S7 Inf A3 — Java for Graphical and Mobile Programming · Centrale Lyon

Stéphane Derrode

CM8

Persistence & App Polish

- SharedPreferences — save key-value data
- Saving & restoring instance state (rotation)
- Snackbar, AlertDialog — better UX
- App icon, theme & colors
- Beyond this course — what Android can do

Module Overview

Part 1 — Java Fundamentals		Part 2 — GUI with Swing		Part 3 — Android	
CM1	From Python to Java	CM4	Swing: Components & Layouts	CM6	Android Architecture
CM2	Collections, Exceptions & I/O	CM5	Graphics 2D & Events	CM7	Layouts, Navigation & Intents
CM3	Inheritance, Interfaces			CM8	Persistence & Threads

Projects: ★ Spider Game (Part 2) ★ Calculator / Currency Converter (Part 3)

Exam 50% + BEs 50%

Stéphane Derrode — Centrale Lyon · S7 Inf A3

CM8 — Agenda

01 SharedPreferences — persist data

02 Instance state — survive rotation

03 Snackbar & AlertDialog

04 App icon, theme & Material Design

05 NotesPad v3 — final app

06 Beyond this course

SharedPreferences — Key-Value Persistence

Stores simple data that survives app close and device restart

```
// ── WRITING ──────────────────────────────────────────

// Get a SharedPreferences file (name = your choice)
SharedPreferences prefs = getSharedPreferences(
    "NotesPadPrefs", Context.MODE_PRIVATE);

// Open an editor
SharedPreferences.Editor editor = prefs.edit();

// Save values
editor.putString("LAST_NOTE", noteText);
editor.putInt("NOTE_COUNT", notes.size());
editor.putBoolean("DARK_MODE", true);

// Commit — always call one of these!
editor.apply();   // async — recommended
editor.commit();  // sync  — blocks UI

// ── READING ──────────────────────────────────────────

SharedPreferences prefs = getSharedPreferences(
    "NotesPadPrefs", Context.MODE_PRIVATE);

// Read values (with defaults if key missing)
String last  = prefs.getString("LAST_NOTE", "");
int    count = prefs.getInt("NOTE_COUNT", 0);
```

apply() vs commit()

apply() writes asynchronously — never blocks the UI. commit() writes synchronously — returns boolean success. Always use apply() unless you need confirmation in the same thread.

Where to read/write

Write in onPause() — called before the app loses focus. Read in onCreate() — restore state on startup. This follows the Activity lifecycle.

Supported types

String int float long boolean Set<String>
For complex data (List, objects) → serialize to JSON with Gson, or use a database (Room).

Storing a List<String> as JSON

```
Gson gson = new Gson();
String json = gson.toJson(notes);
editor.putString("NOTES", json);
// Read: new TypeToken<List<String>>(){}
```

NotesPad — Persisting the Notes List

```
// In MainActivity.java

// Add to build.gradle first:
// implementation 'com.google.code.gson:gson:2.10.1'
import com.google.gson.Gson;
import com.google.gson.reflect.TypeToken;
import java.lang.reflect.Type;

private static final String PREFS = "NotesPadPrefs";
private static final String KEY_NOTES = "notes_json";
private List<String> notes = new ArrayList<>();

@Override
protected void onCreate(Bundle saved) {
    super.onCreate(saved);
    setContentView(R.layout.activity_main);
    loadNotes();        // ← restore from storage
    setupRecyclerView();
    setupAddButton();
}

@Override
protected void onPause() {
    super.onPause();
    saveNotes();        // ← persist before losing focus
}
```

```
private void saveNotes() {
```

Key design choices

JSON for the list

SharedPreferences stores String — Gson serializes List<String> to JSON and back. One Gson line to save, one to load. Add implementation 'com.google.code.gson:gson:2.10.1' in

Save in onPause()

onPause() is the LAST guaranteed lifecycle callback. If the app is killed (e.g. low memory), onStop() and onDestroy() may not be called. onPause() always runs.

Load in onCreate()

Always load before setting up the adapter, otherwise the RecyclerView shows an empty list. If json is null (first launch), notes stays as an empty ArrayList — correct.

Testing persistence

1. Add some notes.
2. Press the device Home button (don't Back).
3. Reopen the app.

→ Notes should still be there.

