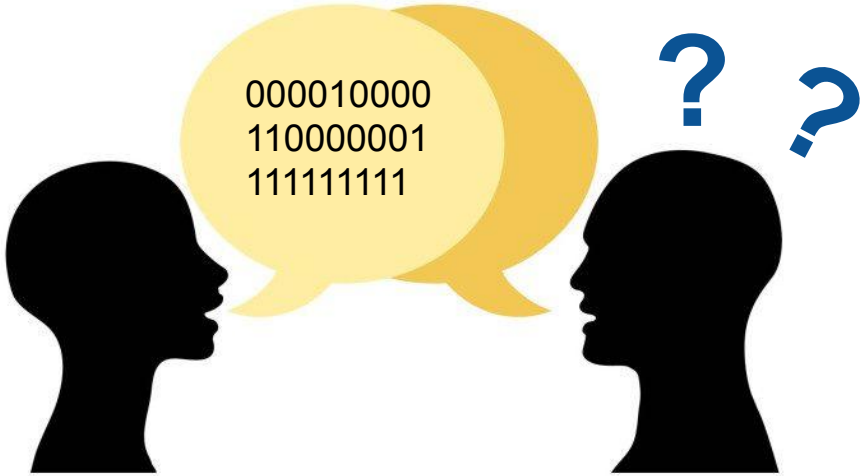
Software Distribuït

SESSIÓ 2: Pràctica 1 - Protocol

Que és un protocol?

Seguit de normes escrites que regeixen la comunicació entre dispositius



En el nostre cas ens permetrà que client i servidor puguin “entendre’s” per tal de jugar a “Ship, Captain and Crew”.

Idees bàsiques

- El protocol ha de ser conegut pels diferents interlocutors i l'han de seguir estrictament
- Les característiques i el contingut de les trama han d'estar ben definides:
  - Longitud de dades a llegir (tot i que no sempre cal llegir tota la trama sencera)
  - Tipus de dades a llegir
  - Significat dels paràmetres
- A l'hora de definir els datagrames sempre triar la opció més senzilla (més fàcil d'entendre, més fàcil d'implementar).
  - Tots els missatges segueixin una mateixa estructura però amb diferenciadors clars
  - Tenir que els diferents tipus de trames siguin fàcilment distingibles

Exemples

Trames GPS (tot i que no és ben bé un protocol client/servidor):

$GPGGA,123519,4807.038,N,01131.000,E,1,08,0.9,545.4,M,46.9,M,,*47
$GPGLL,4916.45,N,12311.12,W,225444,A,*1D

Altres exemples de protocols de correu que trobareu a la pàgina web de l'assignatura:

MAIL FROM:<Smith@Alpha.ARPA>
RCPT TO:<Brown@Beta.ARPA>

Com definir un protocol?

- Identificar les accions que duen a terme el client i servidor
- Definir una trama “inconfusible” per cada tipus d’acció (també els paràmetres associats i el seu tipus)
- Definir una seqüència o seqüències vàlides (tenir en compte els moments en que s’espera una resposta)
- Definir un datagrama d’error (tenint en compte els errors més comuns)

Se us dóna el protocol fet i el podeu consultar en el següent link:
https://github.com/UB-GEI-SD/UB-GEI-SD.github.io/blob/master/Protocol.md

Avaluació

Avaluació de les pràctiques :

- 10% Javadoc i Diagrames
- 20% Autoavaluació
- 10% Junit
- 60% Codi (Sense bugs i amb els modes de client i servidor que es demanen)

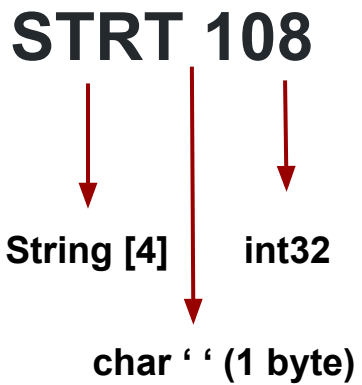
Es recomana llegir atentament l'enunciat de pràctiques per tenir clars tots els requisits del projecte:

http://ub-gei-sd.github.io/Practica1/

Pràctiques en els ordinadors de la UB

- Per la sessió de test de la Pràctica 1, la vostra pràctica haurà de funcionar en els PCs de l'aula.
- Per executar el codi de la Pràctica 0:
  - A través de IntelliJ IDE.
  - Indicar que ho faci com a projecte de MAVEN, si ho demana
  - Especificar la JDK : java-8-openjdk-amd64
  - Executar les comandes MAVEN en la consola del propi IntelliJ.
- L'estructura del fitxer MAVEN del projecte és vàlida fins a JDK 8.

