Upcoming Assignments

- Readings: Chapter 6 by today
- Lab 3 due today (complete survey)
- Lab 4 will be available Friday (due February 5)
- Friday Quiz in Blackboard 2:10-3pm (Furlough)
- Vertical Prototype due Monday, February 1
  - Be prepared to demo them in class
- How-to's and presentations
  - https://sites.google.com/site/androidhowto/presentations

Vertical Prototype

- See https://sites.google.com/site/androidappcourse/assignments/vertical-prototype
- Questions?

Persistence

- Three ways to store data
  - Shared Preferences
  - Files
  - SQLite databases
- Mechanism to store/access private data
  - Content Providers

Shared Preferences

- Three forms:
  - Share across all components in an application
    - getSharedPreferences("SomeString",Activity.MODE_PRIVATE);
  - Store only data needed by this Activity
    - getPreferences(Activity.MODE_PRIVATE);
  - Store only data needed by this Activity when Activity becomes inactive (but not when finished)
    - Ex. Orientation change from portrait to landscape
    - use Bundle in onSaveInstanceState/onRestoreInstanceState/onCreate

Shared Preferences

- Across all components

```
public static final String CMPREFS = "CampusMapSharedPreferences";

private void savePreferences() {
    SharedPreferences cmSharedPreferences =
            getSharedPreferences(CMPREFS,Activity.MODE_PRIVATE);
    SharedPreferences.Editor editor = cmSharedPreferences.edit();
    editor.putBoolean(VIRTUAL_MODE, inVirtualMode);
    editor.putInt(MAP_INDEX, curMapIndex);
    editor.commit();
}
private void restoreUIState() {
    SharedPreferences cmSharedPreferences =
            getSharedPreferences(CMPREFS,Activity.MODE_PRIVATE);
    inVirtualMode = cmSharedPreferences.getBoolean(VIRTUAL_MODE, true);
    curMapIndex = cmSharedPreferences.getInt(MAP_INDEX, 0);
}
```

Shared Preferences

- Only for this Activity (each Activity has one)

```
private void savePreferences() {
    SharedPreferences cmActivityPreferences =
            getPreferences(Activity.MODE_PRIVATE);
    SharedPreferences.Editor editor = cmActivityPreferences.edit();
    editor.putBoolean(VIRTUAL_MODE, inVirtualMode);
    editor.putInt(MAP_INDEX, curMapIndex);
    editor.commit();
}
private void restoreUIState() {
    SharedPreferences cmActivityPreferences =
            getPreferences(Activity.MODE_PRIVATE);
    inVirtualMode = cmActivityPreferences.getBoolean(VIRTUAL_MODE, true);
    curMapIndex = cmActivityPreferences.getInt(MAP_INDEX, 0);
}
```

