



NTNU

Kunnskap for en bedre verden

Øvingsforelesning 7 - Oppgaver

TDT4100 Objektorientert programmering

Onsdag 19.02.2020



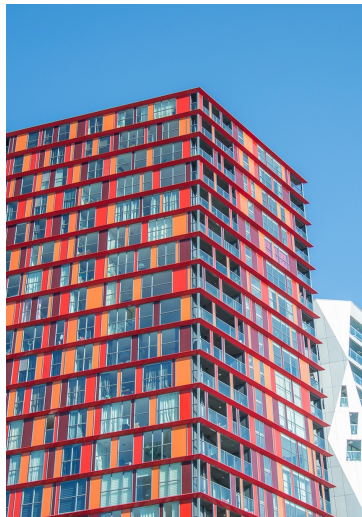


Oppgaver

I denne øvingsforelesningen skal vi lage kode for å representere en bedrift.

Bedriften skal representeres som en gruppe avdelinger, som seg igjen kan ha underavdelinger, osv. Hver avdeling kan ha et sett med ansatte tilknyttet seg.

I tillegg skal vi se på testing. Vi skal skrive noen JUnit tester for å teste koden vi skriver i øvingsforelesningen.



Pakker

Utdelt kode og oppgavene finnes i mappen foreksempel/src/of7.

`foreksempel/src/of7.kode`

Her finner dere utdelt kode og kan skrive deres egen kode.

`foreksempel/src/of7.underveis`

Her blir kode lagt inn rett etter felles gjennomgang.

`foreksempel/src/of7.testar`

Her ligger kode som kan brukes til å teste at en oppgave er løst riktig

`foreksempel/src/of7.lf`

Blir gjort tilgjengelig i etterkant og inneholder et løsningsforslag for oppgavene.

Oppgave 1

 5 min

Motivasjon

Det første vi trenger å gjøre er å kunne representere bedrifter og deres avdelinger. For å gjøre jobben vår litt enklere, vil vi representere en bedrift som en avdeling som ikke har noen av avdelinger over seg. Altså, avdelinger og bedrifter representeres av samme klasse.

Oppgavetekst

Lag en klasse **Department** og tilhørende konstruktører. **Department** skal ta vare på hvilken avdeling den tilhører (annet **Department** objekt), og skal være mulig å initialisere både med og uten å oppgi hvilken avdeling den tilhører.

Testing

Se neste oppgave.

Oppgave 2

 7 min

Motivasjon

Før vi implementerer noe mer kode, skal vi smålig begynne på testing. Dette hovedsaklig for å bli vant til hvordan vi gjør testing.

Oppgavetekst

Lag en klasse **DepartmentTest.java** og skriv en JUnit test som sjekker at avdelingstilhørighet settes riktig.

Tips: Du må nok skrive en getter-metode for feltet for avdelingstilhørighet

*Tips 2: **@Test** annotasjonen og **assertEquals** funksjonen er nyttige*

Testing

Kjør JUnit testen du nettopp har skrevet, og sjekk at den ikke feiler.



Oppgave 3

 5 min

Motivasjon

En avdeling trenger ikke bare å vite hvilken avdeling den tilhører, men også hvilke avdelinger som tilhører den selv.

Oppgavetekst

Lag en privat metode **addDepartment** i **Department**, som tar vare på den oppgitte avdelingen som en underavdeling av seg selv. Denne metoden bør kalles i konstruktøren som tar inn avdelingstilhørighet.

Testing

Bruk printing i konstruktører e.l. eller lag en metode for å hente ut hvilke avdelinger som ligger under en avdeling.



Oppgave 4

 7 min

Motivasjon

Det kan være nyttig å vite om en avdeling er under en annen.

Oppgavetekst

Skriv en metode **contains** i **Department** som tar inn en avdeling og sjekker om den er en underavdeling av avdelingen som **contains** ble kalt på. Her må man både tenke på direkte underavdelinger og underavdelinger av underavdelinger.

