1 TDD generelt

Spesifikasjon: tester og krav og to sider av samme sak. Tester kan derfor beskrive krav og vi slipper to artefakter for samme sak. God, ren kode. Mye trygghet.

Motivasjon

- tester fungerer utmerket som kravdokumentasjon. I agil ånd gir det ikke mening å duplisere kravdokumentasjonen i både kravdok og Tester
- en akseptansetest kan beslriver som scenarior i en user story og funger ebåde som funksjonell test og som kravdok.
- En enhetstest fungerer besdre som dokumentasjon av enn klasse enn javadoc(doc).
- Viktig med detaljerte navn på testene.
- God, ren kode.
- Bruk av DI tydeligjør avhengigheter, og muligjør mocking. Dette er nødvendig for tester som involverer eksterne systemer
- testbar kode er ikke nyttig kun for testene sin skyld, fordi den fremtvinger løse koblinger og høy cohesion
- Tester gir en viss trygghet for at koden gjør det den skal
- Vi får kjørt koden med en gang

Testing generelt

- Vi tester programvare for å validere at systemet:
- responderer riktig på alle typer input
- yter godt
- er brukervennlig
- kan installeres i riktig miljø
- ikke brekker ved endringer

Tradisjonell vs smidig testing

- Utviklere og testere jobber sammen
- CI gjør versjonsforskjellene små

- Dette gjør at det er mindre sannsynelig at det oppstår store feil
- Risk poker for å identifisere risiko
- Testnivåer
-
  - System test: tregt og kostbart, tester fullt integrert system
  - Integration test: medium: tester større komponenter spiller bra sammen
  - Unit test: billig og rask: små enheter, ofte klassen, Junit. En del av byggeprosessen CI
  - og:
  - Akseptansetest: en type systemtestsom typisk utføres av kunden
  - Regresjonstest: skal sikre at systemet ikke brekker ved endringer
  - Smoke-testing: Uformell test, hvor man raskr prøver ut de viktigste delene av et system for å sjekke at alt henger på greip.
  - Utforskende testing: bruker kunnskap til å utføre kreativ testing
  - Destruktiv testing: selvforklarende.
  - Usability testing: teste om et system er lett å bruke.
  - Ytelsetesting: handler om å finne ut hvor godt et system yter. Finne responstider osv.
  - Stabilitetstesting: Finne problemer som oppstr under korte perioder med høyt stressnivå

2 Test first

1. Write a test, watch it fail
2. Write just enough code to pass the test
3. Improve the code without changing its behaviour

3 CD

- Bygge koden automatisk
- Etter bygging så kjører alle testene
- Gradle, Maven osv
- GitLab CI med Docker(hvem som helst kan lage og publisere et Docker image som inkluderer en linux server med all software man trenger.)
- yml fil
-