Addressing Non-Functional Requirements in Mobile Apps

CSE 5236: Mobile Application Development
Instructor: Adam C. Champion, Ph.D.
Course Coordinator: Dr. Rajiv Ramnath
Outline

• Non-Functional Requirements
• Optimize Performance with Profiler
• Maximize Battery Life
• Optimize for Responsiveness
• Improve App Security
• Testing
Non-Functional Requirements (NFRs)

• AKA quality/design requirements: Building the app right (as opposed to the “right app” w.r.t. functional requirements)

• Typical NFRs include:
  – Performance
  – Availability
  – Scalability
  – Usability
  – Security
  – Modifiability
  – Maintainability and testability
  – Cost

• Almost always entail tradeoffs; some aligned combinations (e.g., security and availability)
Key NFRs for Mobile Devices

• Performance
• Responsiveness (different from performance)
• Security
Systematic Steps Towards Meeting NFRs

• Quantify for the app (e.g., 60 frames/sec)
• Make appropriate architectural decisions: often pre-determined by the underlying architecture of the implementation framework (e.g., Android SDK)
• Optimize tactically using real measurements
Architectural Decisions in Tic-Tac-Toe

• Java/Kotlin – reduced cost of development
• Data storage tactics:
  – Preferences: cost of development
  – SQLite: Reliability, queries faster than inserts suited for login use case.
• Data transfer via JSON
• 2-D graphics for speed
Tactical Optimizations Used in Tic-Tac-Toe

• Used variables to cache data retrieved from collections (e.g. arrays)
• Avoided internal use of getters and setters
• Reduced heap access: avoid creating unnecessary objects (see use of Singleton for X, O and Blank symbols)
• Used static final for constants (allows inlining of constants)
• Leveraged optimizations in framework libraries
Outline

- Non-Functional Requirements
- Optimize Performance with Profiler
- Maximize Battery Life
- Optimize for Responsiveness
- Improve App Security
- Testing
Optimize Performance with Profiler (1)

1. Connect Android device to dev machine
2. Click “Android Profiler” icon;
3. App starts running on device
4. Profile CPU use, memory use, etc.
1. Generate a method trace by pressing Record button.
2. Use the app “as normal”.
Method Trace View

- Method trace window appears.
- Find slow parts of program, investigate...
onDraw(), getBitmapForSymbol()

App is using 11.71% of CPU 😞

These methods are using CPU heavily here...
// Board.java
public void onDraw() {
    ...
    for (int i = 0; i < GameGrid.SIZE; i++) {
        for (int j = 0; j < GameGrid.SIZE; j++) {
            Bitmap symSelected = getBitmapForSymbol(grid.getValueAtLocation(i, j));
            offsetX = (int)(((width - symSelected.getWidth())/2) + (i * width));
            offsetY = (int)(((height - symSelected.getHeight())/2) + (j * height));
            canvas.drawBitmap(symSelected, offsetX, offsetY, ditherPaint);
        }
    }
    ...
}

// Only considering Java here. Kotlin optimization is similar.
public Bitmap getBitmapForSymbol(Symbol aSymbol) {
    try {
        Resources res = getResources();
        sSymX = BitmapFactory.decodeResource(res, R.drawable.x);
        sSymO = BitmapFactory.decodeResource(res, R.drawable.o);
        sSymBlank = BitmapFactory.decodeResource(res, R.drawable.blank);
    } catch (OutOfMemoryError ome) {
        Bitmap symSelected = sSymBlank;
        if (aSymbol == Symbol.SymbolXCreate())
            symSelected = sSymX;
        else if (aSymbol == Symbol.SymbolOCreate())
            symSelected = sSymO;
        return symSelected;
    }
}
static Bitmap symX = null, symO = null, symBlank = null;
static boolean sDrawablesInitialized = false;

public Bitmap getBitmapForSymbol(Symbol aSymbol) {
    if (!sDrawablesInitialized) {
        Resources res = getResources();
        symX = BitmapFactory.decodeResource(res, R.drawable.x);
        symO = BitmapFactory.decodeResource(res, R.drawable.o);
        symBlank = BitmapFactory.decodeResource(res, R.drawable.blank);
        sDrawablesInitialized = true;
    }
    Bitmap symSelected = symBlank;
    if (aSymbol == Symbol.SymbolXCreate())
        symSelected = symX;
    else if (aSymbol == Symbol.SymbolOCreate())
        symSelected = symO;
    return symSelected;
}
After Optimization
Outline