Testing

Se neste oppgave.

Oppgave 5

 7 min

Motivasjon

Det er lett å gjøre noe feil når man driver med rekursjon. Vi skal derfor teste funksjonaliteten vi nettopp skrev.

Oppgavetekst

Skriv en JUnit test som lager avdelinger og sjekker at **contains** fungerer ordentlig, både for direkte underavdelinger, indirekte underavdelinger og avdelinger som ikke er underavdelinger av hverandre.

*Tips: **assertTrue** og **assertFalse** funksjonen kan være nyttige*

Testing

Kjør JUnit testen du nettopp har skrevet, og sjekk at den ikke feiler.



Oppgave 6

 7 min

Motivasjon

Hvor en avdeling tilhører kan endre seg.

Oppgavetekst

Skriv en funksjon **moveTo** i **Department** som tar inn en avdeling, og flytter denne avdelingen til å ligge under den oppgitte avdelingen.

*Tips: Du kan trenge en **removeDepartment** metode.*

Testing

Kjør JUnit testen som finnes i filen `oppgave6-test.txt`.

Oppgave 7

 7 min

Motivasjon

Et viktig poeng med testing er å sjekke at alle såkalte *edge-cases* er håndtert.

Oppgavetekst

Skriv en JUnit test for å sjekke at **moveTo** ikke tillater at to avdelinger kan være underavdelinger av hverandre. Fiks eventuelle feil som oppdages.

Testing

Kjør JUnit testen du har skrevet, og sjekk at den kjører riktig.



Oppgave 8

 5 min

Motivasjon

Vi har nå implementert og testet mesteparten av funksjonaliteten rundt avdelinger og tilhørighet blant avdelinger. Vi er derfor klare til å begynne på ansatte.

Oppgavetekst

Lag en klasse **Employee** og tilhørende konstruktør. En ansatt skal vite hvilken avdeling han/hun tilhører og få dette oppgitt ved konstruksjon.

Testing

Ikke veldig relevant.



Oppgave 9

 7 min

Motivasjon

Det er ikke veldig nyttig at kun den ansatte vet hvilken avdeling han/hun tilhører.

Oppgavetekst

Utvid **Department** klassen med metoden **addEmployee** som tar vare på at en oppgitt ansatt tilhører avdelingen. Husk å kalle metoden i konstruktøren til **Employee**.

Testing

Ikke veldig relevant.



Oppgave 10

 7 min

Motivasjon

Det er nyttig å vite hvilke ansatte som tilhører en avdeling eller en av underavdelingene dens.

Oppgavetekst

Lag en metode **getEmployees** i **Department** som returnerer en liste over alle ansatte som tilhører denne avdelingen og dens underavdelinger.

Testing

Kjør JUnit testen som finnes i filen `oppgave10-test.txt`.



Oppgave 11

 7 min

Motivasjon

Ansatte burde kunne flyttes mellom avdelinger.

Oppgavetekst

Lag en metode **moveTo** i **Employee** som tar en avdeling og flytter den ansatte ditt.

*Tips: Du kan trenge en **removeEmployee** metode i **Department***

Testing

Se neste oppgave.

Oppgave 12

 7 min

Motivasjon

På samme måte som for flytting av avdelinger, er det mye som kan gå galt. Derfor ønsker vi å skrive tester som sjekker at ting fungerer som det skal.

Oppgavetekst

Lag en klasse **EmployeeTest** og skriv en test som sjekker at **moveTo** flytter ansatte riktig.

Testing

Kjør JUnit testen du nettopp skrev, og sjekk at den kjører riktig.



Oppgave 13

 5 min

Motivasjon

Ansatte kan få forfremmelser. De vil da bli flyttet til avdelingen over.

Oppgavetekst

Lag en metode **promote** i **Employee** som forfremmer den ansatte hvis mulig.

Testing

Kjør JUnit testen som finnes i filen oppgave13-test.txt