Innlevering 2

PG4100 - Avansert Javaprogrammering 2

Navn: Lars Erling Westbye Dahl
Dato: 19.02.2016

Forutsetninger for å kunne teste løsningen

- En MySQL-database
- At SQL-scriptet *bokliste.sql* er blitt kjørt slik at det eksisterer en database med navn *pg4100innlevering2* og tabell *bokliste* med bokdata.
- Endre hostname, brukernavn, passord eller port i DBHandler.java (Default er: **localhost, root / root, port 3306**)

- Avhengigheter:
  ormlite-core-4.48
  mysql-connector-java-5.0
  hamcrest-core-1.3

Beskrivelse av løsning

Løsningen består av totalt 5 klasser:

Book

Bok-objekt som danner grunnlag for quiz-spørsmål. Et value-objekt med attributter som forfatternavn, tittel, isbn, antall sider og året det ble utgitt. Klassen har nødvendige mappinger for å fungere med ORM lite.

DBHandler

Henter data om bøker fra SQL-database. DBHandler benytter ORM Lite og en DAO for å konvertere tabelldata til Book-objekter.

QuizClient

Selve klient-klassen. Klienten prøver å åpne en TCP-forbindelse mot adresse og port som er definert i SERVER_ADDRESS og SERVER_PORT. Hvis tilkoblingen lykkes kjøres deretter en do-while løkke for å ta imot og sende meldinger til server. Klienten avsluttes ved å skrive 'quit' i konsoll. Har hjelpemetoder for å sende og ta imot meldinger, samt for å koble fra.

QuizServer

Når et QuizServer-objekt opprettes leser den inn Bokdata fra SQL-databasen som er definert i DBHandler og lagrer dette i en arraylist. Dersom man benytter konstruktøren med parameter må man sende inn en liste av Book-objekter fra en annen ressurs (f.eks. fra fil som benyttes i dette tilfelle til testing). Videre, etter at start()-metoden kalles opprettes det en ServerSocket som lytter på porten som er definert i SERVER_PORT (default: 8001). En while-løkke sørger for at serveren tillatter mer enn én klientforespørsel. For hver klientforespørsel oppretter serveren en ny tråd med et objekt av typen ClientHandler som tar

klientsocket og listen med Book-objekter som parametere i konstruktøren. Tråden legges til i en ExecutorService (cachedThreadPool) og startes.
Serveren har også implementert start() og stop() metoder. Når stop()-metoden blir kalt på lukkes serversocketen og alle ClientHandler-tråder blir forsøkt avbrutt med shutdownNow()-metoden i ExecutorService.

ClientHandler
Objektet som setter i gang kommunikasjon med klienten(e). Klassen implementerer Runnable-interfacet for å kunne bli kjørt som en egen tråd. I klassens run()-metode ligger logikken for hvordan kommunikasjonen med klienten foregår. I tillegg har den hjelpemetoder for å sende og motta meldinger, hente et vilkårlig spørsmål fra listen med quizdata og en metode for å undersøke om klientens svar er korrekt. Da klientene kun leser data fra listen med bøker, men ikke endrer dem, var det ikke nødvendig å implementere locks.

Kommentarer til eget resultat

- Testingen av løsningen er veldig mangelfull. Fant heller ingen gode måter å teste sockets på etter utallige google-søk.
- Kunne utvidet løsningen med en score-funksjon.
- Regexen som benyttes for å sjekke om svar fra klient er korrekt er ikke helt robust. Den tolker ikke punktum riktig. Et eksempel er at den gir feil på bl.a. *J.R.R. Tolkien*.
- Ellers fornøyd med løsningen

Spørsmål for å komme videre

Igjen så opplever jeg at testingen blir den største utfordringen. Ønsker eksempler på ulike måter å teste sockets på, eller hvertfall noen tips og triks som kan hjelpe meg i gang.