Androidskolen Sesjon 3

Thou shalt test!

Agenda

- Dagens målsetning
- Hvorfor teste
- Android testing
- Unit tester + praktisk
- Instrumenterte tester + praktisk
- UI tester (Espresso) + praktisk
- Debugging

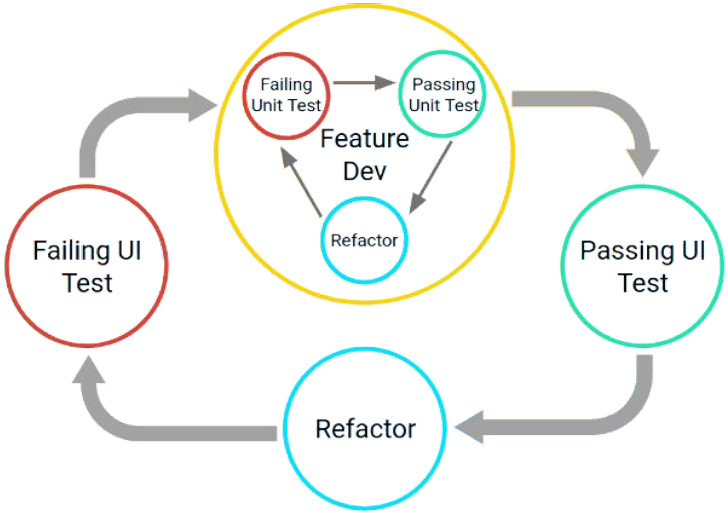
Dagens målsetning

Deltakerne skal være inneforstått med hvorfor og hvordan man tester på Android-plattformen. Dette inkluderer både vanlige JVM-tester og instrumenterte tester som kjøres på en tilkoblet enhet.

Deltakerne skal også bli kjent med hvilke debugging-muligheter som er tilgjengelig, og hvordan dette skiller seg fra f.eks. debugging av vanlige Java-applikasjoner.