4. Press Back (finish) → reopen → still there.

Surviving Rotation — onSaveInstanceState

Rotation destroys and recreates the Activity — all local state is lost

```
// In EditNoteActivity.java

// ── Called BEFORE the Activity is destroyed ──────────
@Override
protected void onSaveInstanceState(Bundle out) {
    super.onSaveInstanceState(out);

    // Save whatever the user was typing
    EditText editor = findViewById(R.id.editor);
    out.putString("DRAFT", editor.getText()
                                 .toString());
    out.putInt("CURSOR", editor.getSelectionStart());
}

// ── Called in onCreate() IF Activity is restored ─────
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_edit_note);

    // Restore draft if rotating (savedInstanceState ≠ null)
    if (savedInstanceState != null) {
        String draft = savedInstanceState
            .getString("DRAFT", "");
        int cursor = savedInstanceState
            .getInt("CURSOR", 0);
```

What triggers recreation?

Screen rotation (most common)
Language change
Keyboard shown/hidden
Font size change
Any configuration change

savedInstanceState vs SharedPreferences

savedInstanceState — temporary, for the current session. Cleared when user presses Back.
SharedPreferences — permanent, survives app close.

Quick fix — android:configChanges

android:configChanges="orientation|screenSize" in Manifest prevents recreation on rotation — the app just stretches. Avoid this — it hides the real problem.

RecyclerView state is automatic

RecyclerView saves its scroll position automatically via the LayoutManager. You only need to save the list data (via SharedPreferences) and the RecyclerView restores its position.

Better UX — Snackbar & AlertDialog

```
// ── Snackbar — better than Toast ──────────────────
// Add to build.gradle:
// implementation
'com.google.android.material:material:1.11.0'
import com.google.android.material.snackbar.Snackbar;

// Show a Snackbar with an undo action
Snackbar.make(
    findViewById(R.id.root),  // anchor view
    R.string.note_deleted,    // message
    Snackbar.LENGTH_LONG)
    .setAction(R.string.undo, v -> {
        notes.add(deletedIndex, deletedNote);
        adapter.notifyItemInserted(deletedIndex);
    })
    .show();


// ── AlertDialog — confirmation dialog ─────────────
new AlertDialog.Builder(this)
    .setTitle(R.string.confirm_delete)
    .setMessage(R.string.delete_message)
    .setPositiveButton(R.string.delete, (d, w) -> {
        // User confirmed — delete the note
        String deleted = notes.remove(position);
        adapter.notifyItemRemoved(position);
        showUndoSnackbar(deleted, position);
    })
```

Snackbar vs Toast

Toast — simple message, no action, appears on top of everything.
Snackbar — anchored to bottom, can have one action (Undo, Retry…), dismisses on swipe.
Prefer Snackbar for reversible actions.

AlertDialog buttons

setPositiveButton — confirmative action (Delete, OK, Send)
setNegativeButton — cancel / dismiss (null = just dismiss)
setNeutralButton — third option (Later, More info)
The (dialog, which) lambda params can be ignored

Delete + Undo pattern

1. User swipes or long-presses → AlertDialog
2. User confirms → remove item, show Snackbar with Undo
3. User taps Undo → re-insert item

This is the Material Design recommended pattern

ItemTouchHelper — swipe to delete

ItemTouchHelper.SimpleCallback lets users swipe a RecyclerView item left/right to delete it — built into the RecyclerView library.

App Icon, Theme & Material Design

```
<!-- res/values/themes.xml -->
<resources>
    <style name="Theme.NotesPad"

parent="Theme.MaterialComponents.DayNight.DarkActionBar"
>

        <!-- Primary brand color -->
        <item name="colorPrimary">
            @color/indigo_500</item>
        <item name="colorPrimaryVariant">
            @color/indigo_700</item>
        <item name="colorOnPrimary">@color/white</item>

        <!-- Secondary accent color -->
        <item name="colorSecondary">
            @color/cyan_400</item>
        <item name="colorSecondaryVariant">
            @color/cyan_700</item>
        <item name="colorOnSecondary">
            @color/black</item>

        <!-- Status bar color -->
        <item name="android:statusBarColor">
            ?attr/colorPrimaryVariant</item>
    </style>
</resources>
```

App icon — Adaptive Icons (API 26+)

```
<!-- res/mipmap-anydpi-v26/ic_launcher.xml --
>
<adaptive-icon
  xmlns:android="...">
  <background
    android:drawable=
      "@drawable/ic_launcher_background"/>
  <foreground
    android:drawable=
      "@drawable/ic_launcher_foreground"/>
</adaptive-icon>
```

Easiest way to set an icon

Right-click res → New → Image Asset. The wizard generates all density versions (mdpi, hdpi, xhdpi…) and the adaptive icon XML automatically. Requires a SVG or PNG source.