Format dels datagrames del Protocol

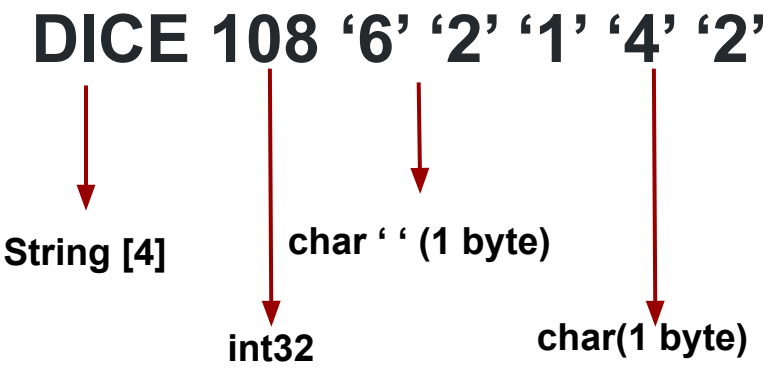


STRT<SP><ID>

<SP> ::= ‘ ‘ Space ascii character

<ID> ::= Over network integer positive number

Format dels datagrames del Protocol



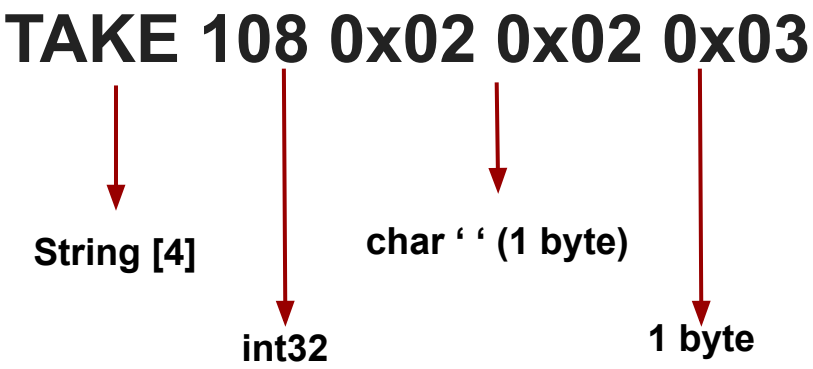
DICE<SP><ID>5(<SP><VALUE>) ; Five times <SP><VALUE>

<VALUE> :: = ‘6’,‘5’,‘4’,‘3’,‘2’,‘1’ in ascii characters

<SP> ::= ‘ ‘ Space ascii character

<ID> ::= Over network integer positive number

Format dels datagrames del Protocol



TAKE<SP><ID><SP><LEN>*5(<SP><POS>) ;

<LEN> :: = byte value from 0x00 to 0x05

<POS> :: = byte value from 0x01 to 0x05

Trames d’ERROR

El protocol especifica el format de les trames d’error però no especifica quan s’utitlitzaran ni com.

