Tạo file giao diện có cùng tên (main.xml) trong hai thư mục:
  - res/layout/
  - res/layout-land/

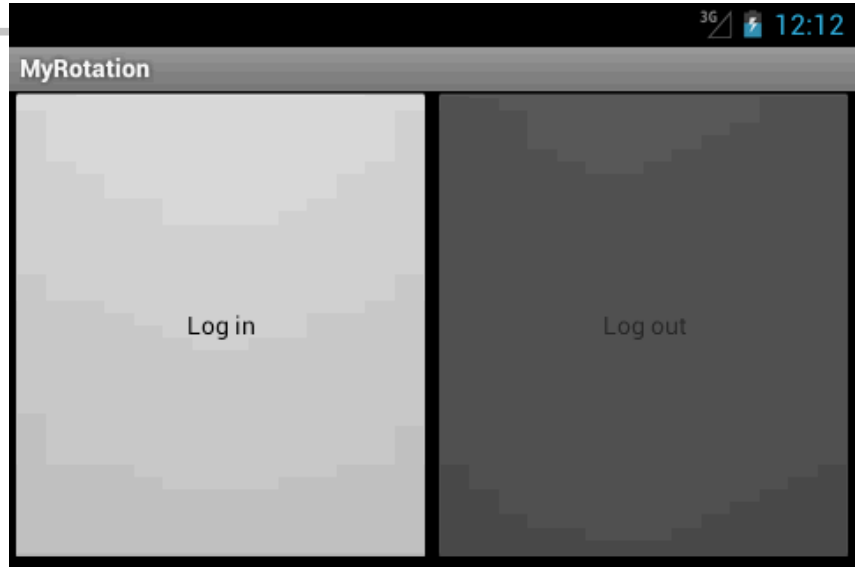
Ví dụ: tạo giao diện "Log In" MyRotation

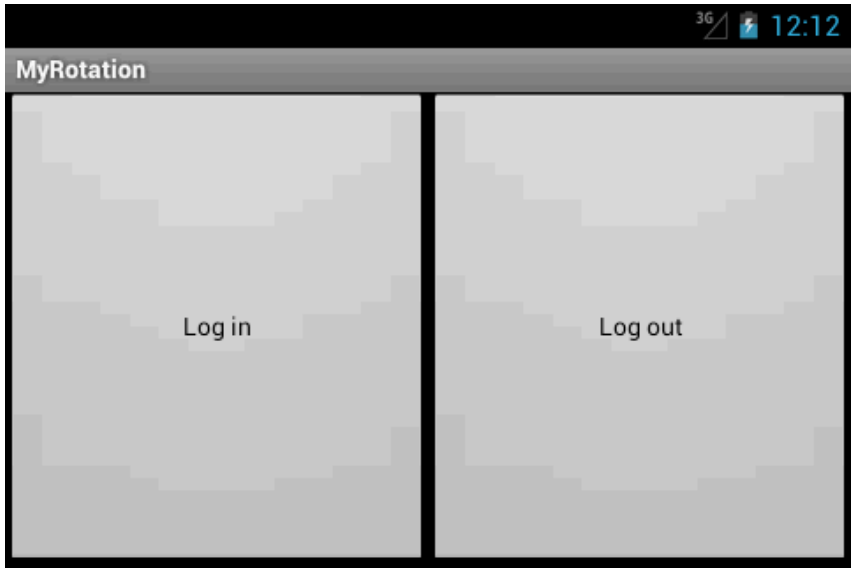




Ví dụ: tạo giao diện “Log In”







XML Layout

```
<LinearLayout>
    <Button android:id="@+id/login" />
    <Button android:id="@+id/logout" />
<LinearLayout>
```

Xử lý sự kiện với nút “Log In”

```
AtomicBoolean isLogin = new AtomicBoolean(false);

final View loginView = getLayoutInflater().inflate(R.layout.login, null);

    AlertDialog.Builder dialog = new AlertDialog.Builder(this);
    dialog.setTitle("Login");
    dialog.setView(loginView);
    dialog.setNegativeButton("Cancel", null);
    dialog.setPositiveButton("OK", …);
    dialog.show();
```