• Non-Functional Requirements
• Optimize Performance with Profiler
• **Maximize Battery Life**
• Optimize for Responsiveness
• Improve App Security
• Testing
Maximize Battery Life

• Reducing computation (same techniques as for performance)
• Reducing network usage
  – Minimizing data services
  – Minimizing location services
• Managing display brightness
Minimize Network Use: Java

• Check for network availability
  ```java
  private boolean hasNetworkConnection() {
    ConnectivityManager connectivityManager =
        (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);
    NetworkInfo networkInfo =
        connectivityManager.getNetworkInfo(ConnectivityManager.TYPE_WIFI);
    boolean isConnected = true;
    boolean isWifiAvailable = networkInfo.isAvailable();
    boolean isWifiConnected = networkInfo.isConnected();
    networkInfo =
        connectivityManager.getNetworkInfo(ConnectivityManager.TYPE_MOBILE);
    boolean isMobileAvailable = networkInfo.isAvailable();
    boolean isMobileConnected = networkInfo.isConnected();
    isConnected = (isMobileAvailable&&isMobileConnected) ||
                  (isWifiAvailable&&isWifiConnected);
    return(isConnected);
  }
  ```

• Use compact data formats (JSON)
Minimize Network Use: Kotlin

• Checking for network availability:
  ```kotlin
  private fun hasNetworkConnection(): Boolean {
    val connectivityManager = 
        activity.applicationContext.getSystemService(
            Context.CONNECTIVITY_SERVICE) as ConnectivityManager
    var networkInfo = connectivityManager
        .getNetworkInfo(ConnectivityManager.TYPE_WIFI)
    var isConnected = true
    val isWifiAvailable = networkInfo.isAvailable
    val isWifiConnected = networkInfo.isConnected
    networkInfo = connectivityManager
        .getNetworkInfo(ConnectivityManager.TYPE_MOBILE)
    val isMobileAvailable = networkInfo.isAvailable
    val isMobileConnected = networkInfo.isConnected
    isConnected = (isMobileAvailable && isMobileConnected) 
        || (isWifiAvailable && isWifiConnected)
    return isConnected
  }
  ```
Minimize Location Services: Preconditions: Java

```java
public class MapsActivity extends SingleFragmentActivity { // . . .
    @Override
    protected Fragment createFragment() { return new MapsFragment(); }
    // . . .
}

public class MapsFragment extends SupportMapFragment implements OnMapReadyCallback {
    private GoogleApiClient mApiClient;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        // Call super.onCreate(), . . .
        mApiClient = new GoogleApiClient.Builder(getActivity()).
                       .addApi(LocationServices.API).build(); // Add ConnectionCallbacks code here
    }
    @Override
    public void onStart() {
        // Call super.onStart(), . . .
        mApiClient.connect();
    }
    @Override
    public void onStop() {
        // Call super.onStop(), . . .
        mApiClient.disconnect();
    }
}
```
Minimize Location Services: Preconditions: Kotlin

class MapsActivity : SingleFragmentActivity() { // . . .
    override fun createFragment(): Fragment { return MapsFragment() } // . . .
}

class MapsFragment : SupportMapFragment(), OnMapReadyCallback {
    private lateinit var mApiClient: GoogleApiClient
    override fun onCreate(savedInstanceState: Bundle?) {
        // . . .
        mApiClient = GoogleApiClient.Builder(activity)
        .addApi(LocationServices.API).build()
        // Add ConnectionCallbacks code here
    }

    override fun onStart() {
        // Call super.onStart(), . . .
        mApiClient.connect()
    }

    override fun onStop() {
        // Call super.onStop(), . . .
        mApiClient.disconnect()
    }
}
Minimize Location Services: Use Last Known Location

Java

```java
// MapsFragment.java
// ...
@Override
public void onConnected(Bundle connectionHint) {
    Location location = LocationServices.FusedLocationApi.getLastLocation(mApiClient);
    if (location != null) {
        mLatitudeText.setText(String.valueOf(location.getLatitude()));
        mLongitudeText.setText(String.valueOf(location.getLongitude()));
    }
}
```

