INF1010 - Puslegruppa

Serverprogrammering

Simen Heggestøyl – simenheg@ifi.uio.no
Sigmund Hansen – sigmunha@student.uio.no

31. mars 2011

ServerSocket

En spesiell type stikk som lytter på en port:

- Er bundet til en bestemt port.
- Man bruker ikke strømmer til lesing og skriving
- Man kaller accept som
  - Returnerer en socket for å snakke med andre maskiner
  - Kaster et unntak i alle andre tilfeller

Lytte på en port

Du kan lage en ServerSocket som lytter på en port slik:

```
try {
    ServerSocket srv = new ServerSocket(1337);
} catch (IOException e) {
    e.printStackTrace();
}
```

Listing 1: Lage et bundet stikk

Lytte på en port

Om du ikke angir en port i kallet til konstruktøren, lager du et ubundet stikk. Da må du bruke bind for å binde det til en port (og adresse).

```
try {
    ServerSocket srv = new ServerSocket();
    srv.bind(new InetSocketAddress(null, 1337));
} catch (IOException e) {
    e.printStackTrace();
}
```

Listing 2: Binde et stikk senere

Motta forbindelser

En ServerSocket lar oss lage andre stikk for å kommunisere med klienter som prøver å koble seg til serveren:

```
while (true) {
    try {
        Socket sock = srv.accept();
        // Create a thread to work with sock here
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Listing 3: Motta forbindelser

Avbryte venting

Vi vil ikke alltid vente for alltid på at accept skal returnere. Da kan vi bruke setSoTimeout kombinert med SocketTimeoutException:

```
// Sets timeout on accept in milliseconds
ServerSocket srv = new ServerSocket(port);
srv.setSoTimeout(1000);
...
try {
    srv.accept();
} catch (SocketTimeoutException ste) {
    // We don't really need to do anything here
}
```

Listing 4: Timeout

Tråder

Tråder lar deg gjøre flere ting samtidig. Man kan lage underklasser av Thread som har en egen run-metode:

```
class MyThread extends Thread {
    public void run() {
        // Do stuff
    }
}
...
new MyThread().start();
```

Listing 5: Thread

Tråder - Runnable

Eller man kan lage klasser som implementerer Runnable og la tråden kjøre dem:

```
class MyRunnable implements Runnable {
    public void run() {
        // Do stuff
    }
}
...
new Thread(new MyRunnable()).start();
```

Listing 6: Runnable

Farer ved bruk av tråder

- Flere tråder endrer samme data samtidig
  - To tråder prøver å utføre setningen: i++;
  - Tråd én henter verdien til i som er 10
  - Tråd to henter verdien til i som er 10
  - Tråd én øker verdien til i til 11
  - Tråd to "øker" verdien til i til 11
  - i skulle ha vært 12, men er 11
- Dette fikses ved hjelp av låser

Farer ved bruk av tråder

- Tråder som vil jobbe med et låst objekt, må vente
- Tråder kan ende opp med å vente på hverandres låste objekter - deadlocks
- Behov for flere samtidige låser i én tråd kan føre til dette
- Vi har heldigvis ikke behov for så mange låser i vårt program

Synkroniserte kodeblokker

Synkroniserte kodeblokker lager en lås for ett objekt:

```
synchronized void doStuff() {
    // this is locked
    synchronized(someOtherObject) {
        // someOtherObject and this is locked
    }
    // this is still locked
}
```

Listing 7: Synkroniserte kodeblokker

Likevel kan man jobbe med objektene i ikke-synkroniserte kodeblokker. Så man bør synkronisere når flere tråder jobber med samme objekter, men pass deg for deadlocks.

Tilkoblingstråder

Vi vil starte en tråd for hver forbindelse:

- Hver tråd prøver å lese fra sitt stikk
- Behandle data når den får noe å lese
- Flere tråder kan ikke sende beskjeder til de andre brukerne samtidig (synkronisert kode)

Tilkoblingstråder

```
class ClientConnection implements Runnable {
    ...
    public void run() {
        while (connected) {
            // read packets from the client
        }
    }
}
...
Socket s = srv.accept();
new Thread(new ClientConnection(s)).start();
```

Listing 8: Klientforbindelser

Kan vi klare oss med én tråd?

Dette er mulig så lenge vi ikke jobber med blokkerende I/O.
All vanlig I/O blokker i Java.
Men i Java 4 kom New I/O-pakken (java.nio). Denne gir tilgang til ikke-blokkerende I/O.
Her er en liten tutorial (men denne bruker litt tråder):
http://rox-xmlrpc.sourceforge.net/niotut/