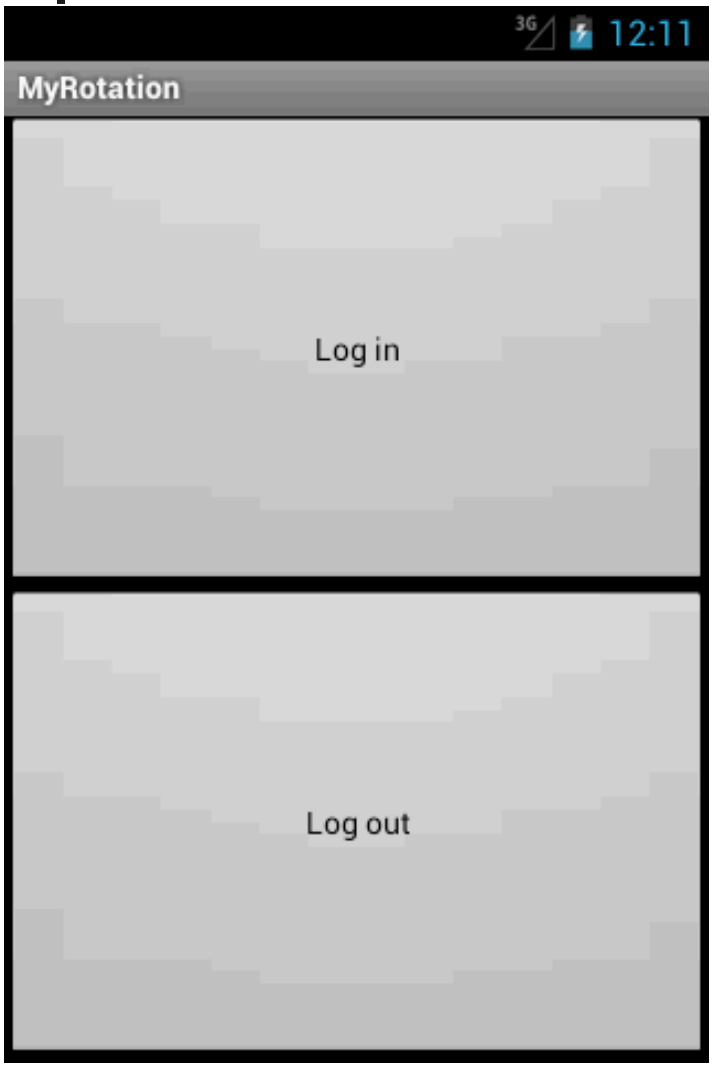
Xử lý sự kiện với nút "Log In"

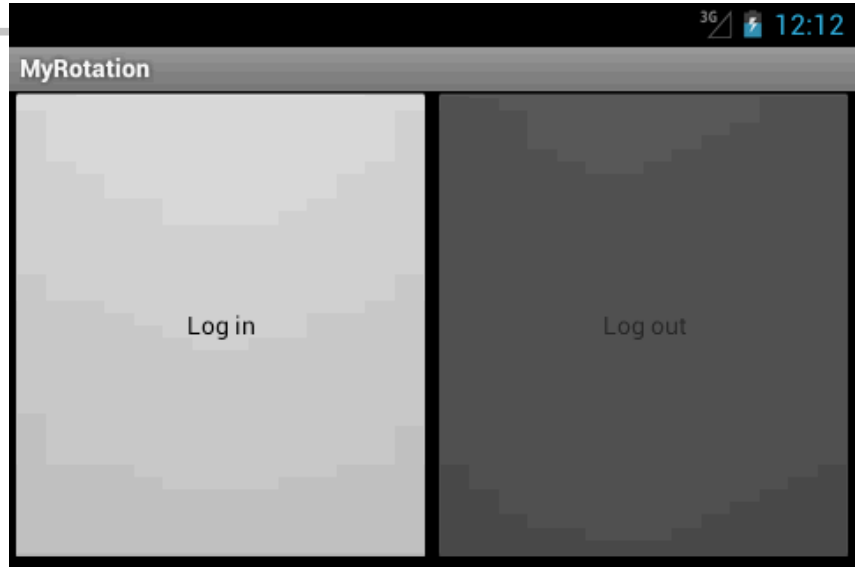
```
new DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog, int which) {
        EditText
          username=(EditText)loginView.findViewById(R.id.username);
        EditText password=(EditText)loginView.findViewById(R.id.password);
        Button btLogOut = (Button)findViewById(R.id.logout);
        String strUsername = username.getText().toString();
        String strPassword = password.getText().toString();
        if ((strUsername.equals("q")) && (strPassword.equals("1"))) {
            isLogin.set(true);
            btLogOut.setEnabled(true);
        }
    }
}
```