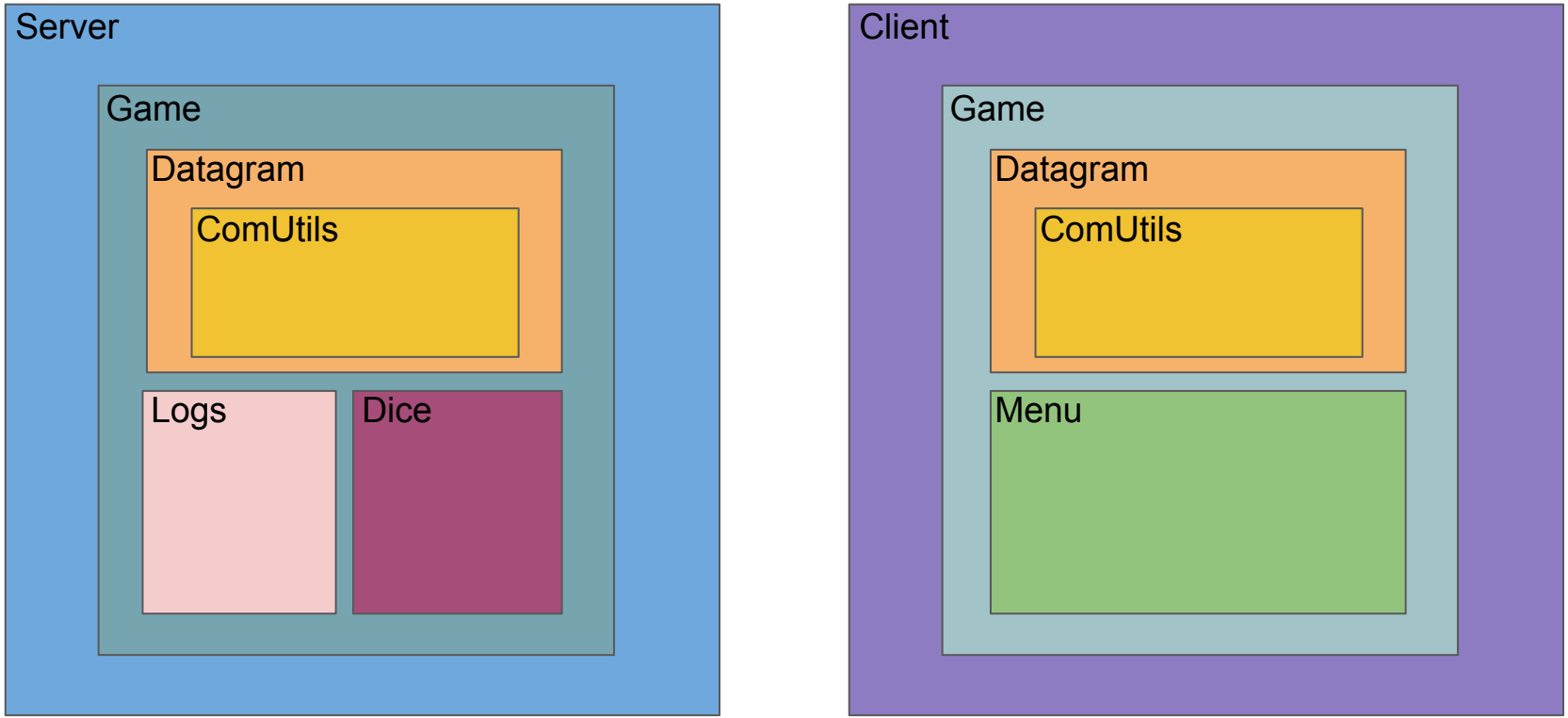
ERRO<SP><Message>

<Message> ::= 2<DIGIT>*99<VCHAR> ;

Especificar amb la resta de la classe si s’utilitzarà algun missatge o codi específic en les següents situacions:

- Datagrames mal formats
- Datagrames rebuts en un moment no esperat
- Datagrames incomplerts
- Datagrames amb accions incorrectes
- ...

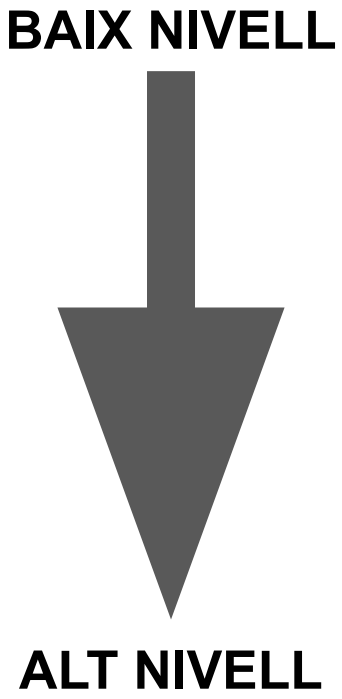
Estructura de Classes



* Estructura básica proposada

Estructura de Classes

- Ampliar funcionalitats ComUtils ⇔ TESTS
  - Lectura/Escriptura de dades
  - Conversió de dades
  - ...
- Crear classe a nivell de datagrames ⇔ TESTS
  - Lectura/Escriptura de datagrames
  - Control d'errors de les trames
  - ...
- Classe/classes auxiliars ⇔ TESTS
  - Control dels daus, tirades i puntuacions
  - Logs
  - Menús i missatges per pantalla
  - ...

