

Lab 7 – "Disk Net Chat"

Designmönster med C++

Syfte: Tillämpning av DP Proxy och Observer

Lab 7 – "Disk Net Chat"

Den här uppgiften bygger på en mer komplex miljö än tidigare uppgifter. Två olika program, ChatServer och ChatClient ska samverka. Ett godtyckligt antal ChatClients kan ansluta till en ChatServer.

ChatServer tar emot meddelanden, som implementeras av klassen Message, från en ChatClient och skickar dem vidare till alla andra anslutna ChatClients. Alla processer ska köras på samma maskin och för att de ska kunna kommunicera trots att de exekverar i olika processer med sina egna adressrymder finns ett enkelt "Hard disc net" implementerat som emulerar ett nätverk på hårddisken. Kommunikationen sker genom att skriva och läsa 'mailboxes', filer med extension mb.

Adresser i detta nätverk utgörs alltså av sökväg+filnamn och implementeras av klassen HDaddress ('Hard Disc Address'), deriverad från Address.

Klienter skapar en förbindelse till servern genom en HDserverConnection och servern skapar en förbindelse till varje klient genom en HDclientConnection. Båda dessa klasser är deriverade från klassen Connection.

Alla klienter skickar meddelanden till servern genom att skriva till serverns mailbox, serv.mb, i serverns bibliotek. Alla klienter öppnar serverns serv.mb i append mode och kan alltså addera nya meddelanden till servern. Varje klient öppnar i sitt eget bibliotek en lokal mailbox med ett namn som är unikt för den aktuella exekveringen, t.ex 4352.mb, som servern kan skicka meddelanden till.

För att klienten ska kunna både läsa och skriva asynkront utan att blockeras startas en ny tråd som "pollar" den lokala mailboxen. Om inget finns att läsa så sover tråden några millisekunder före nästa försök till läsning. Under tiden kan man alltså skriva nya meddelanden.

Kärnan i klienten utgörs av klassen Client och i servern utgörs själva serverfunktionen av klassen Server. Dessa klasser ska samarbeta enligt ett Observer-pattern där Server agerar Subject och Client är Observer (klassen ChatObserver).

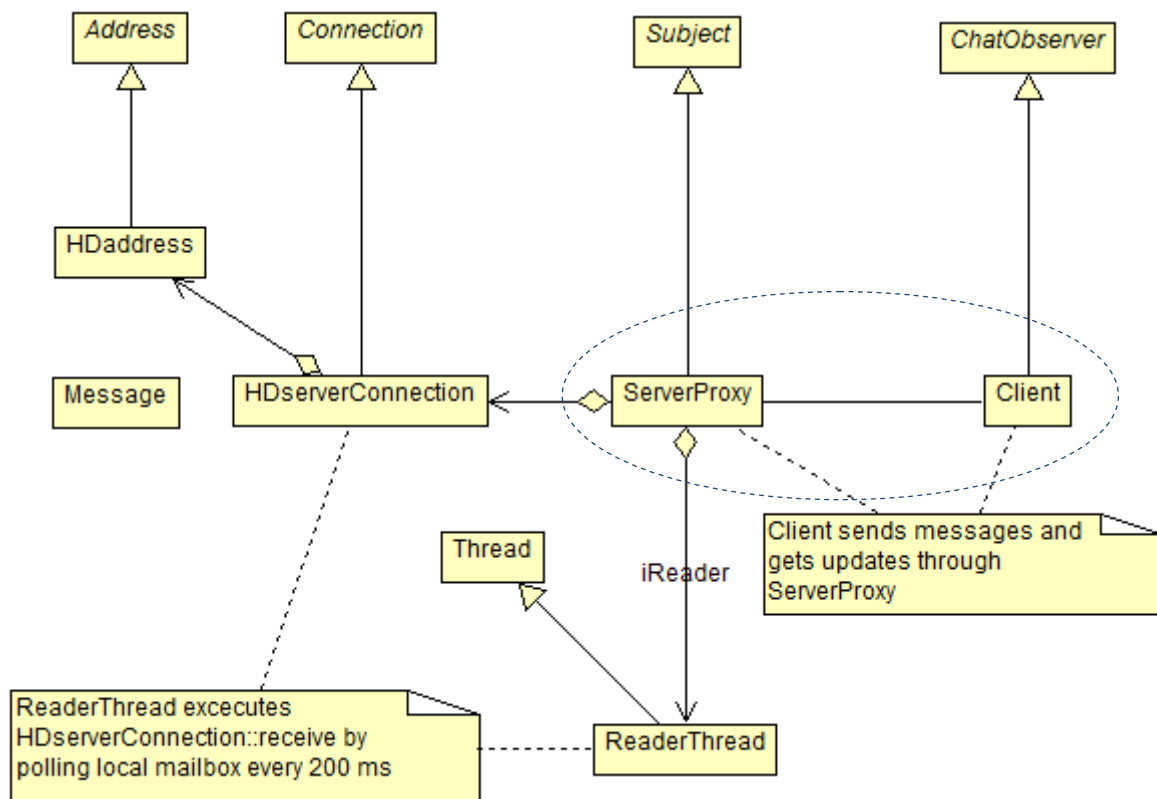
Vanligen använder man ett Observer-pattern inom samma process men här ska det implementeras mellan olika processer. Server och flera Clients ska alltså arbeta som Subject/Observer ovetande om och oberoende av att de exekverar i olika processer och därigenom i olika adressrymder.

