# Uppgift 2:

# Network Programming

# ”Ordning och reda”



## Syfte

Att ta den nya kunskapen om nätverk och få den användbar samt rensa upp föregående uppgift

## Uppgift

Uppgiften består av att **modifiera**[projektstrukturen ska bibehållas!] ert chattprogram till att stödja X klienter, detta genom att skicka serializerade meddelanden skapade av klasser

1. Lägg in stöd för multipla klienter så vi kan börja chatta!
   1. Jag ska kunna välja namn på klienterna
   2. Jag ska få veta om en annan klient har ”joinat” chatten: ”Gustav joined” på både servern och andra klienter
   3. Jag ska veta om klienten har gått ur chatten: ”Gustav left”, detta händer om jag stänger klienten på både servern och andra klienter
   4. Alla meddelanden som skickas ska även skrivas ut i server consolen: ”Gustav wrote: X” på både servern och andra klienter
   5. Som i uppgift1: Jag ska också kunna välja vilken server jag joinar (vilket IP jag ska ansluta till)
2. Lägg in en basklass med namn CNetMessage
   1. Den ska inkludera SerializeHelper.h
   2. Det ska finnas två #defines
      1. SERIALIZE(aStream, aType)
      2. DESERIALIZE(aStream, aType)
   3. De ska kalla på de publika funktionerna i helpern



* 1. Det ska finnas fyra funktioner:
     1. PackMessage 🡪Kör DoSerialize och kolla om jag skickar under 512 bytes
     2. UnPackMessage 🡪skapar en serializer med vår buffer och kör DoDeSerialize på den
     3. DoSerialize 🡪 Kallar vårt macro SERIALIZE med alla variabler vi vill skicka
     4. DoDeSerialize 🡪 Kallar vårt macro DESERIALIZE med alla variabler vi har tagit emot
  2. Denna klass ska serializa fyra variabler:
     1. Char myID 🡪 id på klassen (som vi måste sätta på alla klasser som överlagrar denna)
     2. Uint myTimeStamp 🡪Tiden då vi skapade meddelandet
     3. Ushort mySenderID 🡪 den som skickar
     4. Ushort myTargetID 🡪 den som ska ta emnot

1. Lägg in en klass med namn CNetMessageChatMessage som ärver av CNetMessage
   1. Lägg in en variabel: std::string myMessage
   2. Överlagra DoSerialize och DoDeserialize och gör nödvändig funktionalitet i dom
   3. Glöm inte att kalla på basklassens funktioner först
2. Lägg till en klass CNetMessageManager
   1. Den har en funktion
   2. CreateMessage<> som skapar upp en netmessage klass. (templatiserad funktion)
   3. Den skapar meddelandet och sätter timestamp, id, sender och en default target(servern)
3. Era program ska fortfarande detektera om någon inte svarar: Se Ping i förra labben
   1. Både från servern och klienten
   2. Disconnectas efter 10 sekunder Server🡨🡪Client

Inlämning

* Inlämningen ska ske med körbara Exe filer samt source
* Vill ni göra er egen serializer, gör då det ☺

Jag vill göra mer

* Skapa flera olika meddelanden: KickBan som kastar ut en spelare
* Skapa olika roller, administratör, och vanlig användare
* Testa att bjuda in kamrater till chatten!
* Skriv ut IP nummret när servern startar ”Server started with IP: xxx.xxx.xxx.xxx”

Extra

* Vår serializer klass kan ni läsa mer om här:
  + http://cpplove.blogspot.se/2013/05/my-take-on-c-serialization-part-i.html
* Bra resurs: http://gafferongames.com/networking-for-game-programmers/