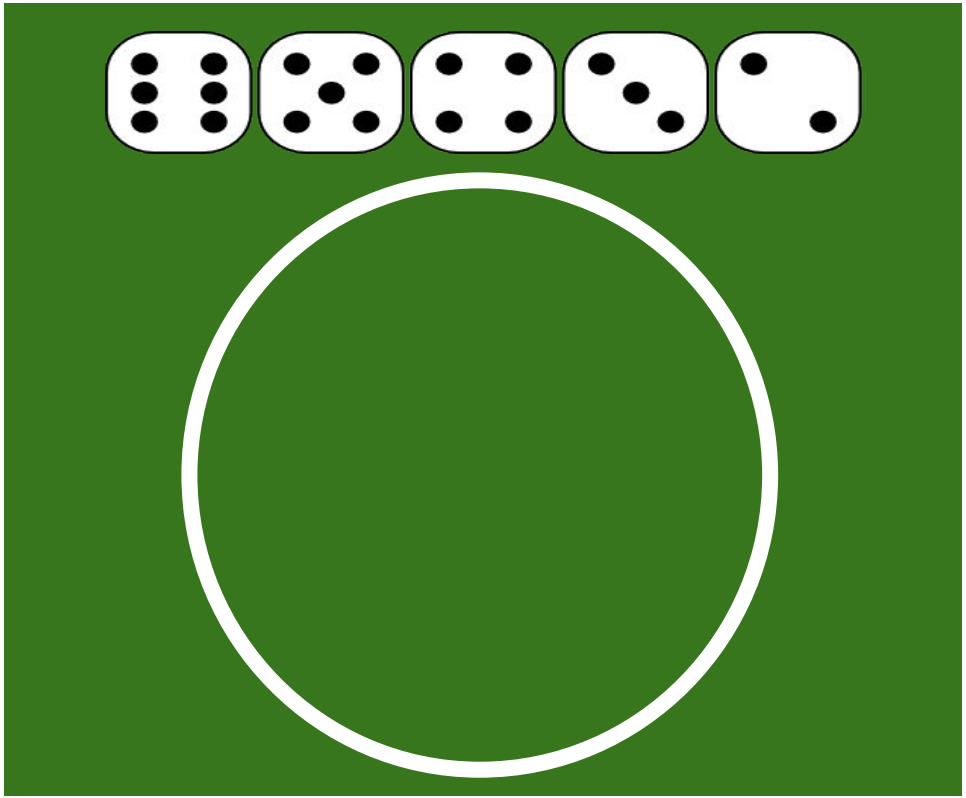


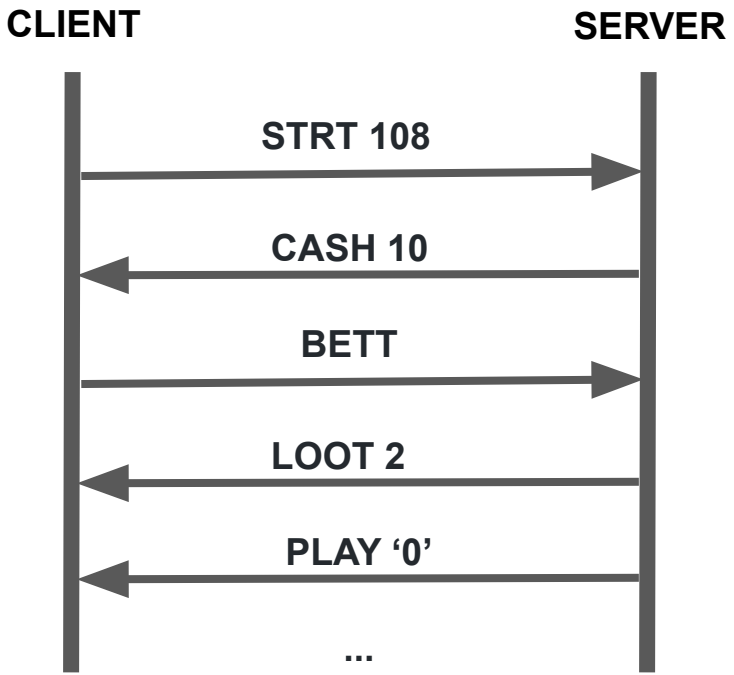
Estructura de Classes

En properes classes parlarem sobre el disseny de client i servidor:

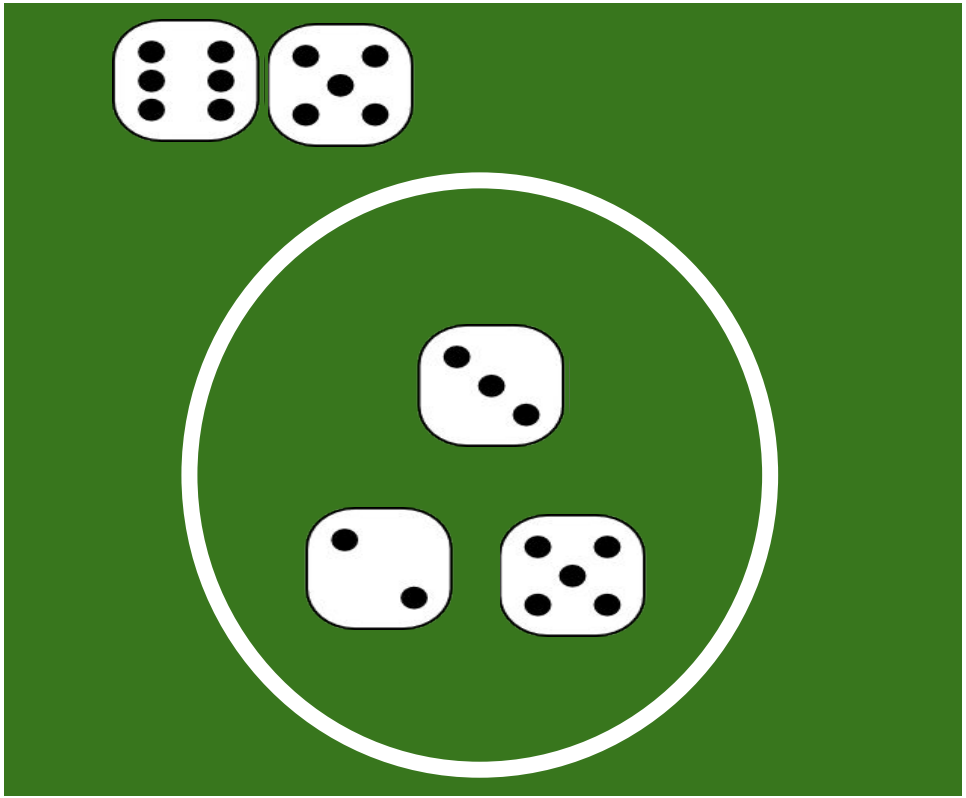
- Classe amb la lògica del Joc ⇔ TESTS
  - Màquina d'estats
  - Control d'errors
- Classes principals de Client i Servidor
  - Creació de sockets
  - Threads