Kotlin

```kotlin
// MapsFragment.kt
// ...
override fun onConnected(connectionHint: Bundle?) {
    Location location = LocationServices.FusedLocationApi.getLastLocation(mApiClient)
    if (location != null) {
        mLatitudeText.setText(location.getLatitude().toString());
        mLongitudeText.setText(location.getLongitude().toString());
    }
}
```
## Minimize Location Services: Location-Request Priorities, Tradeoffs

<table>
<thead>
<tr>
<th>LocationRequest Priority</th>
<th>Technology</th>
<th>Error (m)</th>
<th>Energy Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRIORITY_BALANCED_POWER_ACCURACY</td>
<td>WiFi, cellular</td>
<td>~100 (city block)</td>
<td>Moderate</td>
</tr>
<tr>
<td>PRIORITY_HIGH_ACCURACY</td>
<td>GPS</td>
<td>~10</td>
<td>High</td>
</tr>
<tr>
<td>PRIORITY_LOW_POWER</td>
<td>WiFi, cellular</td>
<td>~10,000 (city)</td>
<td>Low</td>
</tr>
<tr>
<td>PRIORITY_NO_POWER</td>
<td>Varies</td>
<td>Varies</td>
<td>Zero*</td>
</tr>
</tbody>
</table>

* Relies on other apps to get location estimates and uses these estimates.

Update intervals for `LocationRequest`s can be set too.

Outline

• Non-Functional Requirements
• Optimize Performance with Profiler
• Maximize Battery Life
• Optimize for Responsiveness
• Improve App Security
• Testing
Responsiveness: Threading: Ex. (1)

Java

public void onClick(View v) {
    new Thread(new Runnable() {
        public void run() {
            Bitmap b = loadImageFromNetwork();
            // User-written method
            // Do something with the image...
        }
    }).start();
}

Kotlin

override fun onClick(v: View) {
    Thread({
        val b: Bitmap = loadImageFromNetwork()
        // User-written method
        // Do something with the image
    }).start();
}

Note: passing anonymous instance of Runnable to Thread’s constructor.
Threading: Ex. (2): SplashScreen: Java

// SplashScreenFragment.java

@Override
public void onStart() { // . . .
    // Thread for displaying the SplashScreen
    Thread splashThread = new Thread() {
        @Override
        public void run() {
        try {
            int elapsedTime = 0;
            while (mIsActive && (elapsedTime < mSplashTime)) {
                sleep(mSleepTime);
                if (mIsActive) { elapsedTime = elapsedTime + mTimeIncrement; } }
        } catch (InterruptedException e) { // do nothing
        } finally {
            getActivity().finish();
            startActivity(new Intent("com.wiley.fordummies.androidsdk.tictactoe.Login"));  }
    }
    splashThread.start();

    @Override
    public boolean onTouch(View view, MotionEvent motionEvent) {
        if (motionEvent.getAction() == MotionEvent.ACTION_DOWN) {mIsActive = false;return true;}
    }
Threading: Ex. (2): SplashScreen: Kotlin

// SplashScreenFragment.kt
override fun onStart() { // . . .
    // Thread for displaying the SplashScreen
    val splashThread = Thread {
        try {
            var elapsedTime = 0
            while (mIsActive && elapsedTime < mSplashTime) {
                Thread.sleep(mSleepTime.toLong())
                if (mIsActive) { elapsedTime = elapsedTime + mTimeIncrement }
            }
        } catch (e: InterruptedException) { // do nothing
        } finally {
            activity.finish()
            startActivity(Intent("com.wiley.fordummies.androidsdk.tictactoe.Login"))
        }
    }
    splashThread.start()
}

override fun onTouch(view: View, motionEvent: MotionEvent): Boolean {
    if (motionEvent.action == MotionEvent.ACTION_DOWN) {
        mIsActive = false
        return true
    }
}
Threading: Ex. (3): Machine Play

Java

// GameSessionFragment.java
public void scheduleAndroidsTurn() { // ... 
mBoard.disableInput();
if (!mTestMode) {
    Random randomNumber = new Random();
    Handler handler = new Handler();
    handler.postDelayed(
        new Runnable() {
            public void run() {
                androidTakesATurn();
            }
        }, ANDROID_TIMEOUT_BASE +
        randomNumber.nextInt(
            ANDROID_TIMEOUT_SEED)
    );
} else {
    androidTakesATurn();
}
}

