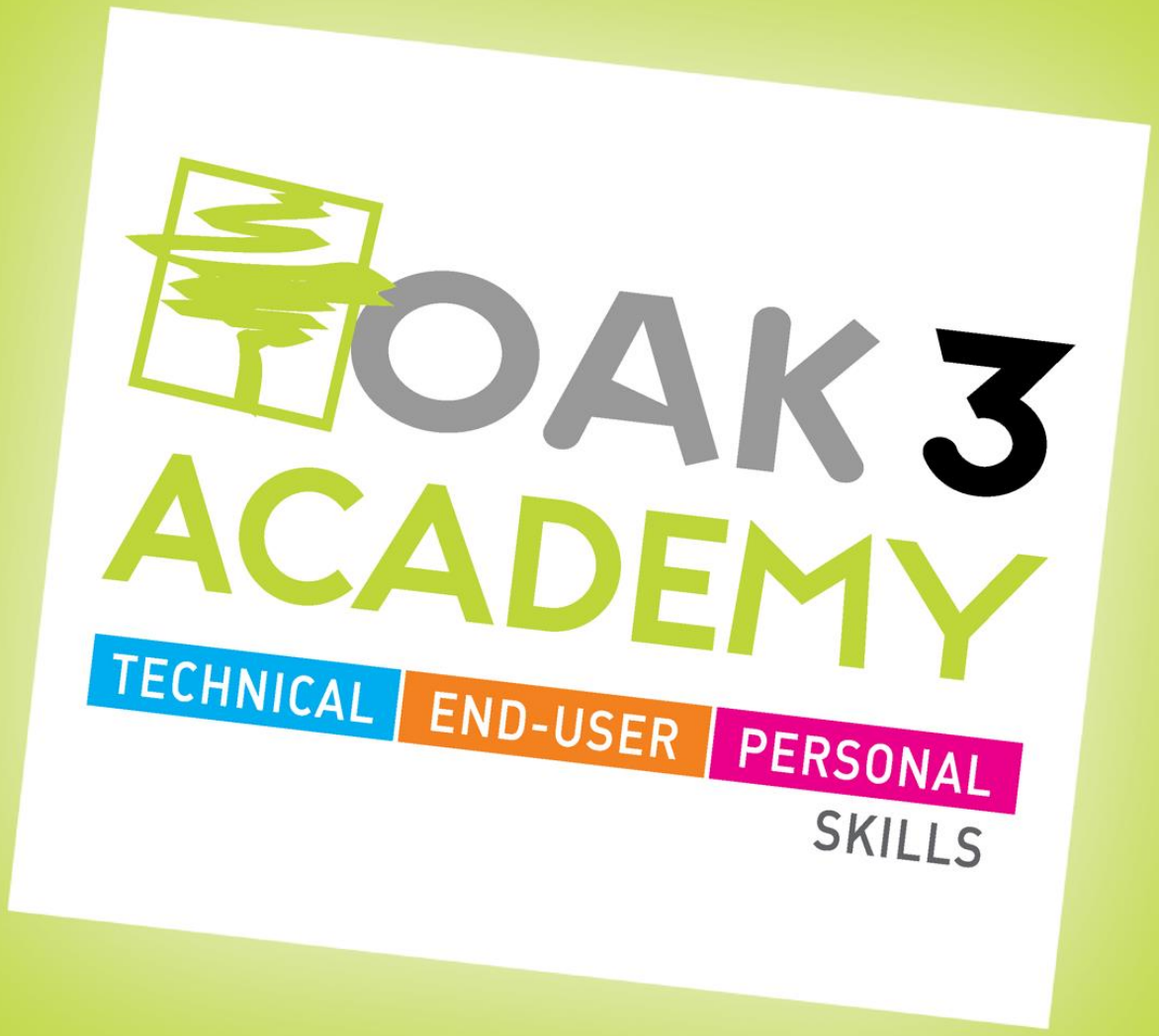