Material Design 3 colors

https://m3.material.io/theme-builder — generate a complete color scheme from a single brand color. Export directly to Android XML. Takes 2 minutes.

NotesPad v3 — The Complete App

Everything built over 3 CMs + 2 TDs

CM6

- ✓ MainActivity with EditText + Button
- ✓ setOnClickListener lambda
- ✓ Toast for feedback
- ✓ Log.d() for debugging

CM7

- ✓ ConstraintLayout UI
- ✓ RecyclerView + NoteAdapter
- ✓ EditNoteActivity (2nd screen)
- ✓ Intent + putExtra / getExtra
- ✓ ActivityResultLauncher for result
- ✓ strings.xml + i18n

CM8

- ✓ SharedPreferences (Gson JSON)
- ✓ onSaveInstanceState (rotation)
- ✓ Snackbar with Undo action
- ✓ AlertDialog confirmation
- ✓ App icon + Material theme

What's left for the project (BE2)

The Calculatrice or Convertisseur project uses the same patterns: one or two Activities, EditText for input, Button listeners, SharedPreferences for history, Snackbar for feedback. All tools are in your toolkit.

Beyond This Course — Android Goes Much Further

What we deliberately didn't cover — and where to learn it:

Room (SQLite database)

Structured persistence for complex data. Annotation-based ORM. Replaces SharedPreferences for large datasets.

developer.android.com/training/data-storage/room

ViewModel & LiveData

Architecture Components — separate UI state from Activity lifecycle. Data survives rotation without onSaveInstanceState.

developer.android.com/topic/libraries/architecture

Fragments

Reusable UI components within an Activity. Used for navigation (Navigation Component), tablets, and Master-Detail layouts.

developer.android.com/guide/fragments

Retrofit & REST APIs

HTTP client library for Android. Connect your app to a web service (weather, maps, any JSON API) in a few lines.

square.github.io/retrofit

Permissions & sensors

Runtime permission requests (camera, location, microphone). Access GPS, accelerometer, compass, Bluetooth.

developer.android.com/training/permissions

Kotlin & Jetpack Compose

Modern Android development: Kotlin instead of Java, declarative UI instead of XML. The future of Android development.

developer.android.com/jetpack/compose

Part 3 Recap — Android Skills Acquired

What you can now build — a two-Activity Android app:

Architecture

Activity, lifecycle, Manifest, R class, resources

Layouts

LinearLayout (weights), ConstraintLayout, XML vs Java

Widgets

TextView, EditText (inputType), Button, ImageView, Spinner

Events

setOnClickListener lambda — same pattern as Swing

Feedback

Toast, Snackbar (+ Undo action), AlertDialog

Navigation

Explicit Intent, putExtra, ActivityResultLauncher

Persistence

SharedPreferences + Gson for simple data

Polish

strings.xml, app icon, Material theme, dark mode

Project BE2 — Calculatrice or Convertisseur
All the patterns are in your toolkit. Choose the project that excites you most and push the ideas further — see the project brief for advanced ideas.

CM8 — Key Takeaways

SharedPreferences	getSharedPreferences(); editor.putString(); apply() — save in onPause, load in onCreate
Gson for lists	Serialize List<String> to JSON string for SharedPreferences; add com.google.code.gson
Instance state	onSaveInstanceState(out) before rotation; restore in onCreate if savedState != null
Snackbar	Better than Toast — anchored, swipeable, supports one Undo action
AlertDialog	AlertDialog.Builder — confirm destructive actions before executing them
Theme & icon	themes.xml + Image Asset wizard; use ?attr/ references for dark mode compatibility

Up next → Project BE2: Calculatrice or Convertisseur