Kotlin

// GameSessionFragment.kt
fun scheduleAndroidsTurn() { // . . .
mBoard.disableInput()
if (!mTestMode) {
    val randomNumber = Random()
    val handler = Handler()
    handler.postDelayed(
        { androidTakesATurn() },
        (ANDROID_TIMEOUT_BASE +
        randomNumber.nextInt(
            ANDROID_TIMEOUT_SEED)).toLong()
    )
} else {
    androidTakesATurn()
}
}

Anonymous instance of Runnable
Threading: Ex. (4): Framework-Managed Threads: Java

// HelpWebViewFragment.java

public View onCreateView(LayoutInflater inflater, ViewGroup container,
    Bundle savedInstanceState) {
    View v = inflater.inflate(R.layout.fragment_help_webview, container, false);

    WebView helpInWebView = (WebView) v.findViewById(R.id.helpwithwebview);
    mProgressBar = (ProgressBar) v.findViewById(R.id.webviewprogress);
    mProgressBar.setMax(100);

    Bundle extras = getActivity().getIntent().getExtras();
    if (extras != null) {
        mUrl = extras.getString(ARG_URI); // ...
    } // More code here

    helpInWebView.loadUrl(mUrl); // Loads in separate thread

    return v;
}
Threading: Ex. (4): Framework-Managed Threads: Kotlin

```kotlin
override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?, savedInstanceState: Bundle?): View? {
    val v = inflater.inflate(R.layout.fragment_help_webview, container, false)
    val helpInWebView = v.findViewById<WebView>(R.id.helpwithwebview)
    mProgressBar = v.findViewById<ProgressBar>(R.id.webviewprogress)
    mProgressBar.apply { max = 100 }

    val extras = activity.intentextras
    if (extras != null) {
        mUrl = extras.getString(ARG_URI) // . . .
    }

    helpInWebView.loadUrl(mUrl) // Loads in separate thread

    return v
}
```
The Android Thread Model

- Main thread *usually* the UI thread (except when testing)
- SDK is NOT thread-safe: Other threads should NOT manipulate UI (just compute, give result to UI thread)
- API to access UI thread:
  - `Activity.runOnUiThread(Runnable myRunnable)` runs specified Runnable object on UI thread (See `GameSessionTest.java`)
  - `View.post(Runnable myRunnable)` adds Runnable to message queue to be run by UI thread
  - `View.postDelayed(Runnable, long)` adds Runnable to message queue after specified period of time
  - `Handler` class lets you run preceding `post()`, `postDelayed(...)` operations when you cannot access an active `View. (see `GameSessionFragment.java`)`
- UI thread subordinated to unit test thread (see section on *Testing*)
Outline

• Non-Functional Requirements
• Optimize Performance with Profiler
• Maximize Battery Life
• Optimize for Responsiveness
• Improve App Security
• Testing
Security Considerations for Mobile Devices

- Devices store valuable personal information
- Larger security “footprint”, more attack surfaces \( \Rightarrow \) more vulnerabilities
  - Existing threats magnified (e.g. poorly secured browsers, mobile web sites)
  - Installed apps sources of insecurity (more apps, hard to trust authors in open market)
  - Sharing between apps.
  - Private data left behind on file system (less on Android 10+)
- Device is inherently less secure
  - Portable, easily stolen
  - Assumption: one user (Typically weaker passwords used (due to difficulty of data entry)
  - Limited screen size, ambient distractions \( \Rightarrow \) users ignore security
- Lesson: App developers share responsibility for security
Systematic Steps to App Security

• **Don’t** randomly implement “security stuff”. Instead, define threat model:
  – What are your assets (data)? What is their value?
  – What attacks can occur (theft, DoS)? Where can they originate (network, apps)?
• Identify security tactics:
  – *Detection*: Determining that attack is in progress (or loss has occurred)
  – *Resistance*: Making loss more difficult to occur.
  – *Mitigation*: Limiting degree of loss/breach.
  – *Recovery*: Restore app/OS to “known good state”
• Implement tactics using security techniques:
  – Authentication (e.g. two-factor, certificates)
  – Access control (e.g. file ownership, encryption, certificates)
  – Audit trail (e.g. logs)
  – Data integrity (e.g. checksums, encryption)
  – Non-repudiation (e.g. logs, certificates)
Android Security Considerations