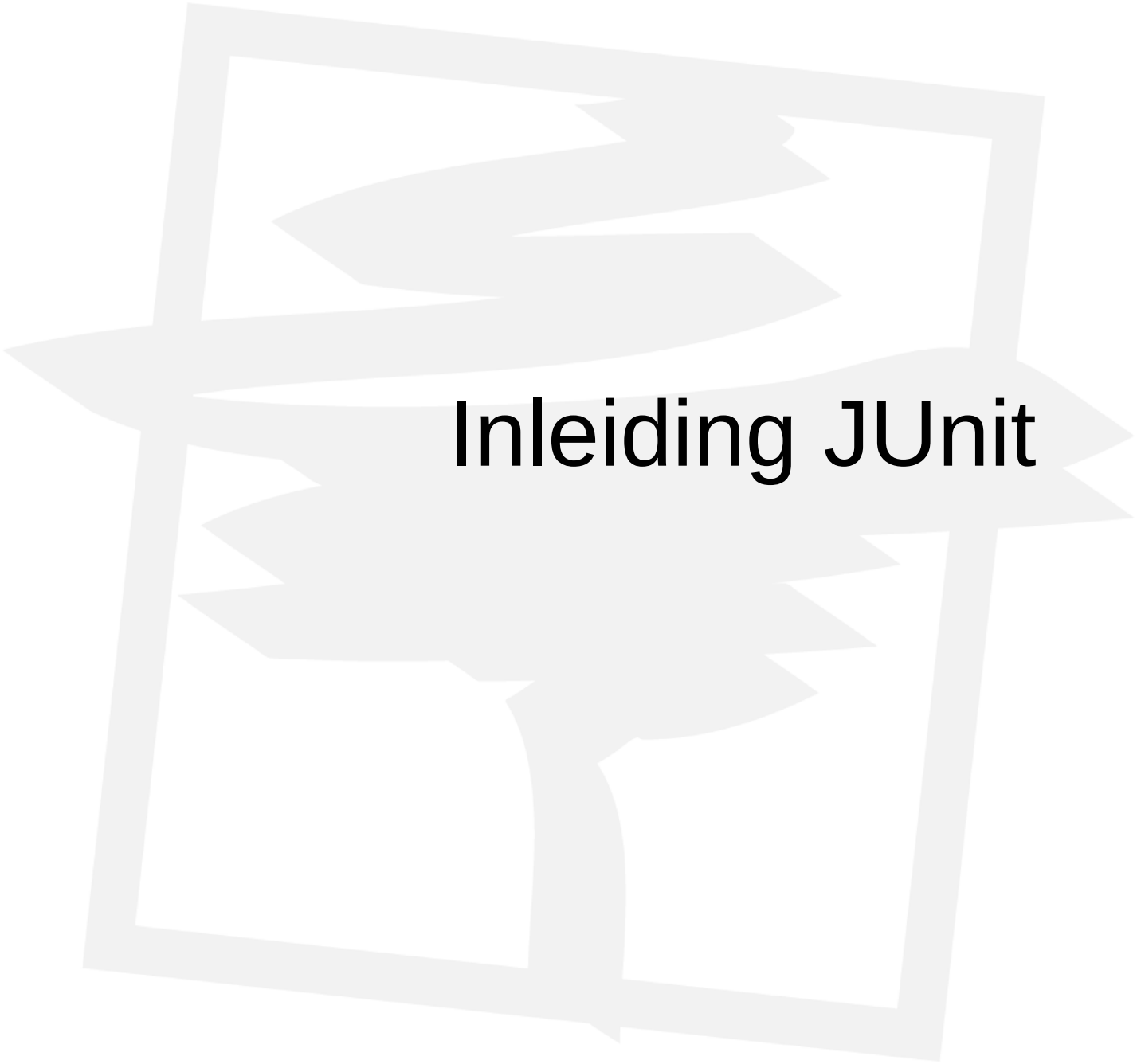


