Laboration – Old school chat

Denna laboration skall lösas individuellt

Innehållsförteckning

Bakgrund

OSC.org har påbörjat utvecklingen emot en nygammal chat. Idén är att locka in den äldre generationen som känner sig ifrån sprungna med alla chatlösningar som presenteras i webbläsaren och saknar den gamla tiden (IRC, MSN-messanger m.fl.).

Därför utvecklas för närvarande en desktopchat i Java. Arbetet är i full gång och en fungerande version 1.0 finns på plats. Men, redan nu har utvecklarna som byggt applikationen kört fast och börjat fundera på några implementationsval samt hur de skall gå vidare.

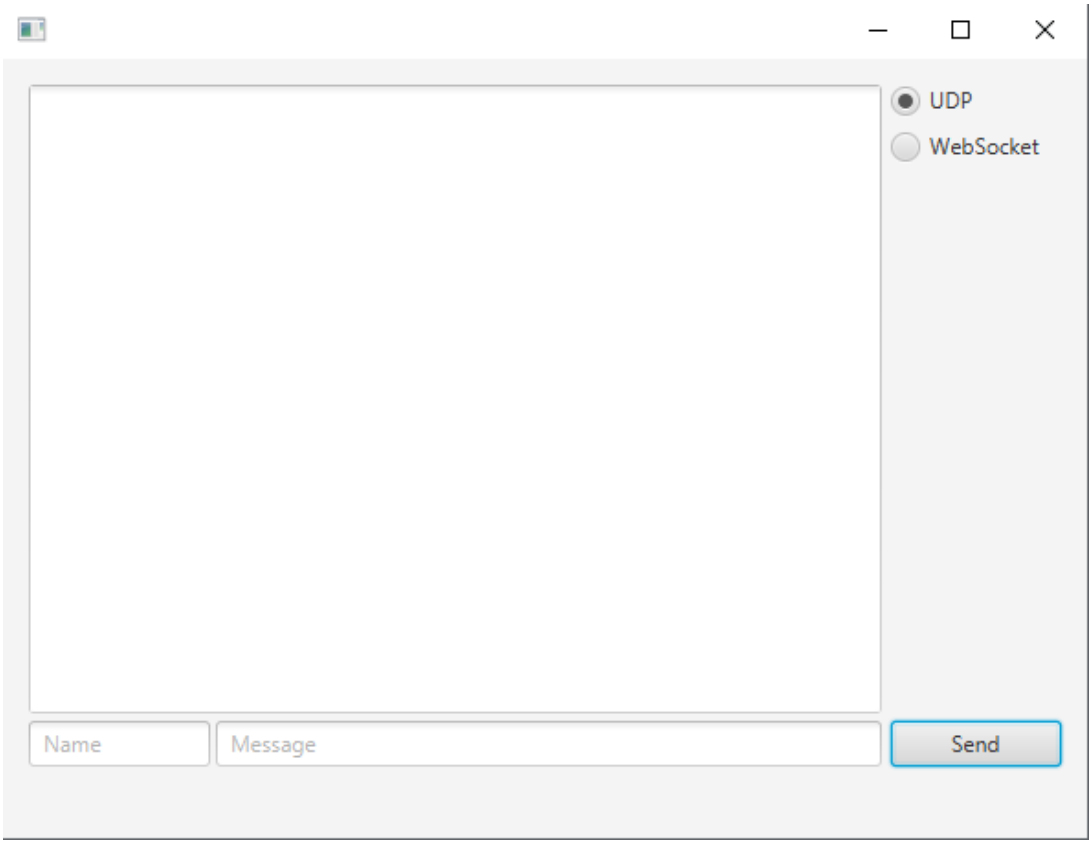
Det är här du kommer in i bilden, ditt uppdrag är att hjälpa utvecklarna genom att göra om och ge förslag på hur samma funktionalitet skulle kunna implementeras annorlunda för att skapa robusthet i lösningen.

Projektet

Kod att utgå ifrån hittas här: https://bitbucket.org/margob/exam_chat_a-d/src/master/
(git clone *https://margob@bitbucket.org/margob/exam_chat_a-d.git*)

Att göra

När du startar applikationen möts du av följande:



I chattens högra hörn finns det möjlighet att välja kommunikationsprotokoll, eller så är tanken och ditt första uppdrag är implementera så detta fungerar.