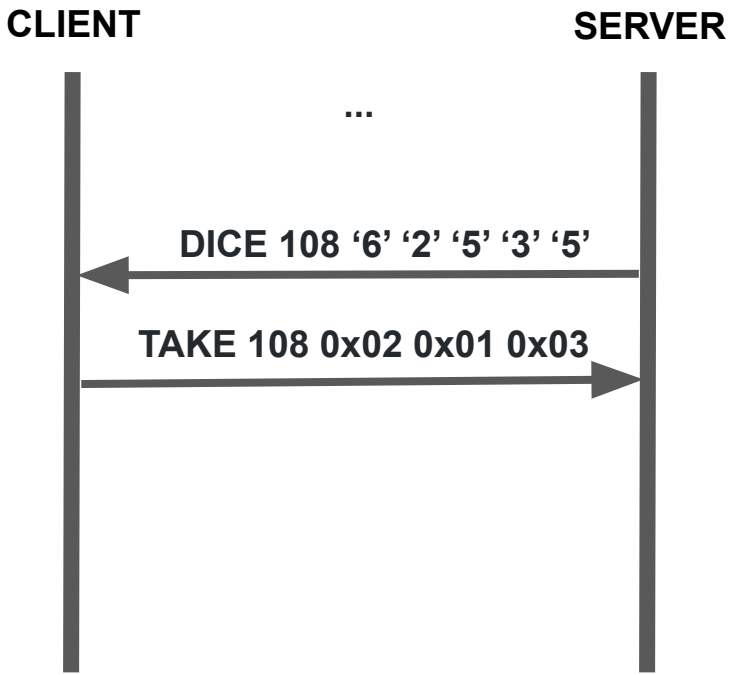
Protocol “Ship, Captain and Crew”



