Øving6: ORM / Java Persistence

Oppgave 1
Optimistisk låsing vil si å sjekke om den den aktuelle posten du prøver å oppdatere ble oppdatert av noen andre før du begår transaksjonen. Låsingen fungerer med at transaksjonen starter med å lese av postens lås-nummer. Når transaksjonen lagres på databasen, inkrementeres lås-nummeret med en. Om to ulike transaksjoner til samme post starter samtidig vil transaksjonene starte med å lese det samme lås-nummeret. Den første transaksjonen som avsluttes blir lagret på databasen. Den andre transaksjonen vil oppdage at postens lås-nummer ikke er det samme som ble lest av i starten av prosessen. Det vil si at posten allerede har blitt oppdatert. Det resulterer i at transaksjonen nummer to blir kansellert. Det sendes også ut OptimistickLockException til klienten for å informere om den kansellerte transaksjonen.
En annen låsemetode som ikke blir spurt om i denne oppgaven er pessimistisk låsing hvor man okkuperer en eksklusiv lås før man starter transaksjonen. På den måten er man helt sikker på at din transaksjon blir gjennomført.

Oppgave 2
Kjør Task2.java for å se resultatet.

Oppgave 3
Kommenter ut følgende kodelinjer i Account.java:

```
/@Version
//private int locker;
```

Kjør Task3And4.java

Resultat: De to første transaksjonen blir gjennomført. Siden vi har kommentert ut lås-nummeret 'locker' får ingen av klientene beskjed om at de to neste transaksjonene som feilet. Det resulterer i at balansen til konto1 lagres som 500 og balansen til konto2 lagres som 1500. Dette er feil. Det riktige resultatet om transaksjonen hadde gått gjennom er konto1 = 0 og konto2 = 2000.

Oppgave 4
Sett Account.java tilbake til

```
@Version
private int
```

Kjør Task3And4.java

Resultat: Samme resultat som i oppgave3, men klientene får OptimistickLockException som tilbakemelding på de transaksjonen som ikke ble gjennomført.