Upcoming Assignments

• Quiz Friday?
• Lab 5 due today
• Alpha Version due Friday, February 26
  – Inject one subtle defect (fault seeding)
  – To be reviewed by a few class members
  – Usability study by CPE 484 students
Background Processes

• One of the key differences between Android and iPhone is the ability to run things in the background on Android
  – Threads
    • Run something in the background while user interacts with UI
  – Services
    • Regularly or continuously perform actions that don’t require a UI
Threads

• Recall that Android ensures responsive apps by enforcing a 5 second limit on Activities
• Sometimes we need to do things that take longer than 5 seconds, or that can be done while the user does something else
Threads

• Activities, Services, and Broadcast Receivers run on the main application thread
• We can start background/child threads to do things for us
private void methodInAndroidClass() {
    Thread thread = new Thread(null, doSomething, "Background");
    thread.start();
}
private Runnable doSomething = new Runnable() {
    public void run() { /* do the something here */ }
};
```java
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    new Thread() {
        public void run() {
            String maps = null;
            try {
                URL updateURL = new URL("http://simexusa.com/cm/mapdata.txt");
                URLConnection conn = updateURL.openConnection();
                int contentLength = conn.getContentLength();
                InputStream is = conn.getInputStream();
                BufferedInputStream bis = new BufferedInputStream(is, 1024);
                ByteArrayOutputStream baf = new ByteArrayOutputStream(contentLength);
                int current = 0;
                while ((current = bis.read()) != -1) {
                    baf.append((byte) current);
                }
                maps = new String(baf.toByteArray()); // Convert the Bytes read to a String.
            } catch (Exception e) {
            }
        }
    }.start();
}
```

Example

Get data from web in background

Now we want to display data on screen
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    TextView hello = (TextView) findViewById(R.id.hellotv);
    hello.setText("testing");
    new Thread() {
        public void run() {
            String maps = null;
            try {
                URL updateURL = new URL("http://simexusa.com/cm/mapdata.txt");
                URLConnection conn = updateURL.openConnection();
                int contentLength = conn.getContentLength();
                InputStream is = conn.getInputStream();
                BufferedInputStream bis = new BufferedInputStream(is);
                ByteArrayOutputStream baf = new ByteArrayOutputStream(contentLength);
                int current = 0;
                while((current = bis.read()) != -1){
                    baf.append((byte)current);
                }
                maps = new String(baf.toByteArray()); //Convert the Bytes read to a String.
            } catch (Exception e) {} }
            TextView hello = (TextView) findViewById(R.id.hellotv);
            hello.setText(maps);
        }.start();
    }
}

CalledFromWrongThreadException: Only the original thread that created a view hierarchy can touch its views.
Android Thread Constraints

• Child threads cannot access UI elements (views); these must be accessed through the main thread

• What to do?
  – Give results to main thread and let it use results

• In Campus Maps I set a flag in the thread, then I added the menu item dynamically in Activity.onPrepareOptionsMenu
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    final TextView hello = (TextView) this.findViewById(R.id.hello_tv);
    hello.setText("testing");
    new Thread() {
        public void run() {
            String maps = null;
            try {
                URL updateURL = new URL("http://simexusa.com/cm/mapdata.txt");
                URLConnection conn = updateURL.openConnection();
                int contentLength = conn.getContentLength();
                InputStream is = conn.getInputStream();
                BufferedReader bis = new BufferedReader(new InputStreamReader(is, 1024));
                ByteArrayOutputStream baf = new ByteArrayOutputStream(contentLength);
                int current = 0;
                while ((current = bis.read()) != -1) {
                    baf.append((byte) current);
                }
                maps = new String(baf.toByteArray()); // Convert the Bytes read to a String.
            } catch (Exception e) {}  
            hello.setText(maps);
        }
    }.start();
}
Post to GUI Thread

- Handler allows you to post Messages and Runnable objects to threads
  - These can be scheduled to run at some point in the future, or enqueued for another thread
  - We will use the latter
public class BackgroundDemos extends Activity {
    private Handler handler = new Handler();
    private String maps = null;
    TextView hello;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        hello = (TextView)this.findViewById(R.id.hellotv);
        hello.setText("testing");
        new Thread() {
            public void run() {
                try {
                    URL updateURL = new URL("http://simexusa.com/cm/mapdata.txt");
                    //code omitted here
                    maps = new String(baf.toByteArray()); //Convert the Bytes read to a String.
                    handler.post(doUpdateMaps);
                } catch (Exception e) {} }
        }.start();

        private Runnable doUpdateMaps = new Runnable() {
            public void run() {
                hello.setText(maps);
            }
        };
    }
}
Services

• Services are like Activities, but without a UI
• Services are not intended as background threads
  – Think of a media player where the song keeps playing while the user looks for more songs to play or uses other apps
  – Don’t think of a cron job (e.g. run every day at 3am), use Alarms to do this
• Several changes in 2.0 related to Services
• Add to AndroidManifest.xml

<service android:enabled="true" android:name=".NewMapService"></service>

• Create the Service class

public class NewMapService extends Service {
    @Override
    public void onCreate() {
    }
    @Override
    public void onStart(Intent intent, int startId) {
        //do something
    }
    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }
}
Start/Stop the Service

- Other alternatives in text

```java
public class MyActivity extends Activity {
    @Override
    public void onCreate() {
        …
        startService(new Intent(this, NewMapService.class));
    }
    @Override
    public void onStop() {
        …
        stopService(new Intent(this, NewMapService.class));
    }
}
```
NotificationManager nm = (NotificationManager)getSystemService(Context.NOTIFICATION_SERVICE);

Notification notification = new Notification(R.drawable.icon, "NewMaps", System.currentTimeMillis());

String expandedTitle = "New Map Service";
String expandedText = "New Map Service is running";
Intent i = new Intent(this, NewMapService.class);
PendingIntent launchIntent = PendingIntent.getActivity(getApplicationContext(), 0, i, 0);

notification.setLatestEventInfo(getApplicationContext(), expandedTitle, expandedText, launchIntent);

nm.notify(1, notification);

nm.cancel(1); //cancel a notification (id/parameters must match)
Other Notification

• Sounds
• Vibration
• Flashing lights
• Ongoing and Insistent