• **Good: “Privilege-supported” OS**
  – Processes “sandboxed” in user space
  – User files and databases are removed on uninstallation
  – Apps must request and be granted permissions (install, run time): to system resources, content providers, resources of other apps
  – Apps must be “signed” by developer (however, self-signing allowed!)
  – Google verifies new apps installed in Android 4.3+

• **Bad:**
  – **No security through obscurity**: Linux is open-source, APK files can be decompiled
  – Limited vetting process on Google Play (tests apps via QEMU system emulator*)
  – Privileges enforced by *installer* (hacked phones’ run-times may not enforce privileges)

• **Things to watch out for:**
  – Leaving private data in files on device, SD card (external memory)
  – Database hacking techniques (SQL injection)
  – Your app being the Trojan horse
  – Secret literals left in code (e.g. special passwords)
  – Using reversible security algorithms

* [https://jon.oberheide.org/files/summercon12-bouncer.pdf](https://jon.oberheide.org/files/summercon12-bouncer.pdf)
Examples of Permission Requests

• `<uses-permission android:name="android.permission.READ_CONTACTS"/>
• `<uses-permission android:name="android.permission.INTERNET"/>
• `<uses-permission
  android:name="android.permission.ACCESS_NETWORK_STATE"/>
• `<uses-permission
  android:name="android.permission.ACCESS_COARSE_LOCATION”/>
• `<uses-permission
  android:name="android.permission.ACCESS_FINE_LOCATION”/>
• `<uses-permission android:name=
  "com.wiley.fordummies.androidsdk.tictactoe.LAUNCHACTIVITY"/>

  Example of a custom permission

Note: Permission elements must be outside the `<application>` block and inside the `<manifest>` block of the AndroidManifest.xml
Custom Permissions: Definition and Placement

Permission must be declared:

```xml
<permission
    android:name =
    "com.wiley.fordummies.androidsdk.tictactoe.LAUNCHACTIVITY"
    android:label="Launch Tic-Tac-Toe Activity"
    android:description="@string/permission_launch_activity"
    android:protectionLevel="normal" />
```

Place in `AndroidManifest.xml` file outside the `<application>` block, inside the `<manifest>` block (same as `<uses-permission>` elements).
Custom Permissions: Declaring Need, Request

• Declare need via `android:permission` attribute in `activity` definition in manifest file:

```xml
<activity
    android:name=".Login"
    android:label="@string/app_name"
    android:launchMode="standard"
    android:screenOrientation="portrait"
    android:permission="...LAUNCHACTIVITY"/>
```

• Request: `<uses-permission android:name="...LAUNCHACTIVITY"/>`

• Requested in any separate package, containing package
Permission Checking in Android

• When a call is made to a system function: To prevent an unauthorized invocation
• When starting an Activity: To prevent an unauthorized application from launching the Activity of other applications
• When sending or receiving Broadcasts: To determine who can receive a Broadcast or send it to you
• When accessing, and operating on, a Content Provider: To prevent an unauthorized app from accessing the data in the Content Provider
• When binding to, or starting, a Service: To prevent an unauthorized application from using the Service.
Example logcat Entries During Permission Failures

02-28 12:48:00.864: ERROR/AndroidRuntime(378):
    java.lang.SecurityException: Permission Denial: starting Intent {
        act=com.wiley.fordummies.androidsdk.tictactoe.Login
        cmp=com.wiley.fordummies.androidsdk.tictactoe/.Login
    } from ProcessRecord{407740c0
        378:com.wiley.fordummies.androidsdk.tictactoe/10033} (pid=378, uid=10033)
    requires
        com.wiley.fordummies.androidsdk.tictactoe.permission.LAUNCHACTIVITY

02-28 21:04:39.758: ERROR/AndroidRuntime(914): at
    com.wiley.fordummies.androidsdk.tictactoe.SplashScreen$1.run
    (SplashScreen.java:36)
Example logcat Entries for Permission Definition or Placement Errors

02-28 16:53:09.838: DEBUG/PackageManager(77):   Permissions:
   com.wiley.fordummies.androidsdk.tictactoe.LAUNCHACTIVITY