Deluppgift A (G)

Kod finns för att kommunicera antingen via UDP i (klassen **UDPChatCommunicator**) eller via Web Socket (klassen **WebSocketCommunicator**)**.** Men ingen logik finns för att skifta vilken som används när användaren av chatten ändrar val i applikationen.

Det är möjligt för dig att ändra vilken kommunikatör som används i koden genom att ändra vilken rad som är bort kommenterad (se rad 33–38 i MainWindowController.java).

```
//---------- Deluppgift A test:
//Change what line is commented to change communicator:
//private WebSocketCommunicator _communicator = new WebSocketCommunicator(this);
private UDPChatCommunicator _communicator = new UDPChatCommunicator(this);
//----------
```

Uppdraget i deluppgift A är att implementera så det i applikationen är möjligt att dynamiskt ändra vilken kommunikatör som används genom att klicka i radioknapparna.

När du implementerar din lösning skall minst ett vedertaget designmönster användas. Till kodlösningen skall även en .pdf lämnas in som besvarar samt motiverar:

- Varför valde du den aktuella lösningen?
- Vilket eller vilka designmönster använde du?
- Föra ett resonemang kring hur din implementation förhåller sig till minst en vedertagen designprincip.

Deluppgift B (G)

Förnärvarande föreligger det en stark koppling emellan kontrollerklassen (MainWindowController) och kommunikationsklasserna (UDPChat/WebsocketCommunicator). Kommunikationsklasserna är således låsta till att endast kunna kommunicera med MainWindowController klassen. Detta är något vi önskar luckra upp genom att implementera en Observer pattern lösning **(*OBS Javas native observer lösning får ej nyttjas.*)**.

Ditt uppdrag i deluppgift B är alltså att implementera detta i applikationen. Till din kodlösning skall även en .pdf lämnas in som besvarar samt motiverar:

- Vilka vedertagna designprinciper styrks via implementationen av Observer mönstret?

Deluppgift C (VG)

Programmet behöver utökas med loggnings funktionalitet. I denna första version bör samtliga meddelanden och fel i programmet loggas till System.out. Men tanken är att i framtiden kunna ändra hur loggningen sker (t.ex. logga till olika typer av filer).

Ditt uppdrag i deluppgift C är: Chatmeddelanden och fel (exceptions) loggas till System.out tillsammans med datum och tid. Tänk på att göra en lösning så det är enkelt att i framtiden välja hur loggning sker. Till din kodlösning skall även en .pdf lämnas in som besvarar samt motiverar:

- Hur din lösning gör det enkelt att i framtiden ändra loggningsmetod.
- Vilket eller vilka designmönster använde du?
- Vilka andra lösningar och designmönster hade du kunnat använda och varför valde du inte dessa?

Förberedelser

- Föreläsningar och laborationer på sal
- Kursbok

Inlämning

Du laddar upp en .zip innehållandes:

1. Koden för en fungerande lösning av applikationen med efterfrågad funktionalitet.
2. En eller flera .pdf ’er med efterfrågat innehåll.

Betyg

Inlämningen kan ge något av betygen komplettera (U)*, godkänd (G) eller väl godkänd (VG).

För att ha chans på betyget G skall: deluppgift A och B vara lösta.

För att ha chans på betyget VG skall: deluppgift A, B och C vara lösta.

**Om din inlämning inte lever upp till kraven för G har du möjlighet att komplettera uppgiften för att nå godkänd.*
Om din inlämning ej lever upp till kraven för väl godkänd har du regel ej möjlighet att komplettera upp till väl godkänd. Men ifall din inlämning är väldigt nära VG men slarv eller enklare fel förekommer kan du ges möjlighet att komplettera till väl godkänd.

Betygskriterier

När vi betygsätter inlämningen tittar vi på om:

- Ingen funktionalitet är bruten av omdesignen.
- Omdesignen har gjort koden enklare att underhålla och bygga vidare på.
- Motiveringen till lösningen är rimlig.
- Klasser, metoder, variabler och package har tydliga och informativa namn som följer Javas namnkonvention.
- Indentering och övrig kodstruktur är tydlig och konsevent.
- Klasserna är indelade i lämpliga paket.
- Klasser och metoder har tydliga ansvar.

För VG tittas även på:

- Alternativa lösningar presenteras och hur dessa relateras till den valda lösningen.
- Det är enkelt att lägga till eller ändra funktionalitet.
- Hög kvalitet på lösningar för deluppgift A, B och C.

UML nuvarande design