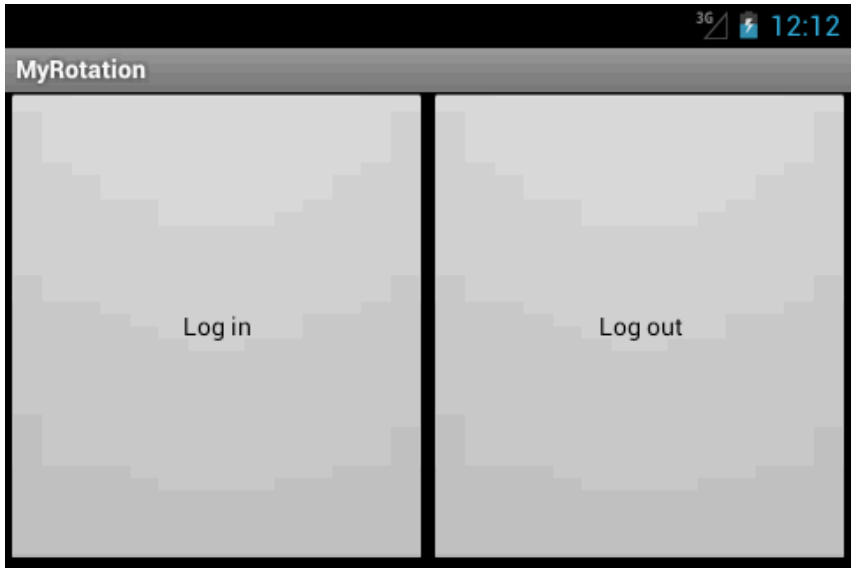
Xử lý sự kiện với nút "Log Out"

```
public void logOut(View v) {
    isLogin.set(false);
    Button btLogOut =
      (Button)findViewById(R.id.logout);
    btLogOut.setEnabled(false);
}
```

Ví dụ: tạo giao diện “Log In”







Lưu trạng thái của biến kiểm tra - onSaveInstanceState

```
@Override
protected void onSaveInstanceState(Bundle
   outState) {
   super.onSaveInstanceState(outState);
   outState.putString("login", isLogin.toString());
}
```

Khôi phục trạng thái của biến kiểm tra - savedInstanceState

```
if (savedInstanceState != null) {
        String strLogin =
                savedInstanceState.getString("login");
        Log.w("MyLog", strLogin);
        if (strLogin.equals("true")) {
          isLogin.set(true);
          Button btLogOut = (Button)findViewById(R.id.logout);
          btLogOut.setEnabled(true);
        }  else
          isLogin.set(false);
}
```

onSaveInstanceState

- Chỉ lưu được dữ liệu ở dạng xâu ký tự
- Để lưu đối tượng:
  - Sử dụng onRetainNonConfigurationInstance

Giới hạn của onSaveInstanceState

- The problem with onSaveInstanceState() is that you are limited to a Bundle.
- That's because this callback is also used in cases where your whole process might be terminated (e.g., low memory), so the data to be saved has to be something that can be serialized and has no dependencies upon your running process.