Shared Preferences

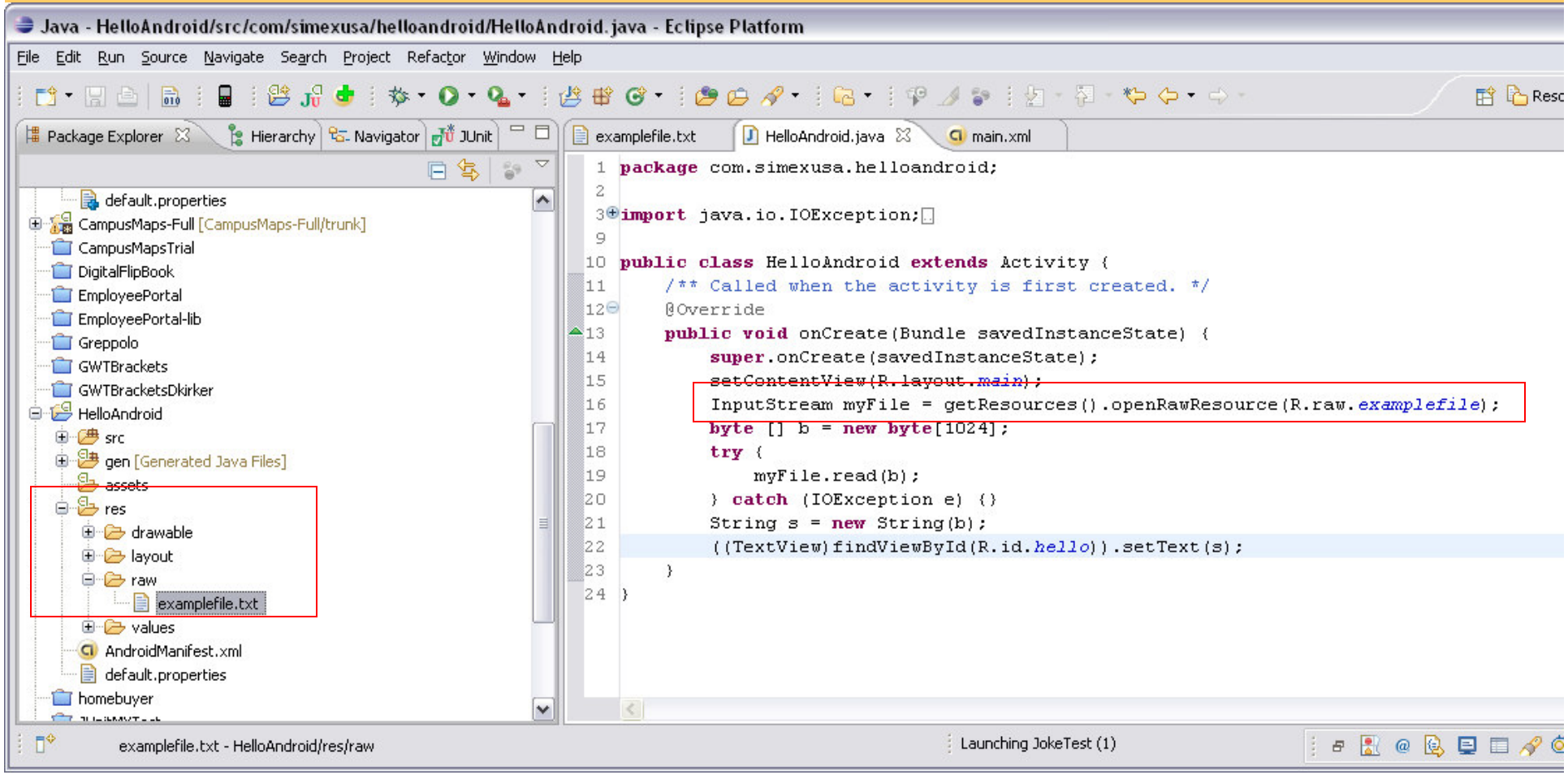
- Only for this Activity when inactive/active
  - Note: onSaveInstanceState called when an Activity becomes inactive, not when closed by finish() or user pressing back button

```
public static final String TEXTVIEW_STATE_KEY = “TEXTVIEW_STATE_KEY";

private void onSaveInstanceState(Bundle outState) {
    outState.putString(TEXTVIEW_STATE_KEY,
            (TextView)findViewById(R.id.myTextView).getText().toString());
    super.onSaveInstanceState(outState);
}
private void onRestoreInstanceState(Bundle inState) {  //same for onCreate()
    if(inState != null && inState.containsKey(TEXTVIEW_STATE_KEY)) {
        (TextView)findViewById(R.id.myTextView).
            setText(inState.getString(TEXTVIEW_STATE_KEY));
    }
}
```

Files

- Generally not recommended to use files
- Store read-only static files in **res/raw**



Files

- Standard java.io is available
- Can also use openFileOutput and openFileInput

```
byte[] b = new String("Yo").getBytes
try {
        FileOutputStream fos =
                openFileOutput("anotherExampleFile.txt",
                Context.MODE_WORLD_WRITEABLE);
        fos.write(b);
        FileInputStream fis =
                openFileInput("anotherExampleFile.txt");
        fis.read(b);
} catch (IOException e) {}
```

Also available:
Context.MODE_PRIVATE
Context.MODE_WORLD_READABLE

SQLite Databases

- RDBMS provided through a library so it becomes part of your app
- Use the SQL you learned in database course
- Use Db best practices
  - Normalize data
  - Encapsulate db info in helper or wrapper classes
- Don't store files (e.g. images or audio)
  - Instead store the path string

Creating a Database

```
SQLiteDatabase myDb =
        openOrCreateDatabase("example.db",
                             Context.MODE_PRIVATE,
                             null);  //optional CursorFactory
myDb.execSQL("drop table if exists jokeTable");
myDb.execSQL("create table jokeTable " +
                    " ( _id integer primary key autoincrement," +
                    "author text not null, joke text not null);");
```

Inserting Data

- ContentValues are key/value pairs that are used when inserting/updating databases
- Each ContentValue object corresponds to one row in a table

```
ContentValues newValues = new ContentValues();
newValues.put("author", "David Janzen");
newValues.put("joke", "This is a joke");
myDb.insert("jokeTable",null,newValues);
```

