# Tentamen
# Nätverksprogrammering
# Del 2

**2013–06–03, 8.00–13.00**

Tillåtna hjälpmedel för denna del av tentamen:

- Java snabbreferens.
- Kursboken: Java Network Programming av Eliotte Rusty Harold.
- Valfri lärobok i Java.
- Utskrift av OH-bilder från föreläsningarna.

Denna tentamen i kursen Nätverksprogrammering består av två delar – en del som innehåller frågor av teoretisk/principiell/utredande karaktär och en del som innehåller praktiska programmeringsuppgifter. Detta är del 2. Den ska du ha erhållit tillsammans med ett färgat tentamensomslag när du lämnade in din lösning på del 1 av tentamen.

För godkänt betyg på tentamen krävs sammanlagt minst 20 poäng på tentamen, varav minst 8 poäng på vardera deltentamen. För högre betyg krävs naturligtvis mer, så gör så många uppgifter du kan.

---

## 1. Strömmande video

I kursen Realtidsprogrammering, som ges vid Institutionen för datavetenskap, förekommer ett projekt i vilket man konstruerar ett nätverksbaserat videoövervakningssystem. Bilder överförs i ”realtid” från ett antal kameror kopplade till nätverksanslutna datorer (kallade ”kameraservrar”) spridda i byggnaden som ska övervakas till en, eller flera, central(a) övervakningsstation(er) (kallad(e) ”övervakningsklient(er)”) där de visas på en skärm. Videoströmmen från kamerorna utgörs av en sekvens av individuella JPEG-bilder (stillbilder). Din uppgift är att hjälpa studenterna på kursen att skriva koden som behövs för att överföra bilderna via nätverket.

### Övervakningsklienten

På övervakningsklienten, dvs den dator som tar emot bilderna, används en klass enligt nedanstående beskrivning:

```
class VideoReceiver {
    /** Create an object which allows the client to connect to a camera server
        and request images at will. The address to the server is given as the
        argument to the constructor. Returns as soon as the object is initialized. */
    public VideoReceiver(String address) throws IOException;
    /** Send a request to the server to send the next image over the network.
        Then, the method blocks until the image arrives. The image is returned
        as a byte vector with a length (<30000 bytes) exactly fitting the size
        of the JPEG image. */
    public byte[] receive() throws IOException;
     /** Disconnects from the server. */
    public void close() throws IOException;
}
```

---

Som antyds av beskrivningen så kastas alla eventuella IOException bara vidare upp i anropskedjan om något går fel vid anrop av metoderna.

**Kameraservern**
På serversidan behövs en motsvarande klass för att ta emot uppkopplingar från övervakningsklienter och för att skicka bilder över nätverket:

```
class VideoTransmitter {
    /** Create an object capable of sending images to supervision clients, if
        connected. Throws an IOException if there is problems initializing the
        object due to network problems. Returns as soon as the object is initialized. */
    public VideoTransmitter() throws IOException;
    /** Informs the object that a new image is available for sending to the
        supervision clients if they should so request. */
    public void send(byte[] image);
}
```

Det är inte säkert att övervakningsklienten/nätverket hinner med att ta emot/förmedla bilder i samma takt som kameran genererar dem. För att få bra flyt i videosekvensen och inte överbelasta nätverket måste vi därför kasta bilder som vi inte hinner med att ta hand om. Därför ska bilder bara skickas ut på nätverket efter en direkt begäran från klienten. Klienten skickar inte en ny begäran förrän den har tagit emot och behandlat den gamla bilden. Vi vill å andra sidan inte heller skicka samma bild två gånger. Om klienten begär bilder snabbare än vad kameran kan generera vill man i stället invänta nästa bild innan denna skickas över till klienten.

Klassen `VideoTransmitter` fungerar därför enligt följande: Så fort ett objekt av klassen skapats ska den vara beredd att acceptera uppkopplingar från övervakningsklienter. Den ska kunna hantera flera uppkopplingar samtidigt. Övriga delar av servern, som du inte ska skriva koden för, kommer att kontinuerligt ta nya bilder från kameran (25 Hz) och för var och en av dessa bilder anropa operationen `send()` – även innan dess en klient kopplat upp sig. Operationen `send()` får aldrig blockera (i ett `wait()` eller p.g.a. I/O) utan ska returnera med minimal fördröjning. Annars hinner vi kanske inte med att ta hand om alla bilder från kameran.

Bilder skickas däremot bara ut på nätverket om en klient verkligen är uppkopplad och bara då klienten faktisk har begärt att en ny bild ska skickas (genom anrop av `VideoReceiver.receive()`). När en begäran om en ny bild anländer ska den senast från kameran erhållna bilden skickas (parametern i senaste anropet av `send()`). Om vi har skickat en bild och vi inte erhåller en ny bild från kameran innan nästa bildbegäran från klienten tas emot ska servern vänta med svaret till klienten tills `send()` anropats på nytt och en ny bild finns tillgänglig.

Om det uppstår nätverksfel av något slag på serversidan, eller om klienten kopplar ned förbindelsen till servern, ska servern sluta sända bilder till klienten (koppla ned anslutningen).

Observera: JPEG-formatet som bilderna lagras i utnyttjar en form av datakomprimering vars effektivitet beror på bildens innehåll. Storleken på de färdiga bilderna kan därför variera från bild till bild, men är dock aldrig större än 30000 bytes. Bilderna lagras därför i byte-vektorer av varierande längd (beroende på bildstorlek). De enskilda byte-elementen kan ha godtyckliga värden, dvs alla tillåtna värden för en byte är möjliga. De byte-vektorer som returneras av operationen `receive()` ska vara lika stora som de byte-vektorer som kom in som parametrar till operationen `send()`.

Implementera de två klasserna `VideoReceiver` och `VideoTransmitter` ovan tillsammans med eventuella hjälpklasser/trådar du kan tänkas vilja införa [1]. Använd TCP som överföringsmekanism vilket garanterar att information inte försvinner på nätverket. Eftersom det tar för lång tid att öppna en ny TCP-förbindelse för varje bild måste vi låta förbindelsen vara öppen så länge klienten är ansluten till servern och skicka alla bilderna över samma ström.

*(20p)*

*Slut – Glad Sommar!*

[1] Ledning: Skapa/starta eventuella hjälpobjekt/hjälptrådar i två givna klassernas konstruktorer. Lägg dock ingen serverfunktionalitet i konstruktorerna. I enlighet med kodkommentarerna ovan ska ett anrop av konstruktorerna bara initialisera objekten.