För att detta ska bli möjligt kan vi använda oss av 'remote proxies':

- På klientsidan kommunicerar Client inte direkt med servern utan med en proxy, (klassen ServerProxy), som tar Server's plats. ServerProxy använder en ServerConnection för kommunikationen med servern.
- På serversidan kommunicerar Server med en ClientProxy som tar en Clients's plats som mottagare av nya meddelanden. Klassen ClientListener ansvarar för att läsa serverns mailbox. ClientListener hanterar attach/detach från klienter och skapar/tar bort ClientProxy-objekten. ClientListener skapar alltså en ClientProxy för varje klient som ansluter sig, och tar bort objektet när klienten lämnar chatten. ClientProxy får information om adressen till motsvarande ChatClient's lokala mailbox från ClientListener och skapar förbindelse som behövs för att skicka meddelanden.
- Förutom klienterna som är 'remote observers' har servern en lokal observer, en ChatLogger som loggar alla meddelanden till filen ChatServer.log.

ServerProxy på klientsidan och ClientListener på serversidan implementerar tillsammans ett protokoll för att hantera Attach/Detach och Messages från klienter. ChatClient är fullständigt implementerad i de medföljande filerna.

Klassdiagram:

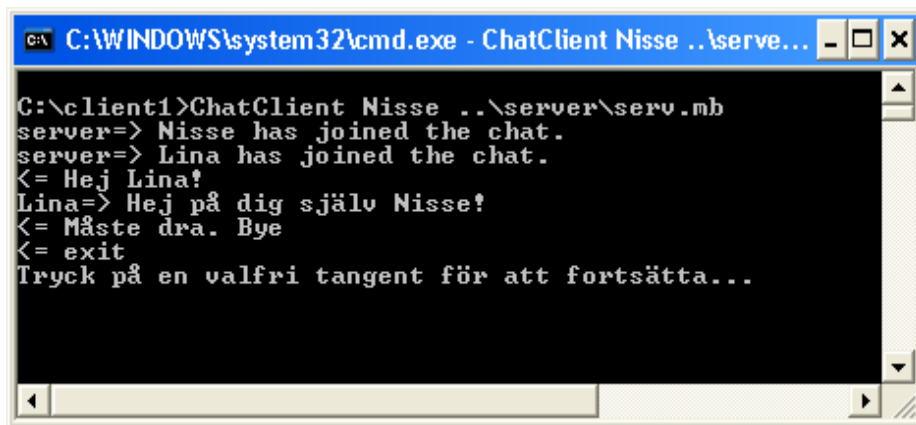


ChatClient startas med två parametrar, namnet på chattaren och fullständig sökväg till serverns mailbox, t.ex.