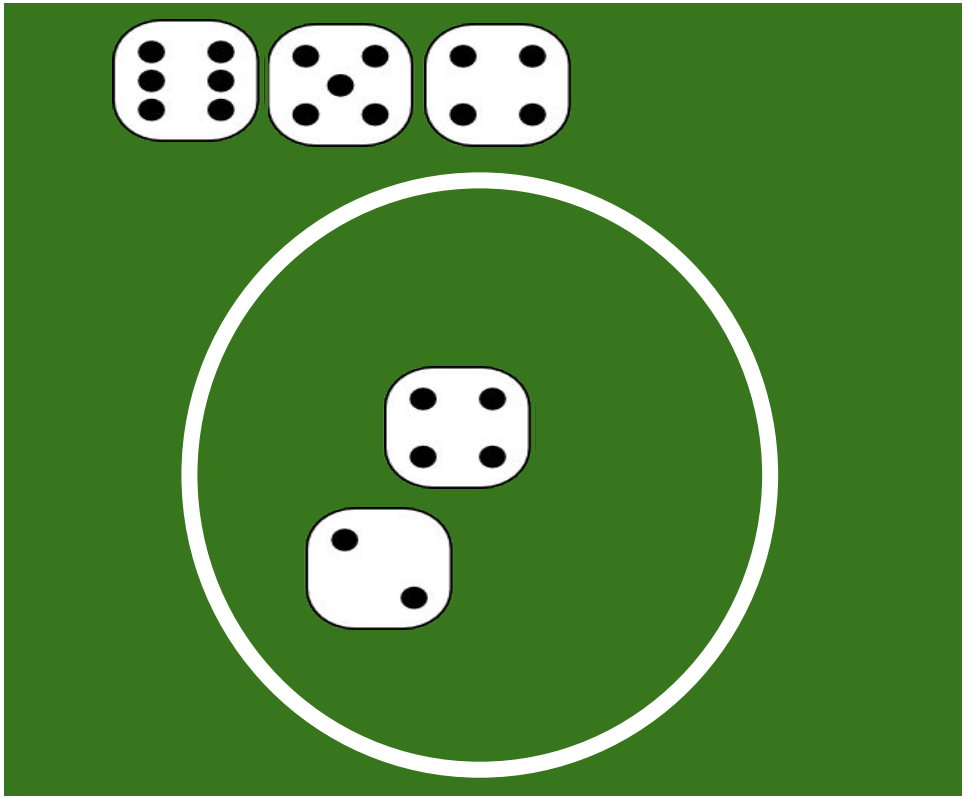


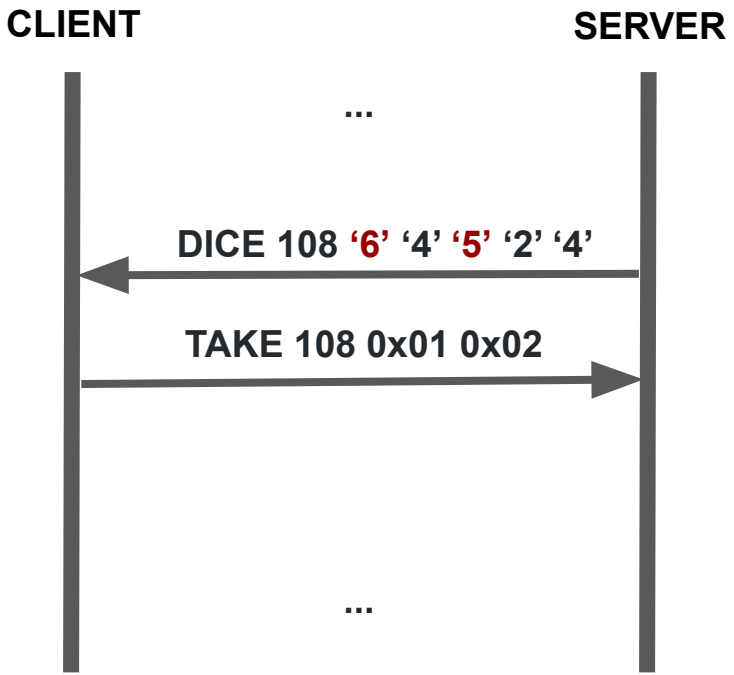
Protocol “Ship, Captain and Crew”