Giới hạn của onSaveInstanceState

- For some activities, that limitation is not a problem. For others, though, it is more annoying.
- Take an online chat, for example.
  - You have no means of storing a socket in a Bundle, so by default, you will have to drop your connection to the chat server and re-establish it.
  - That not only may be a performance hit, but it might also affect the chat itself, such as you appearing in the chat logs as disconnecting and reconnecting.

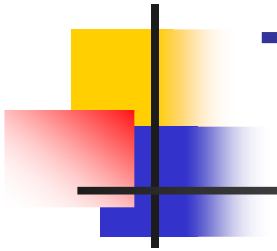
Lưu trạng thái đối tượng sử dụng onRetainNonConfigurationInstance

```
@Override
public Object onRetainNonConfigurationInstance() {
    return isLogin;
}
```

Khôi phục trạng thái đối tượng trong hàm onCreate()

```
if (getLastNonConfigurationInstance()!=null) {
        isLogin =
          (AtomicBoolean)getLastNonConfigurationInstance();
        if (isLogin.get()) {
          Button btLogOut = (Button)findViewById(R.id.logout);
          btLogOut.setEnabled(true);
        }
}
```

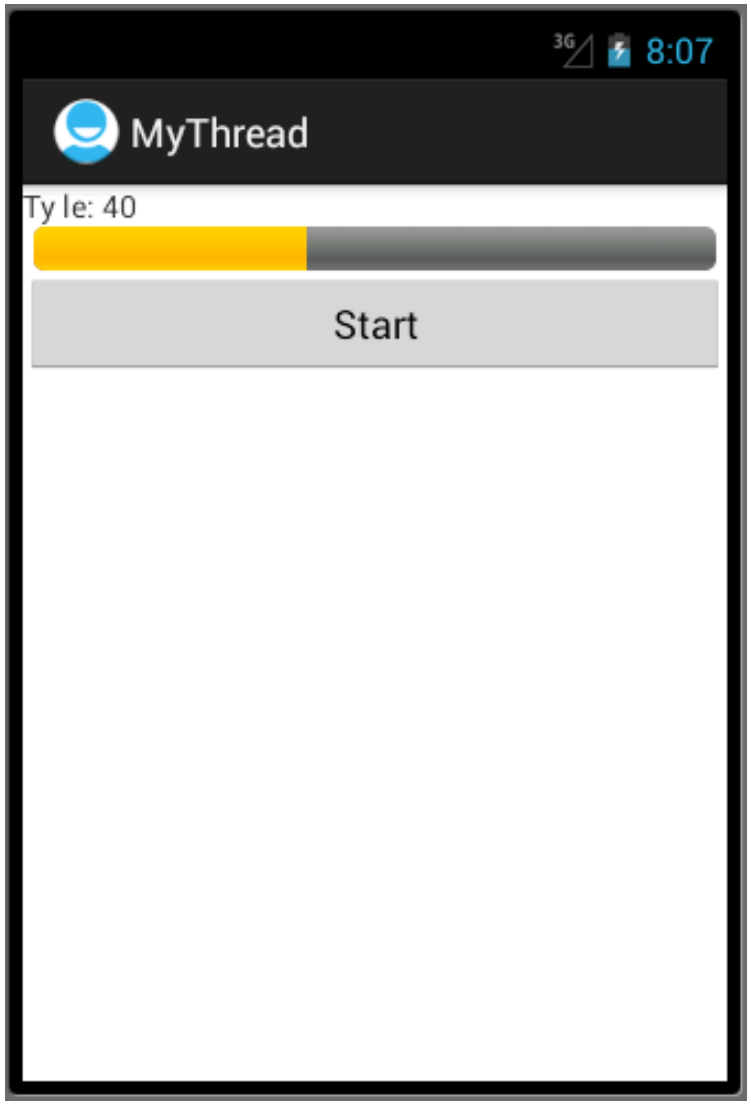
Threads and Rotation

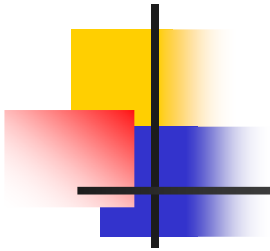


Threads and Rotation

- If the activity has started some background work – through an AsyncTask, for example – and then the activity is destroyed and re-created, somehow the AsyncTask needs to know about this.
- Otherwise, the AsyncTask might well send updates and final results to the *old activity, with the new activity none* the wiser.
- In fact, the new activity might start up the background work *again, wasting resources.*