02-28 17:04:18.888: WARN/PackageParser(77): Unknown element under <application>:

   permission at /data/app/vmdl1654102309.tmp Binary XML file line #11

02-28 17:04:20.438: WARN/PackageManager(77): Unknown permission
   com.wiley.fordummies.androidsdk.tictactoe.LAUNCHACTIVITY in package
   com.wiley.fordummies.androidsdk.tictactoe
Runtime Permission Checks (Android 6+)

- Certain permissions require explicit user authorization at runtime:

<table>
<thead>
<tr>
<th>Permission Group</th>
<th>Permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>CALENDAR</td>
<td>READCALENDAR, WRITECALENDAR</td>
</tr>
<tr>
<td>CAMERA</td>
<td>CAMERA</td>
</tr>
<tr>
<td>CONTACTS</td>
<td>READCONTACTS, WRITECONTACTS, GETACCOUNTS</td>
</tr>
<tr>
<td>LOCATION</td>
<td>ACCESSFINELOCATION, ACCESSCOARSELOCATION</td>
</tr>
<tr>
<td>MICROPHONE</td>
<td>RECORD_AUDIO</td>
</tr>
<tr>
<td>PHONE</td>
<td>READPHONESTATE, CALLPHONE, READCALLLOG,</td>
</tr>
<tr>
<td></td>
<td>WRITECALLLOG, ADDVOICEMAIL, USESIP,</td>
</tr>
<tr>
<td></td>
<td>PROCESSOUTGOINGCALLS</td>
</tr>
<tr>
<td>SENSORS</td>
<td>BODY_SENSORS</td>
</tr>
<tr>
<td>SMS</td>
<td>SENDSMS, RECEIVESMS, READSMS, RECEIVEWAP_PUSH,</td>
</tr>
<tr>
<td></td>
<td>RECEIVEMMS</td>
</tr>
<tr>
<td>STORAGE</td>
<td>READEXTERNALSTORAGE, WRITEEXTERNALSTORAGE</td>
</tr>
</tbody>
</table>

Runtime Permission Checks: Java

// ContactsFragment.java

@override
public void onActivityCreated(Bundle savedInstanceState) { /* ... */ requestContacts(); }

private void requestContacts() {
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
        if (!hasReadContactPermission()) {
            // Request permission if we don’t have it
            requestPermissions(new String[]{Manifest.permission.READ_CONTACTS},
                                PERMISSION_REQUEST_READ_CONTACTS);
        } else { showContacts(); }
    } else { showContacts(); }
}

@RequiresApi(api = Build.VERSION_CODES.M)
private boolean hasReadContactPermission() { // Check if we have perm. to read contacts
    return getActivity().checkSelfPermission(Manifest.permission.READ_CONTACTS)
               == PackageManager.PERMISSION_GRANTED;
}

@Override
public void onRequestPermissionsResult(/* ... */) {
    if (requestCode == PERMISSION_REQUEST_READ_CONTACTS) {
        // Callback: permission granted
        if (grantResults[0] == PackageManager.PERMISSION_GRANTED) { showContacts(); }
    } else { /* Callback: Permission denied */ }
}
// ContactsFragment.kt
override fun onActivityCreated(savedInstanceState: Bundle?) {
    /* ... */
    requestContacts()
}

private fun requestContacts() {
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
        if (!hasReadContactPermission()) {
            // Request permission if we don’t have it
            requestPermissions(arrayOf(Manifest.permission.READ_CONTACTS),
                PERMISSION_REQUEST_READ_CONTACTS)
        } else {
            showContacts()
        }
    } else {
        showContacts()
    }
}

@RequiresApi(api = Build.VERSION_CODES.M)
private fun hasReadContactPermission(): Boolean {
    // Check if we have perm to read contacts
    return activity.checkSelfPermission(Manifest.permission.READ_CONTACTS) ==
        PackageManager.PERMISSION_GRANTED
}

override fun onRequestPermissionsResult(requestCode: Int, grantResults: IntArray) {
    if (requestCode == PERMISSION_REQUEST_READ_CONTACTS) {
        /* Callback: perm granted */
        if (grantResults[0] == PackageManager.PERMISSION_GRANTED) {
            showContacts()
        } else {
            /* Callback: permission denied */
        }
    }
}
SQLite Security: SQL Injection