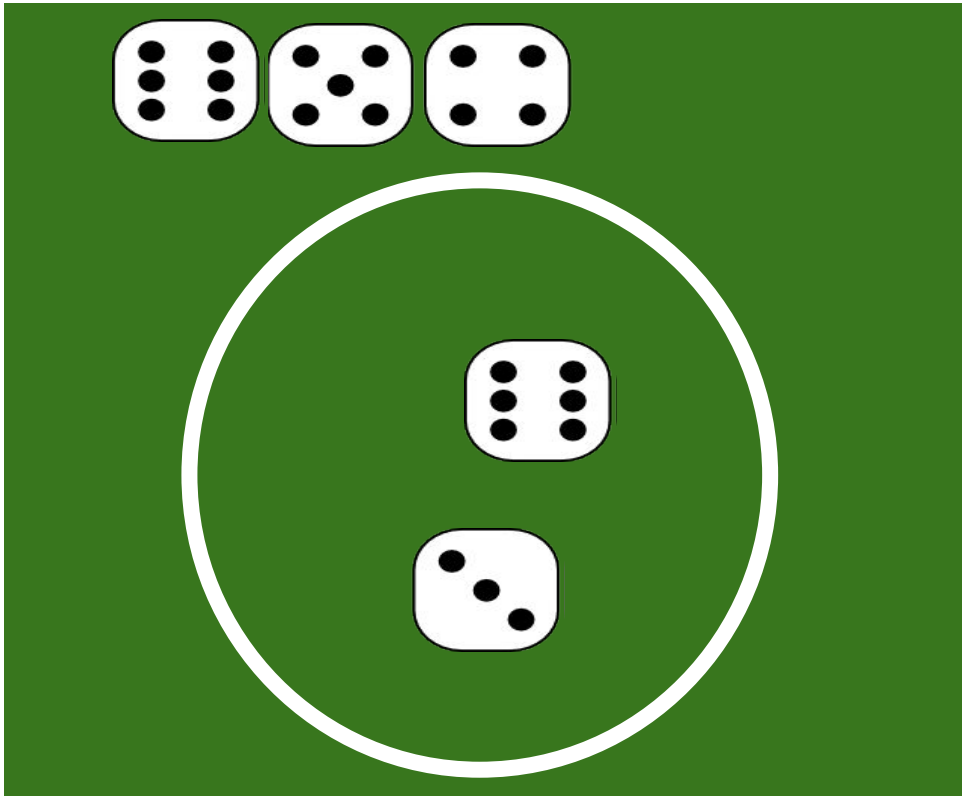


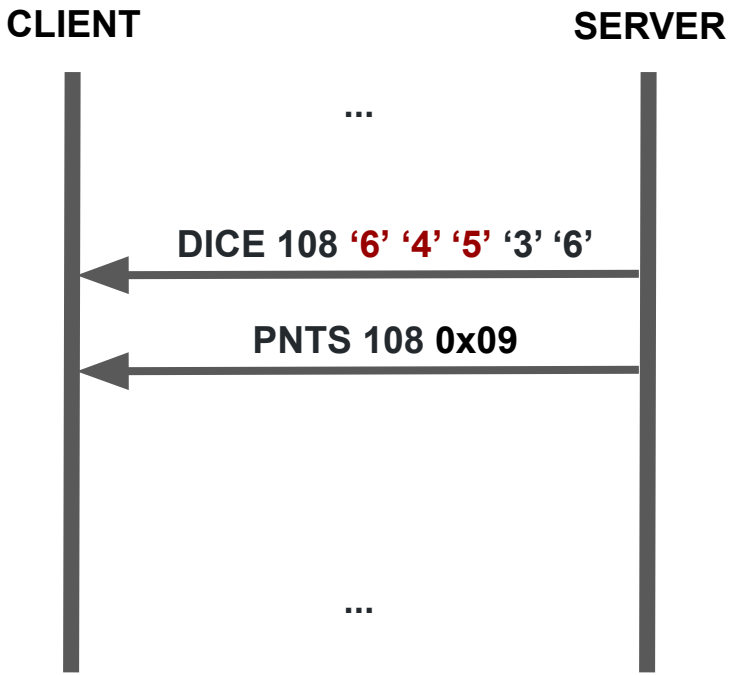
Protocol “Ship, Captain and Crew”



