Tentamen
Nätverksprogrammering
Del 2

2018–03–14, 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.

---

1. **Distribuerad high-score**

I ett datorspel skrivet i Java vill man ha en gemensam high-score för alla som spelar spelet på olika platser i världen. Tillverkaren av spelet har därför satt upp en server som håller reda på det högsta kända high-score-värdet och namnet på spelaren som satte rekordet. Spelet är uppkopplat mot servern för att hålla sig uppdaterat på de senaste globala resultatet och om spelaren skulle lyckas få fler poäng är det, så skickar spelet över information om det nya resultatet till servern. Sevens tillstånd uppdateras då (om det fortfarande är bästa resultatet; nya poäng kan ju ha kommit in från annat håll under tiden). Vid start av spelet kan du anta att bästa resultatet på servern är 0 poäng. Javakoden för spelet innehåller bland annat följande kodsnuttar som har hand om kommunikationen med servern:

```
public class HiScore {
    public int score;
    public String name;

    HiScore(int score, String name){
        this.score = score;
        this.name = name;
    }
}

public class HiScoreCommunication {
    ...

    public HiScoreCommunication() {
```

```
        // Establishes a connection to the game server
        // The address to the server is assumed to be "scoreserver.megagames.se"
            ...
        }

    public void sendNewScore(HiScore score) {
        // Sends a new high-score to the server
            ...
    }

    public HiScore awaitNewHiScore(int oldScore) {
        // Opens a connection to the server and waits until someone, possibly
        // we ourselves (in another thread), sends in a HiScore that is better
        // than the value of the parameter oldScore.
        // More formally: This method blocks until the high score on the
        // server is better than oldScore. The new, updated,
        // high score is returned.
            ...
    }
}
```

Spelet använder sig av klassen `HiScoreCommunication` för att kommunicera med tillverkarens server. När spelprogrammet tror att ett nytt bästaresultat har uppnåtts skickar den poängen och namnet på spelaren till servern genom att anropa metoden `sendNewScore(HiScore score)`. Metoden `awaitNewHiScore(int oldScore)` används för att vänta på att det centrala high-score-värdet ändras (relativt det vi hämtade ner senast och som anges i parametern oldScore).

Implementera klassen `HiScoreCommunication` ovan tillsammans med ett serverprogram, inklusive main-metoden, i Java som fungerar ihop med spelprogrammet beskrivet ovan. Implementera även eventuella andra ytterligare klasser/hjälpmetoder som kan tänkas behövas på klienten och servern för att din lösning ska fungera.

(20p)

Slut!