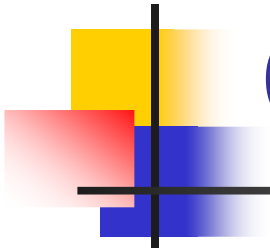
Example





Example

- Uses a ProgressBar
- It also has a TextView to indicate when the background work is completed, initially invisible
- an AsyncTask to do some (fake) work in the background, updating the ProgressBar along the way, and making the TextView visible when it is finished.
- More importantly, it needs to do this in such a way as to behave properly if the screen is rotated:

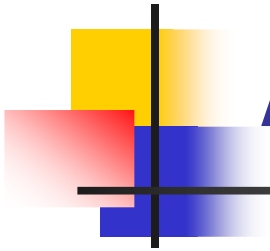


Comment

- We cannot "lose" our AsyncTask, having it continue doing work and updating the wrong activity
- We cannot start a second AsyncTask, thereby doubling our workload
- We need to have the UI correctly reflect our work's progress or completion

Example
XML Layout

```
<LinearLayout>
    <ProgressBar android:id="@+id/progress”
      style="?android:attr/progressBarStyleHorizontal“
      >
    <TextView android:id="@+id/completed“
      android:text="Work completed!"
      android:visibility="invisible“ />
</LinearLayout>
```



AsyncTask – inner class

- Having the AsyncTask as an inner class of the Activity means you get ready access to the Activity for any place where you need a Context.
- However, for the rotation scenario, the AsyncTask will think it knows the Activity it is supposed to work with, but in reality it will be holding onto an implicit reference to the old activity, not one after an orientation change.

AsyncTask – static inner class

- So, our ASyncTask class is a static inner class.
- This means RotationAwareTask does not have any implicit reference to any RotationAsync Activity (old or new)

AsyncTask doInBackground

```
static class RotationAwareTask extends AsyncTask<Void, Void, Void> {
    RotationAsync activity=null;
    int progress=0;
    RotationAwareTask(RotationAsync activity) {
        attach(activity);
    }
    @Override
    protected Void doInBackground(Void... unused) {
        for (int i=0;i<20;i++) {
            SystemClock.sleep(500);
            publishProgress();
        }
        return(null);
    }
```

AsyncTask onProgressUpdate

```
@Override
protected void onProgressUpdate(Void... unused) {
    if (activity==null) {
        Log.w("RotationAsync", "onProgressUpdate() skipped – no
          activity");
    }
    else {
        progress+=5;
        activity.updateProgress(progress);
    }
}
```

AsyncTask
onPostExecute

```
@Override
protected void onPostExecute(Void unused) {
    if (activity==null) {
        Log.w("RotationAsync", "onPostExecute() skipped
        – no activity");
    }
    else {
        activity.markAsDone();
    }
}
```

AsyncTask
detach() and attach

```
void detach() {
   activity=null;
}
void attach(RotationAsync activity) {
   this.activity=activity;
}
```

onCreate

```
task=(RotationAwareTask)getLastNonConfigurationInstance();
if (task==null) {
    task=new RotationAwareTask(this);
    task.execute();
}
else {
    task.attach(this);
    updateProgress(task.getProgress());
    if (task.getProgress()>=100) {
        markAsDone();
    }
}
```

onRetainNonConfigurationInstance

```
@Override
public Object
   onRetainNonConfigurationInstance() {
   task.detach();
   return(task);
}
```

updateProgress

```
void updateProgress(int progress) {
    bar.setProgress(progress);
}
void markAsDone() {
    findViewById(R.id.completed).setVisibility(View.VIS
    IBLE);
}
```

End of Lecture

Q&A