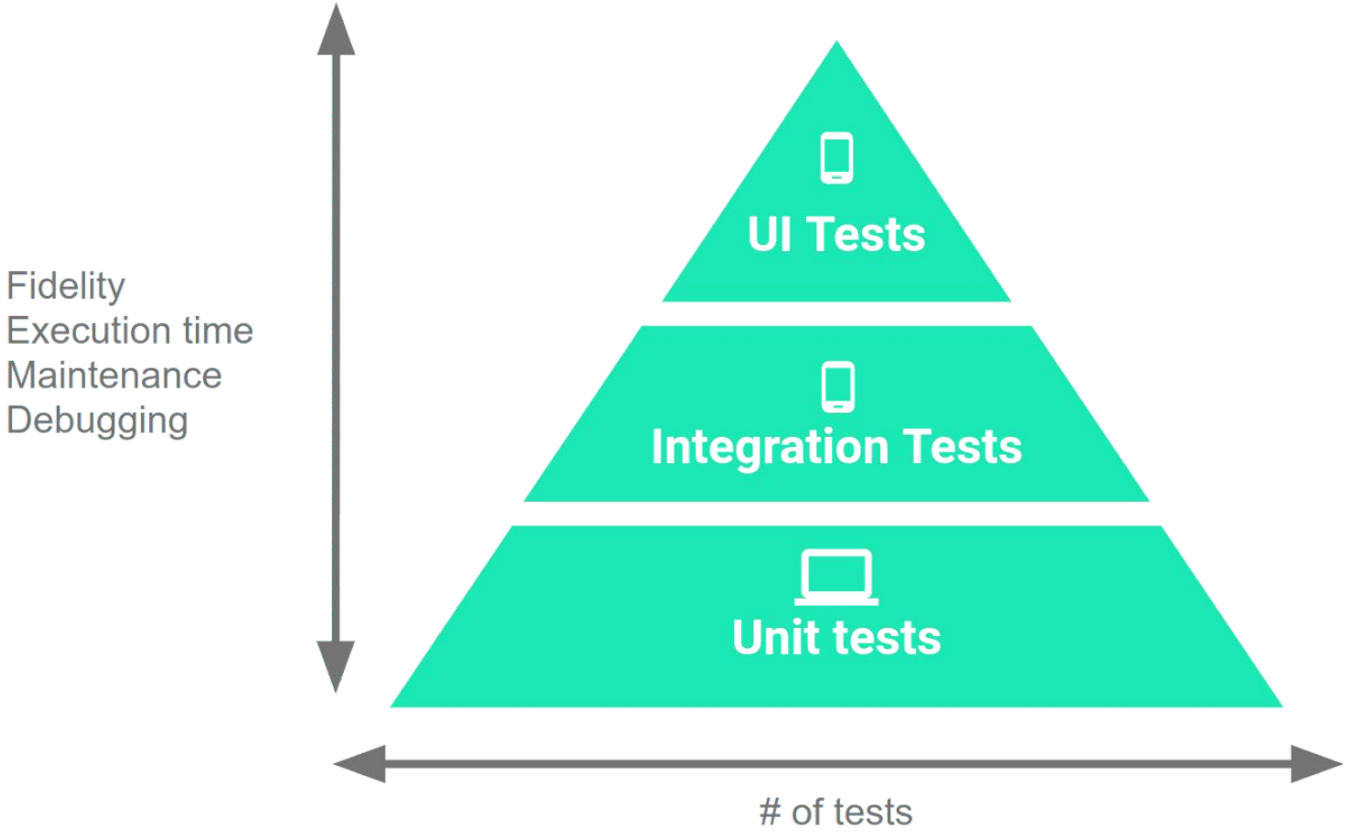
Hvorfor teste

- Forbedret produktkvalitet
- Forenkle vedlikehold og QA
- Tryggere release
- Redusere manuelle rutinesjekker
- Hjelper deg å skrive bedre kode



bouvet

Android testing



Android testing

Testene burde fokusere på business logikk, en god tommelfingerregel er å ha følgende distribusjon:

- 70-80% *unit tests* - forsikre stabilitet av kodebasen.
- 20-30% *functional tests* - validere at applikasjonen faktisk fungerer.
- *Cross functional tests* - hvis applikasjonen integrerer med andre applikasjon komponenter.

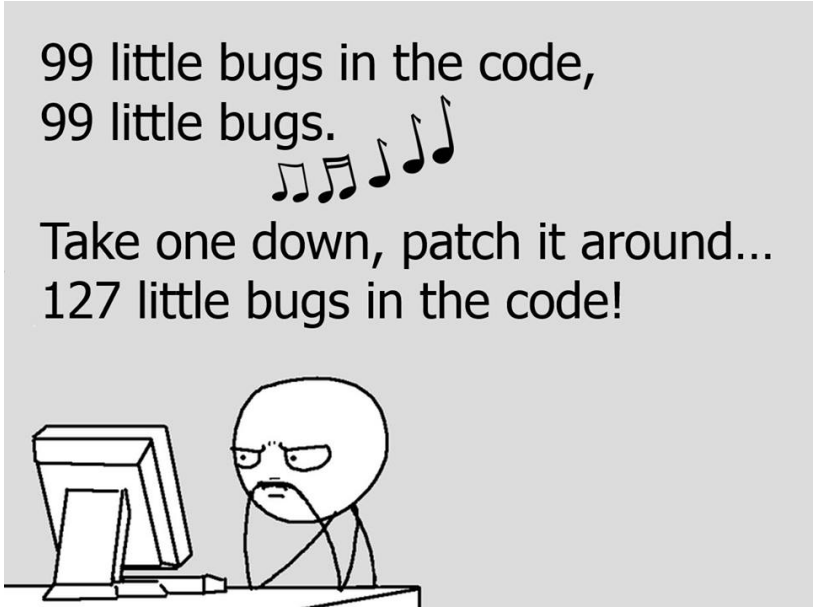
Android prosjektorganisering for tester

Default mappestruktur for applikasjon og test kode:

- `app/src/main/java`- Applikasjonens kildekode
- `app/src/test/java` - Unit tester som kan kjøre i JVM
- `app/src/androidTest/java` - Alle tester som må kjøre på en Android enhet

Unit tester

- Raske å kjøre
- Lette å skrive, kjøre og måle dekningsgrad
- Separere - MVP eller MVVM



“Separate your (testable) app logics from Android specific (difficult-to-test) stuff”

Praktisk del 1 (valgfritt)

- Sjekk ut kode fra
  http://github.com/bouvet/AndroidSkolen.git

- Implementer testlogikk i CalculatorViewModelTest

Instrumenterte tester

- Kjører på emulator eller device
- Tilgang på det meste applikasjonen har
  - Context funker
  - Resources kan hentes
- Kan til dels emulere interaksjoner i grensesnittet
  - Input-felter via setText
  - Knapper via performClick

- Tar vesentlig lengre tid og mer ressurser

Praktisk del 2

- Test funksjonaliteten i “Hello” aktiviteten ved hjelp av ActivityTestRule i HelloInstrumentedTest

- Test at å åpne “Calculator” aktiviteten med en predefinert Intent gir korrekt resultat i CalculatorInstrumentedTest

UI tester (Espresso)

- Når vanlige instrumenterte tester ikke er “ekte” nok

- Kjører på emulator eller device
- Emulerer bruker-input mer eller mindre direkte
- Kan enten skrives manuelt eller “recordes”

- Kan også kjøres i Firebase Test Lab

Praktisk del 3

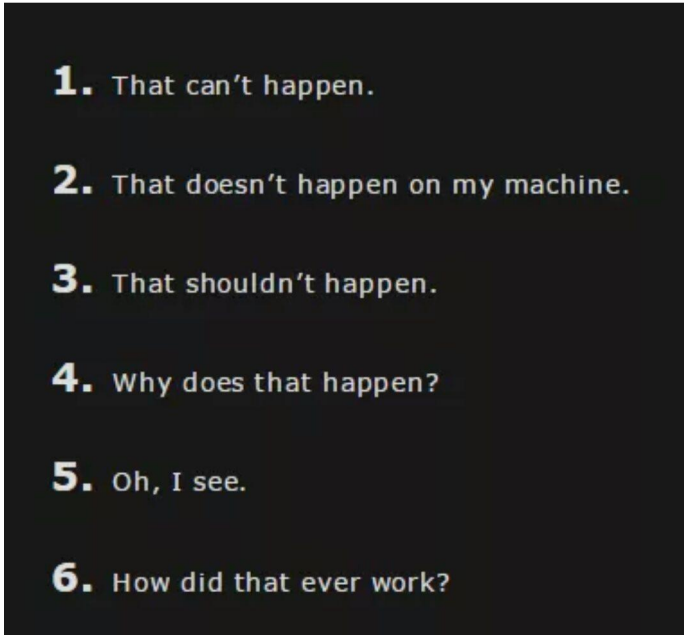
- Repeter oppgavene fra praktisk del 2 med Espresso-kode i HelloEspressoTest og CalculatorEspressoTest

- Prøv å recorde en manuell gjennomføring av testen med Record Espresso Test, og sammenlign med den manuelle koden

Debugging

- Stort sett som vanlig JVM debugging
- Koble til emulator eller device
- Aktive prosesser (app / service)
- Release-bygg kan ikke uten videre debugges

Six Stages of Debugging



bouvet

Oppsummering

- “Vanlige” tester er dramatisk raskere og enklere å både lage og kjøre. Legg opp applikasjonen så det meste kan testes slik
- Instrumenterte tester lar deg teste “resten”, men krever mer ressurser både å sette opp og kjøre
- Debugging er *stort sett* som vanlig JVM debugging