INF142
Obligatorisk oppgave 1
2018

Dette er den *første* av *to* obligatoriske oppgaver i INF142 i løpet av vårsemesteret 2018. Studenter som ikke leverer eller som ikke får innleveringen godkjent, får ikke ta eksamen. To og to (men ikke fler enn to) studenter kan jobbe sammen og kan produsere en felles oppgave. I så fall skal begge studenter levere inn hver sin kopi av besvarelsen, som skal ha en eksplisitt henvisning til den andre studenten. Ut over to er *detaljert* samarbeid ikke tillatt, og *kopiering* fra andre studenter, fra nettet eller fra andre kilder er selvfølgelig juks og fanteri og forbudt. Hver av innleveringene dine blir evaluert med en poengsum: Denne kan gi høyst **150** poeng. Endelig eksamen kan gi høyst 750 poeng, og samlet poengsum (to innleveringer + eksamen) bestemmer endelig karakter.
Vær oppmerksom på at du må ha fått begge innleveringene godkjent for å få ta eksamen. Krav til godkjent for denne innleveringsoppgaven er minst 60 poeng, det vil si 40% av maksimum. Dersom innleveringen ikke blir godkjent får du likevel muligheten til å levere inn igjen, men du vil uansett beholde poengene fra første innlevering av oppgaven.

Frist: Fredag 23. mars kl. 15.00.

Beskrivelse

Dere skal implementere en enkel web-proxy-server (WPS) og en klient (K) i Java.

- WPS skal bruke TCP-socketer (se forelesningsnotater for 9. februar) til å hente *header*informasjon med **http** fra en *webserver* på en vilkårlig *vertsmaskin*, angitt ved en tegnstreng som WPS får fra klienten K. K skal også i tillegg kunne beskrive et filnavn/*stinavn* som skal hentes på webserveren. Hvis K ikke oppgir *stinavn* skal WPS bruke default *stinavn* «/». Deretter skal WPS sende svaret fra webserveren tilbake til klienten K, eventuelt en forklarende feilmelding hvis noe går galt.
- K skal kommunisere med WPS gjennom en egen applikasjonsprotokoll som dere skal beskrive som en egen deloppgave. Denne protokollen skal dere implementere gjennom at K og WPS kommuniserer ved hjelp av UDP/datagramsocketer.

Oppgaver (gjør 1, 2, og 3)

1. (Max 50 poeng) Beskriv applikasjonsprotokollen mellom K og WPS. K skal sende *vertsnavn* og eventuelt *stinavn* til WPS. Beskriv spesielt hvordan protokollen håndterer situasjonen med at det er valgfritt for K å sende *stinavn*.
2. (Max 50 poeng) Implementér klienten K i Java. Kommunikasjon mellom K og WPS skal foregå ved hjelp av UDP. K skal skrive ut (på standard utputt) alle innkommende meldinger fra WPS.
3. (Max 50 poeng) Implementér WPS i Java. (Det finnes forskjellige mekanismer for kommunikasjon mellom en klient og en http-server, men dere skal eksplisitt bruke utveksling av http-meldinger over TCP-socketer.

Du kan se bort fra klient/tjener-autentisering og andre sikkerhetsrelaterte aspekter.

Teknisk hjelp og hint:

- Se
  https://docs.oracle.com/javase/8/docs/api/java/net/ServerSocket.html
  https://docs.oracle.com/javase/8/docs/api/java/net/Socket.html
  https://docs.oracle.com/javase/8/docs/api/java/net/DatagramSocket.html

  eller forelesningsnotater for 9. februar.
- For http: se https://tools.ietf.org/, og blant annet RFC-ene 2616, 7230,….
- Oppretting av Socketer er operasjoner som kan gå feil. Hvis du ber om et portnummer som er i bruk vil du for eksempel få en feil. Bruk *try/catch* for å fange opp slike feil.
- Husk å *close()* alle TCP-forbindelser når du er ferdig med å bruke dem.
- Det er morsomst å kjøre K og WPS på forskjellige maskiner. Hvis du ikke har tilgang på flere maskiner, eller dersom du har tilgang på flere maskiner men kommunikasjonen stoppes av brannmurer eller andre tekniske forhold, eller for testformål, kan du kjøre begge prosesser på samme maskin. Du kan da bruke IP-adressen «127.0.0.1» som navn på maskinen, når K skal kommunisere med WPS.

- ***Tilleggsoppgave – dette er ikke et krav og gir ikke poeng, men kan være lærerikt*:**
  Du kan få lyst til å lage en *multitrådet* versjon av WPS som kan håndtere flere samtidige klienter. En *tråd* er en utføringstråd (thread of execution) i Java. Java Virtual Machine tillater en applikasjon å starte flere samtidige tråder som utføres parallelt. Dette er nødvendig når en applikasjon skal utføre flere operasjoner samtidig. Husk at en lese-operasjon i Java (enten det gjelder lesing fra tastatur eller fra en socket) innebærer at tråden som utfører denne operasjonen må vente passivt. Ved å ha flere tråder, kan man ha en tråd som er ansvarlig for å lese samtidig som andre tråder kan utføre nyttig arbeid. Se

  https://docs.oracle.com/javase/8/docs/api/java/lang/Thread.html

  Se også

  https://docs.oracle.com/javase/8/docs/api/java/lang/Runnable.html

  En tråd startes ved å bruke metoden *start()*. En tråd må ha metoden *run()*, og operasjonene tråden skal utføre defineres i denne metoden. En tråd er *død* når den er ferdig med sin *run()*-metode.

Eksempel:

ClientProcessor.java

```
public class ClientProcessor implements
Runnable{

   public ClientProcessor (){//constructor

     } // constructor()

   public void run(){// different executions

     } // run()

}
```

Hovedprogram.java:

```
public class Hovedprogram {

      public static void main(String[] args) {

          Thread thread =

               new Thread(ClientProcessor ());

          thread.start();

        } // main()

}
```

Teknisk hjelp og hint:
Oppgavene skal leveres inn som en zip-fil som inneholder minst to javafiler K.java og WPS.java, samt en pdf-fil som beskriver protokollen mellom K og WPS (oppgave 1) og andre detaljer dere ønsker å forklare. Filen skal lastes opp på Studentportalen (mitt.uib.no) og skal ha navnet

INF142-Oblig1-*brukernavn*.pdf

der du erstatter *brukernavn* med *ditt brukernavn*.

Maksimum to studenter kan samarbeide og levere inn identiske oppgaver. **I så fall skal dere likevel hver for dere levere inn oppgavene på studentportalen, med en referanse til den du har samarbeidet med.**

Frist: Fredag 23. mars kl. 15.00.