• Entry field: Name: <Enter Name>

• Intended query:
  – SELECT e-mail FROM user_information WHERE NAME='Bob'

• Attacker enters string:
  – ‘Bob’; SELECT table_names FROM user_tables

• Query becomes:
  – SELECT e-mail FROM user_information WHERE name='Bob'; SELECT table_names FROM user_tables

• Attacker knows all the tables. Augh!
SQL Injection Solution: Bind Variables

// AccountDbSchema.java
public class AccountDbSchema {
    public static final class AccountsTable {
        public static final String NAME = "accounts";
        public static final class Cols { /* Name and password columns */ }
    }
}

// AccountSingleton.java
private static final String INSERT_STMT = "INSERT INTO " + AccountsTable.NAME + " (name, password) VALUES (?, ?)";

// Account model object includes name, password fields to insert into DB
public void addAccount(Account account) {
    ContentValues contentValues = getContentValues(account);
    mDatabase.beginTransaction();
    try {
        SQLiteStatement statement = mDatabase.compileStatement(INSERT_STMT);
        statement.bindString(1, account.getName());
        statement.bindString(2, account.getPassword());
        statement.executeInsert();
        mDatabase.setTransactionSuccessful();
    } finally {
        mDatabase.endTransaction();
    }
}

// ...
General Rule: Minimize App Vulnerabilities

- Don’t hardwire “secrets” in code
- Mask sensitive data entry (e.g. passwords)
- Encrypt sensitive files
- Don’t write unnecessary temporary files
- Use bind variables
- Ask for the least permissions
- Create checkpoints of app data
- Log data (encrypt your logs too!)
- Keep intent filters specific so Activities don’t respond to generic Intents
- Prompt user for permission to access sensitive data
Outline

• Non-Functional Requirements
• Optimize Performance with Profiler
• Maximize Battery Life
• Optimize for Responsiveness
• Improve App Security
• Testing
Creating Unit Tests (1)

• In Android Studio, right-click the project name, select app, click on Dependencies tab, click “+” icon, select “Library Dependency”, then type “junit” into the dialog (if JUnit is not already included)

• Create test classes under `<project-name>/app/src/androidTest/java/<package-name>`

• Set up test run configuration (of type Android Test)
Creating Unit Tests (2)
Creating Unit Tests (3)
Passed Unit Test
Failed Unit Test

```java
public void testUICTest() {
    board.requestFocus();
    MotionEvent newMotionEvent = MotionEvent.obtain((long) 1,
            MotionEvent.ACTION_DOWN,
            (float) 53.0,
            (float) 53.0,
            0);
    board.dispatchTouchEvent(newMotionEvent);
}
```

```java
@UiThreadTest
public void testUICTest() {
    System.out.println("Thread ID in testUI: + Thread.currentThread().getId());
    board.requestFocus();
    for (i = 0; i < 3; i++) {
        MotionEvent newMotionEvent = MotionEvent.obtain((long) 1,
                (long) 1,
                MotionEvent.ACTION_DOWN,
                (float) x[i],
                (float) y[i],
                0);
        board.dispatchTouchEvent(newMotionEvent);
    }
}
```
Unit Test Class

// GameSessionFragmentTest.java
public class GameSessionFragmentTest extends ActivityInstrumentationTestCase2<GameSessionActivity> {
    // Template Class
    private GameSessionActivity mGameSessionActivity; // Activity to be tested
    private GameSessionFragment mGameSessionFragment; // Fragment to be tested
    private Board mBoard; // Member variable of activity
    // Data for the tests - touch coordinates
    final float x[]={(float)56.0, (float) 143.0, (float) 227.0};
    final float y[]={(float)56.0, (float) 143.0, (float) 227.0};
    int i = 0;

db
    public GameSessionFragmentTest() {...} // Constructor
    protected void setUp() throws Exception {...} // Setup - gets member variables
    public void testPreconditions() {...} // Test 1
    public void testUI() {...} // Test 2
    @UiThreadTest // Annotation to force the test to run in the UI thread
    public void testUiThreadTest(){...} // Test 3
    protected void tearDown() throws Exception {...} // Cleanup
}
Constructor

```java
public GameSessionFragmentTest() {
    super(GameSessionActivity.class);
}
```
protected void setUp() throws Exception { // Access member variables
    super.setUp();
    mGameSessionActivity = getActivity();
    mGameSessionFragment = new GameSessionFragment();
    mGameSessionActivity.getSupportFragmentManager()
        .beginTransaction()
        .add(R.id.fragment_container, mGameSessionFragment, null)
        .commit();

