## Group Learning (QLC‑6): Reporting

### 

### Objective

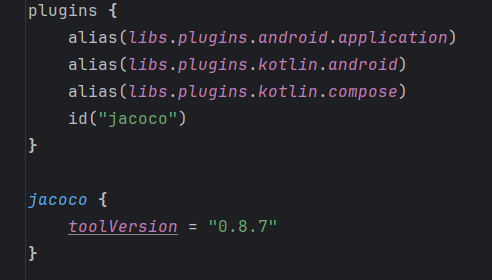
Deepen your understanding of both UI‑testing (QLC4) and async loading (QLC 5) by exploring how small code changes affect test outcomes and coverage. You’ll work in groups to introduce “What If…” tweaks, observe test failures and coverage gaps, then discuss how to shore up your tests.

### Scenario (Optional)

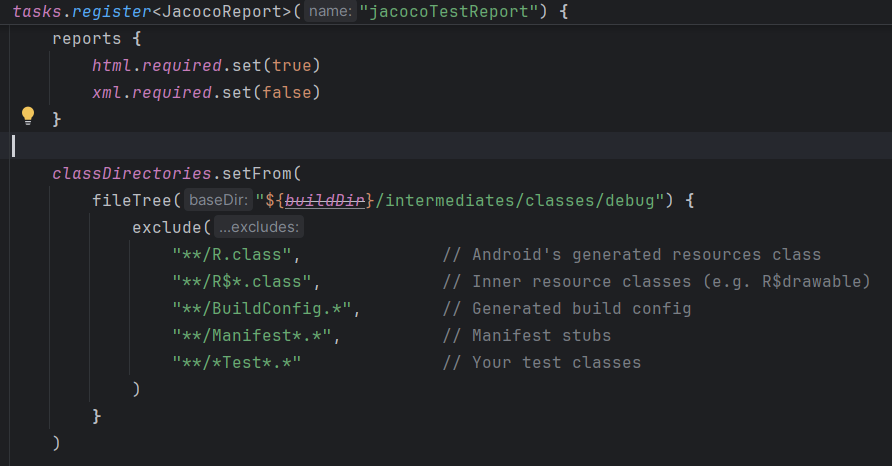
In previous labs you retrofit UI tests (QLC4) and added async state logic (QLC 5). Now, you’ll intentionally break or extend behavior in multiple layers, run your existing tests and JaCoCo report, and discover where coverage and tests fall short.

### Prerequisites :

* **JaCoCo plugin enabled** in your module’s **build.gradle**:



* Below you buildFeatures section in the same file,. Place:



* Run with .\gradlew testDebugUnitTest jacocoTestReport

### Steps

1. **Form Groups** (2–4 students) and pick scenarios from the lists below (or invent your own).
2. **Apply the Change**
   * Make the code tweak in either the UI layer (MoodSelectorScreen) or ViewModel (MoodSelectorViewModel).
3. **Run Tests & Generate Coverage**
   * Execute all UI and unit tests.
   * Generate the JaCoCo report (e.g., ./gradlew jacocoTestReport).
4. **Observe Results**
   * Note which tests fail (if any).  
     Identify untested lines/branches in the HTML report.
5. **Discuss in Your Group**
   * Why did coverage drop or tests break?
   * What does that reveal about test focus or missing assertions?
   * Which additional test(s) would you write to catch the gap?
6. **Report Back (2 min per group)**
   * Show your scenario and code change.
   * Highlight coverage differences.
   * Summarize one key lesson about test coverage vs. behavior.

### Scenario Options

* **QLC 4 UI‑Testing**
  + Remove a Modifier.testTag("button‑$mood").
  + Delete the “pre‑selection” UI test.
  + Change the fake repo to return only one mood.
* **QLC 5 Async‑Loading**
  + Comment out \_state.value = UiState.Loading.
  + Remove the catch { … } block in loadMoods().
  + Omit advanceUntilIdle() in your success test.
* **Combined Scenarios**
  + Remove a testTag **and** bypass the Success emission.
  + Insert an unreachable branch (e.g., if(false) \_state.value = UiState.Empty).
  + Hard‑code Dispatchers.IO in the ViewModel instead of injecting.