Junit

Table of contents

Inleiding JUnit



OAK 3 ACADEMY
TECHNICAL END-USER PERSONAL SKILLS

Wat is het?

Framework om software te testen.

Stabiele software maken.

- **Unit test:** afzonderlijke objecten/klassen testen
- **Functional test:** Stuk functionliteit testen (verschillende objecten/ klassen)
- **Integration test:** Gehele systeem testen

Test Driven Development

Eersten testen schrijven → Nadien klasse die aan voorwaarden voldoet.

→ Je gaat meer nadenken over je code, gaat betere code schrijven!

JUnit

Focus ligt op testen van UNITS.

→ Iedere unit (klasse) afzonderlijk testen.

Opdracht

Samen eerste test schrijven.

→Dependency toevoegen in MAVEN!

→Alle testen uitvoeren: **mvn test**

OAK 3 ACADEMY
TECHNICAL END-USER PERSONAL SKILLS

Levenscyclus

testklasse

OAK 3 ACADEMY
TECHNICAL END-USER PERSONAL SKILLS

Lifecycle

Testrunner → gaat op zoek naar @**Test**

Elke testmethode moet afzonderlijk getest kunnen worden!

Volgorde testen mag geen rol spelen!

Lifecycle

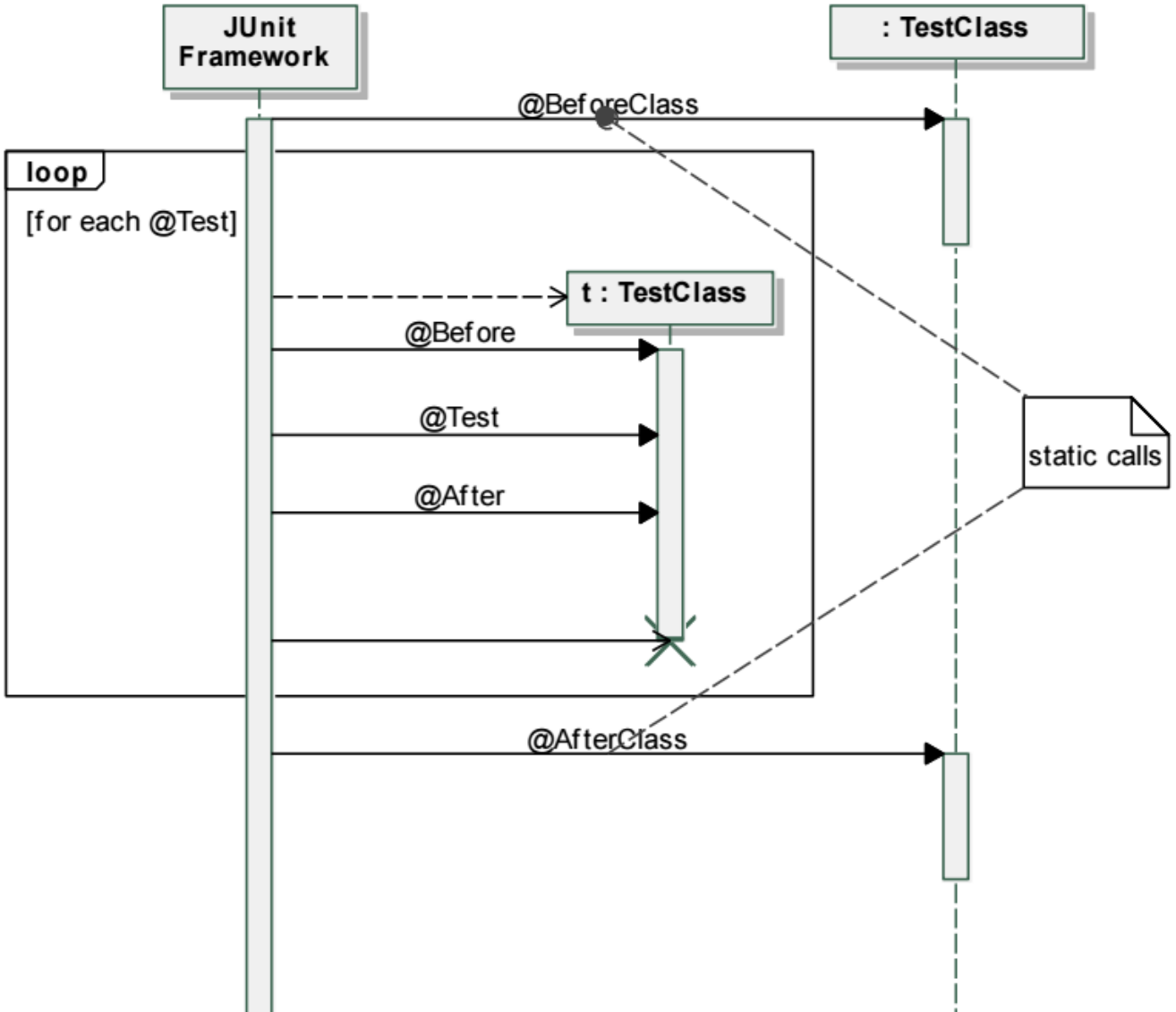
@Before: Methode wordt vooraf elke test uitgevoerd.