    // Wait for the Activity to become idle to avoid null Fragment references.
    getInstrumentation().waitForIdleSync();

    setActivityInitialTouchMode(false);
    mBoard = (Board) mGameSessionFragment.getView().findViewById(R.id.board);
} // ...

protected void tearDown() throws Exception { // cleans up
    mGameSessionActivity.finish();
    super.tearDown();
}
Test 1 – Test Preconditions

```java
@Test
public void testPreconditions() {
    assertNotNull(mGameSessionActivity);
    assertNotNull(mGameSessionFragment);
    assertNotNull(mBoard);
}
```
public void testUI() {
    System.out.println("Thread ID in testUI.run:" + Thread.currentThread().getId());
    getInstrumentation().waitForIdleSync();
    get.Activity().runOnUiThread(new Runnable() { // Run on UI thread
        public void run() {
            System.out.println("Thread ID in TestUI.run:" + Thread.currentThread().getId());
            board.requestFocus();
            // Simulates touch event
            // Hint: Instrumented the onTouchEvent(MotionEvent event) to get good pixel
            values for touch. Why not call onTouchEvent of Board directly?
            MotionEvent newMotionEvent =
                MotionEvent.obtain((long)1, (long)1, MotionEvent.ACTION_DOWN,
                    (float) 53.0, (float) 53.0, 0);
            board.dispatchTouchEvent(newMotionEvent); // Dispatches touch event
            mGameSessionFragment.scheduleAndroidsTurn();
            assertEquals(mGameSessionFragment.getPlayCount(), 0); // Assert 0 moves
        }
    });
    // Assertion does not work outside UI thread
}
Test 3 – Series of Moves

```java
final float x[] = {(float)56.0, (float) 143.0, (float) 227.0};
final float y[] = {(float)56.0, (float) 143.0, (float) 227.0};
int i = 0;
...

@UiThreadTest
public void testUiThreadTest() {
    System.out.println("Thread ID in testUI:" + Thread.currentThread().getId());
    mBoard.requestFocus();
    for (i=0; i<3; i++) {
        MotionEvent newMotionEvent = MotionEvent.obtain((long)1, (long)1,
                MotionEvent.ACTION_DOWN,
                (float) x[i], (float) y[i], 0);
        mBoard.dispatchTouchEvent(newMotionEvent);
    }
    assertEquals(mGameSessionFragment.getPlayCount(), 0);
}```
Tests and Threading

• Must explicitly run certain tests on UI thread
  – Via Annotations
  – Via explicit command
• Main UI thread subordinated to unit test thread
• Main UI thread terminated when tests run
• Tasks queued for main UI thread may not launch!
Modifications Required by Thread Model

// GameSessionFragment.java

public void scheduleAndroidsTurn() {
    Log.d(TAG, "Thread ID in scheduleAndroidsTurn:" + Thread.currentThread().getId());
    mBoard.disableInput();
    if (!mTestMode) {
        Random randomNumber = new Random();
        Handler handler = new Handler();
        handler.postDelayed(
            new Runnable() {
                public void run() {
                    androidTakesATurn();
                }
            },
            ANDROID_TIMEOUT_BASE + randomNumber.nextInt(ANDROID_TIMEOUT_SEED)
        );
    } else {
        androidTakesATurn();
    }
}

// Similar modifications needed for Kotlin. . .
Useful Links for Testing

• See: http://developer.android.com/reference/android/view/MotionEvent.html for details of MotionEvent class
• See: http://developer.android.com/reference/android/view/View.html for how to send an event to a View (Board is a subclass of View)
• http://blog.blundell-apps.com/android-gradle-app-with-robolectric-junit-tests/ (JUnit and Robolectric via Gradle and Android Studio)
References

• Chapter 8: Making Your Application Fast and Responsive, from *Android SDK 3 Programming for Dummies*
• Jon Bentley, *Writing Efficient Programs*, [www.crowl.org/lawrence/programming/Bentley82.html](http://www.crowl.org/lawrence/programming/Bentley82.html)