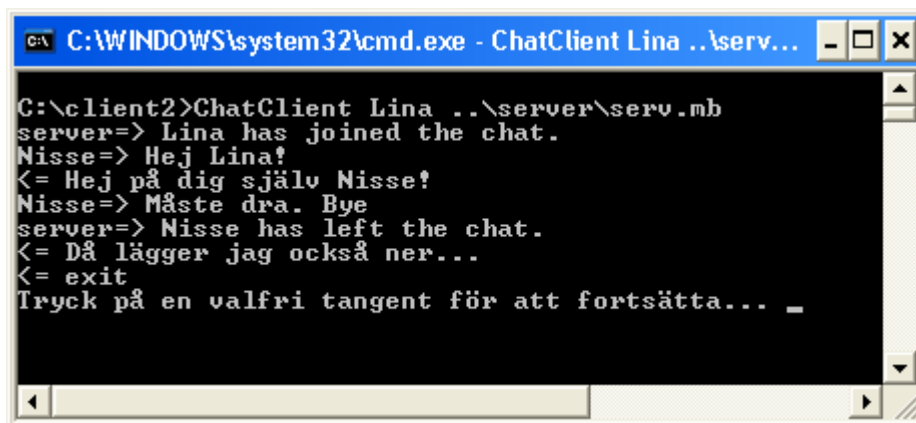
```
C:\Client>ChatClient Nisse ..\server\server.mb
```

Om allt fungerar kommer meddelandet att Nisse är ansluten och en prompt för att skriva meddelanden. Klienten avslutas med 'exit'.

Ett körningsexempel med två klienter:



```
C:\client1>ChatClient Nisse ..\server\serv.mb
server=> Nisse has joined the chat.
server=> Lina has joined the chat.
<= Hej Lina!
Lina=> Hej på dig själv Nisse!
<= Måste dra. Bye
<= exit
Tryck på en valfri tangent för att fortsätta...
```



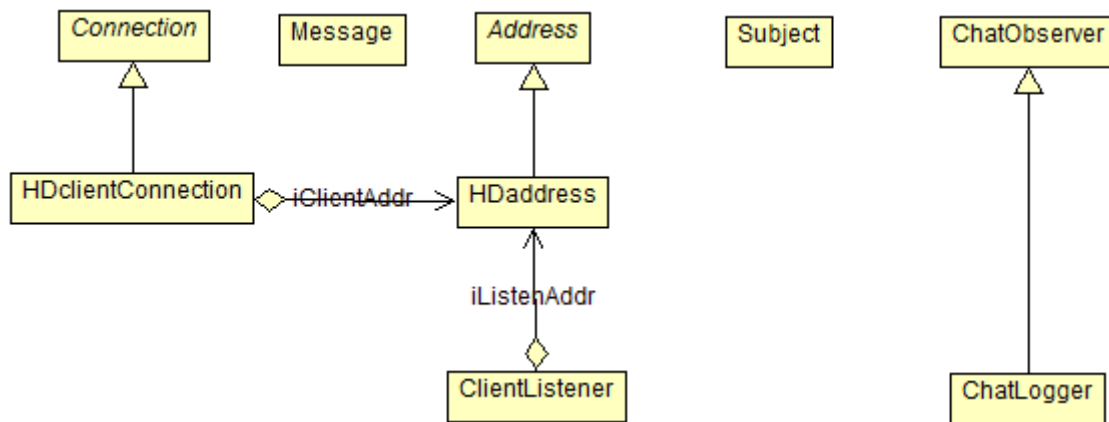
```
C:\client2>ChatClient Lina ..\server\serv.mb
server=> Lina has joined the chat.
Nisse=> Hej Lina!
<= Hej på dig själv Nisse!
Nisse=> Måste dra. Bye
server=> Nisse has left the chat.
<= Då lägger jag också ner...
<= exit
Tryck på en valfri tangent för att fortsätta...
```

Och nu, för att citera Kung Lois i Djungelboken, *till Din del av vårt avtal...*

Uppgift

I följande klassdiagram över ChatServer fattas klasserna Server och ClientProxy och deras relationer till resten av klasserna. Din uppgift är att skriva de klasserna så att servern fungerar enligt beskrivningen här ovan utan att ändra på resten av

koden. Klassen Server ska implementera ett Observer-mönster och CleintProxy ska implementera ett Proxy-mönster.



ChatClient startar en ny tråd för att detektera nya meddelanden. Om du kompilar med g++ i Linux måste du ha
-std=c++11 -pthread
på kommandoraden när du bygger ChatClient

Redovisning

Zippad fil med den kompletta ChatServer. De klasser du skrivit ska vara ordentligt kommenterade.