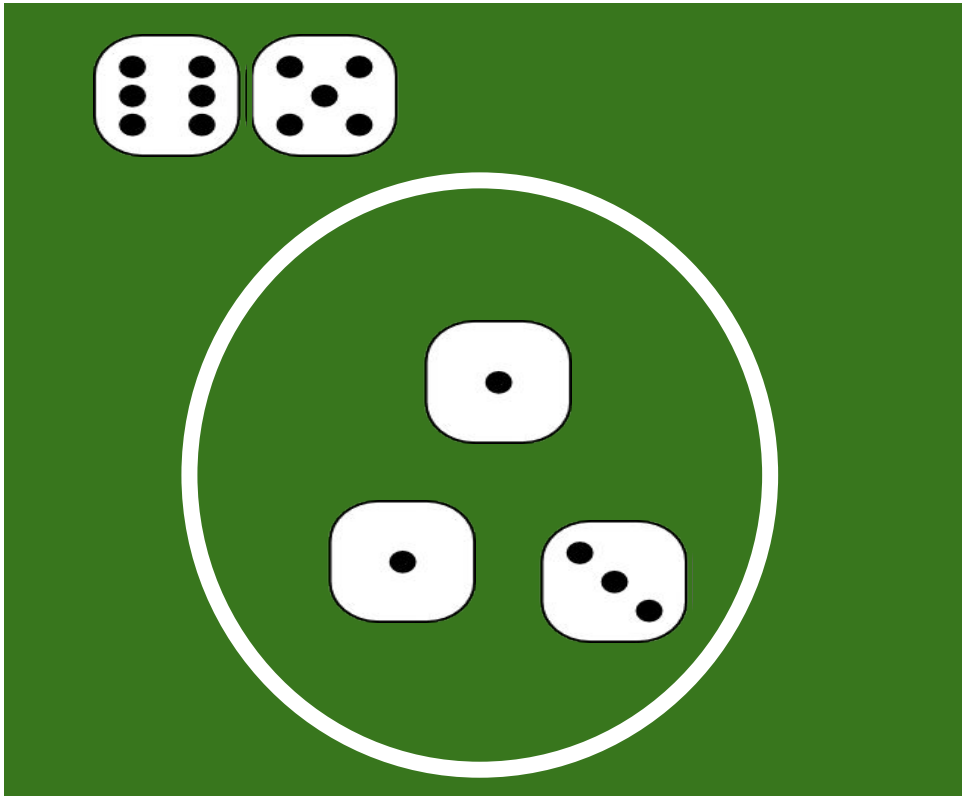


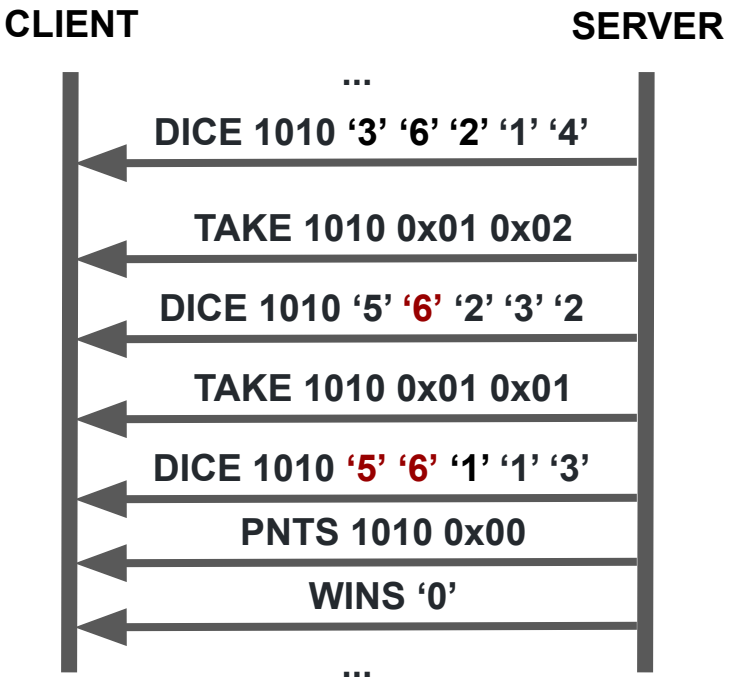
Protocol “Ship, Captain and Crew”



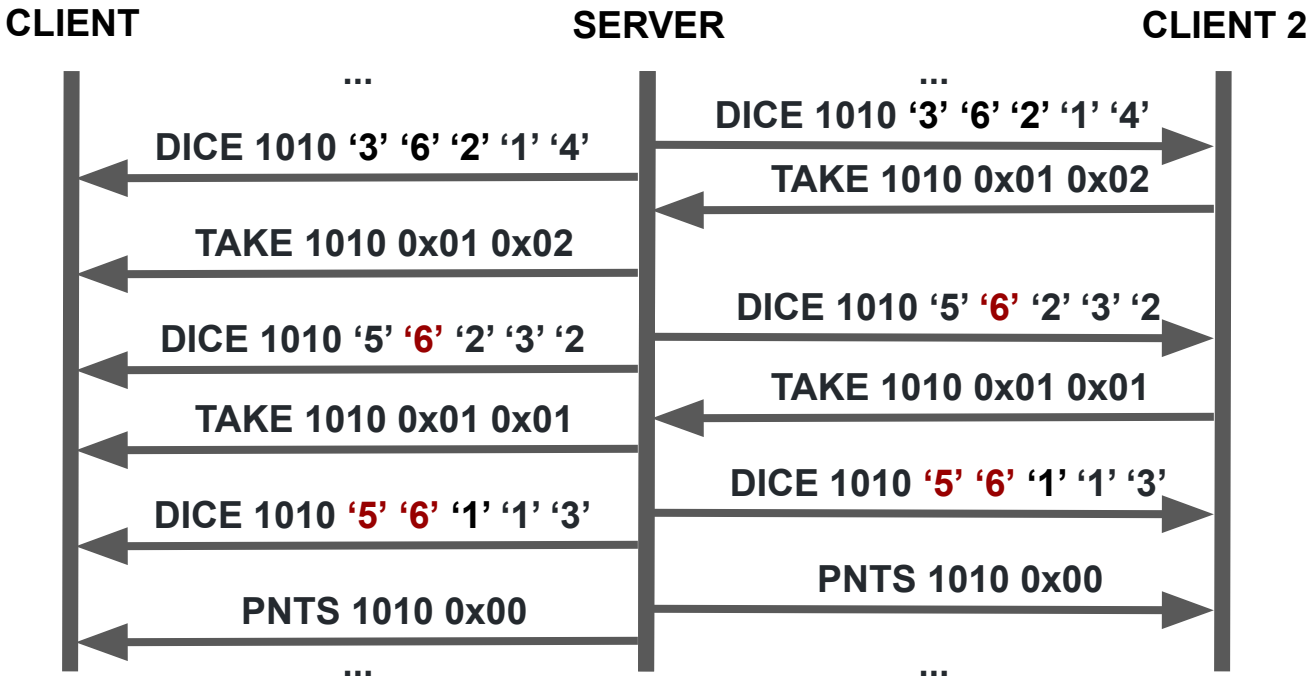


Pràctica 1: Ship, Captain and Crew





Variant del Protocol



* Exemple només d’una part de la partida en mode 2 jugadors

Per a la pròxima setmana ...

- Llegir detingudament el protocol i detectar possibles dubtes, errors o ambigüitats.
- Ampliar les funcions de ComUtils i crear noves classes per poder implementar el protocol (tenir en compte l'herència i encapsulament de les funcions de nivell a bits a nivell de trames).
    - Podeu executar el codi emulant els sockets sobre fitxers.
- Realitzar els tests corresponents (conjuntament amb les funcions).
- Fer un esquema de les trames que fan avançar en el joc (diagrama de bloc, diagrama d'estats).