@After: Methode wordt achter elke test uitgevoerd.

@BeforeClass: éénmalig voor geheel van testen uitgevoerd.

@AfterClass: éénmalig na geheel van testen uitgevoerd.

Schematisch



Voorwaarden

- public
- void
- Geen argumenten
- Ze mogen exceptions gooien → test fail!
- Naam maakt niet uit

@Ignore: test negeren

Annotaties

Herhaling: Sinds java 5

Meta informatie voorzien in klassen/ interfaces.

Source: Enkel beschikbaar in broncode (Bv: @Override)

Class: Aanwezig tijdens compilatie maar niet tijdens uitvoeren

Runtime: Ook tijdens uitvoeren aanwezig. BV: JUnit annotaties

Annotaties

- Annotatie test: (interface met @ symbool voor)

```
@Retention(value=RUNTIME)
@Target(value=METHOD)
public @interface Test
```

→ Documentatie bekijken

Assert methoden

Assert klasse

Klasse met alleen static methodes

Worden gebruikt om te vergelijken.

BV: Assert.assertEquals(“Java”, “Java”);

http://junit.org/junit4/javadoc/latest/

Opdracht

Samen opdracht 5 in cursus samen maken

OAK 3 ACADEMY
TECHNICAL END-USER PERSONAL SKILLS

Fixtures

Gemeenschappelijke code samenbrengen

→@Before en @After: code die voor of na elke test gebruikt moet worden.

→@BeforeClass en @AfterClass: code die eenmalig voor testen of na testen moet uitgevoerd worden
Bv: connectie maken en sluiten

Grenzen testen

Moeilijk om alles te testen

→ Zinvol testen: bv grenswaarden, null, uitzonderingen …

OAK 3 ACADEMY
TECHNICAL END-USER PERSONAL SKILLS

Opdracht

Opdracht 6 p. 108 samen maken.

Exceptions testen

In bepaalde gevallen moeten er Exceptions gegooid worden.

→Ook testen!

→Bv: @Test(expected=IOException.class)

We verwachten dat deze test IOE gooit.

Opdracht

Opdracht 7 p. 109 samen maken

OAK 3 ACADEMY
TECHNICAL END-USER PERSONAL SKILLS

Stub objecten

Object A is afhankelijk van ander Object B.

Je wil alleen Object A testen?

→Nep Object B maken= Stub of dummy-object

→Meestal lege implementatie

→Geïsoleerd stukje testen!

Mock objecten

Stub object met functionaliteit → Bepaalde verwachtingen nabootsen.

Framework gebruiken hiervoor!

Opdracht

Opdracht 8 en 9 p. 110 – 111samen maken

→ Zie UML diagram cursus.

Mockito

Framework voor het maken van Mock objecten.

Implementatie gebeurt door framework zelf.

→ Opdracht 10 zelfstandig!

OAK 3 ACADEMY
TECHNICAL END-USER PERSONAL SKILLS

Test suites - Categoriën

Test klasse aanmaken die verschillende andere tests uitvoert.

Testen kunnen opgedeeld worden in Categoriën

→ In test suite kunnen dan alle testen in bepaalde categorie uitgevoerd worden.

Maven commando's

Alleen deze klasse testen:

mvn test –Dtest=MijnTest

Tijdens een fase testen negeren:

mvn clean install -DskipTests

Opdracht: JUnit in AutoApp

Maak van de AutoApp een Maven applicatie. Voeg JUnit toe als Dependency.

Maak voor elke functionaliteit in de auto klasse een aparte test.