Updating Data

```
ContentValues updatedValues = new ContentValues();
updatedValues.put("joke", "This is a better joke");
myDb.update("jokeTable",
                    updatedValues,
                    "author='David Janzen'",  //where clause
                    null);                    //whereArgs
```

Querying Data with query()

```
Cursor jokes = myDb.query("jokeTable",
                                    null, //columns
                                    null, //where clause
                                    null, //args if ? in where clause
                                    null, //groupBy
                                    null, //having
                                    null); //orderBy
if (jokes.moveToFirst()) {
   do {
      String author = jokes.getString(1);
      String joke = jokes.getString(2);
      ((TextView)findViewById(R.id.hello)).setText(author +
                                                ": " + joke);
   } while(jokes.moveToNext());
}
```

Other Cursor Functions

- moveToPrevious
- getCount
- getColumnIndexOrThrow
- getColumnName
- getColumnNames
- moveToPosition
- getPosition

Deleting Data

```
myDb.delete("jokeTable",
            "author='David Janzen'",
            null);
```

SQLiteOpenHelper

- See example in text p. 177-179

Content Providers

- Apps can expose their data layer through a Content Provider, identified by a URI
- Some native apps provide Content Providers
- Your apps can provide Content Providers

Content Resolver

- Each application Context has a single ContentResolver that can be used to access a Content Provider

URI,colums,where,whereArgs,orderBy

```
Cursor allRows = getContentResolver().query(
                    People.CONTENT_URI, null, null, null, null);
startManagingCursor(allRows);
int nameIdx = allRows.getColumnIndexOrThrow(People.NAME);
int phoneIdx = allRows.getColumnIndexOrThrow(People.NUMBER);
if (allRows.moveToFirst()) {
   do {
      String name = allRows.getString(nameIdx);
      String number = allRows.getString(phoneIdx);
      //do something with name and number
   } while(allRows.moveToNext());
}
```

Requires: <uses-permission android:name="android.permission.READ_CONTACTS" />

Insert, Update, Delete

- Similar to SQLiteDatabase

```
SQLiteDatabase myDb = openOrCreateDatabase(…);
ContentValues newValues = new ContentValues();
newValues.put("author", "David Janzen");
newValues.put("joke", "This is a joke");
myDb.insert("jokeTable",null,newValues);
```

```
ContentValues newValues = new ContentValues();
newValues.put("author", "David Janzen");
newValues.put("joke", "This is a joke");
getContentResolver().insert(MY_CONTENT_URI, newValues);
//or
//ContentValues[] valueArray = new ContentValues[10];
//getContentResolver().bulkInsert(MY_CONTENT_URI, valueArray);
```

- Update and Delete are similar

Creating Your Own Content Provider

```
public class MapProvider extends ContentProvider {
  private static final String mapURI =
                        "content://com.simexusa.provider.campusmaps/maps";
  public static final Uri CONTENT_URI = Uri.parse(mapURI);
  @Override public boolean onCreate() {
        //create the database and tables
  }
  //handle both forms of Content URIs (…/maps or …/maps/3 for third one)
  private static final int ALLROWS = 1;
  private static final int SINGLE_ROW = 2;
  private static final UriMatcher uriMatcher;
  static {
     uriMatcher = new UriMatcher(UriMatcher.NO_MATCH);
     uriMatcher.addURI("com.simexusa.provider.campusmaps","maps",ALLROWS);
     uriMatcher.addURI("com.simexusa.provider.campusmaps","maps/#",SINGLE_ROW);
  }
}
```

Provide query, insert, delete, update

```
@Override public Cursor query(Uri uri, String[] projection, String selection,
                              String[] selectionArgs, String sort) {
    switch (uriMatcher.match(uri)) {
        case SINGLE_ROW :
          …                                  //see p.201 for example
        case ALLROWS :
          …
        default: :
          …
    }
    return null;
}
```

getType() and manifest

```
@Override public String getType(Uri uri) {
        switch (uriMatcher.match(uri)) {
            case SINGLE_ROW :
              return “vnd.simexusa.cursor.item/mapprovider”;
            case ALLROWS :
              return “vnd.simexusa.cursor.dir/mapprovider”;
            default: :
              throw new IllegalArgumentException(“Unsupported URI: “ + uri);
        }
 }
```

- In ApplicationManifest.xml

```
<provider android:name=“MapProvider”
              android:authorities=“com.simexusa.provider.campusmap” />
```