

A dark blue vertical bar runs down the left side of the page. A blue arrow points to the right from this bar, containing the date.

09/06/2019

# Rapport projet SH13

Several thin, curved lines in dark blue and light grey originate from the bottom left and sweep upwards and to the right.

Mathieu Esteban et Stéphane Padrao  
POLYTECH SORBONNE

## Table des matières

I) Présentation du projet .....	2
A) Présentation des règles.....	2
B) Présentation de la partie client/serveur .....	2
II) Les protocoles d'échanges entre le serveur et le client.....	3
A) Echanges Client vers Serveur .....	3
B) Echanges Serveur vers Client .....	6
III) Le programme SH13.c et Server.c .....	7
A) Le programme server.c .....	7
B) Le programme SH13.c.....	9

## I) Présentation du projet

### A) Présentation des règles

Le jeu SH13 que nous avons à développer en partie, est un jeu se composant de 13 cartes. Ces cartes représentent des personnages, qui ont comme caractéristique des symboles. Chaque combinaison de 2 ou 3 symboles, caractérise donc un personnage en particulier. Chaque symbole est présent dans un nombre limité répartis entre chaque joueur, allant de 3 à 5.

Le jeu se compose de 4 joueurs qui reçoivent chacun 3 cartes, la 13<sup>ème</sup> carte représentant la carte coupable. Ainsi, chaque joueur va pouvoir au début du jeu identifier les symboles qui composent leur carte afin de pouvoir identifier ensuite si d'autres joueurs possèdent leur symbole ou non et ainsi trouver la carte coupable par déduction.

A chaque tour un joueur va pouvoir effectuer une action :

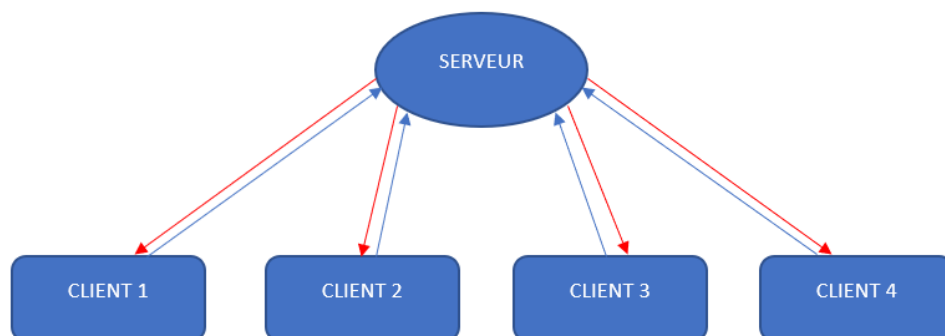
- Désigner une carte coupable, dans la liste des 12 cartes suspectes. Si celui-ci trouve le coupable, il gagne alors la partie. Sinon, il est éliminé et ne peut plus jouer.
- Faire une demande à tous les joueurs si ceux-ci possèdent un symbole en particulière sans en connaître le nombre précis.
- Demander à un joueur choisi le nombre qu'il a d'un symbole en particulier au total.

### B) Présentation de la partie client/serveur

Le jeu est décomposé en 2 programmes .c :

- Un programme « server.c » qui gère la partie server du jeu.
- Un autre programme « sh13.c » qui gère la partie client du jeu.

Ainsi, au niveau réseau le programme se décompose comme ci-dessous :



Lorsque le serveur est lancé, une demande de connexion est envoyée du client au serveur :

— : une demande de connexion est envoyée du client au serveur avec les informations suivantes :  
 @serveur n°portduserveur @client n°portduclient nomduclient

— : la connexion est acceptée par le serveur qui informe le client de ID et le diffuse la liste des ID des joueurs à tous les clients.

## II) Les protocoles d'échanges entre le serveur et le client

### A) Echanges Client vers Serveur

Le client communique avec le serveur. Cependant, comme on joue à 4 joueurs, il y a 4 clients qui communiquent avec le serveur. On utilise donc un mutex afin de bloquer l'accès à l'information avant que celle-ci ne soit traitée. Nous utilisons également un thread pour la réception du message, et celui-ci est indépendant du code principal. Cela se fait en partie à l'aide des fonctions `fn_serveur_tcp()` ou `pthread_create()`. Cependant, pour envoyer un message au serveur, le client va utiliser la fonction `sendMessageToServer()`. Cette fonction prend en paramètres d'entrées un pointeur sur chaîne de caractères qui représente l'adresse IP, le numéro de port et un pointeur sur chaîne de caractères qui correspond au message à envoyer. Une variable synchro permet enfin de savoir quand un message reçu peut être traité.

Le nombre de joueurs étant de 4, il y a donc 4 clients différents communiquant avec le serveur. Nous utilisons donc un « mutex » qui permet de bloquer l'accès à l'information avant que celle-ci ne soit traitée. Nous utilisons également un thread indépendant du code principal, servant à la réception du message. Cela est réalisé à l'aide des fonctions « `fn_serveur_tcp ()` » et « `pthread_create ()` ». La fonction « `sendMessageToServer ()` » sera utilisée pour envoyer un message du client au serveur. Ainsi, cette fonction prendra en paramètres d'entrées :

- Un pointeur sur chaîne de caractère qui représente l'adresse IP
- Le numéro de port
- Un pointeur sur chaîne de caractère correspondant au message à envoyer

```
// RAJOUTER DU CODE ICI
sendMessageToServer(gServerIpAddress, gServerPort, sendBuffer);
connectEnabled=0;
```

*Appel de la fonction « `sendMessageToServer ()` »*

En outre, une variable « synchro » permet de savoir lorsqu'un message reçu peut être traité. Ci-dessous, les différents états que peut prendre la variable « synchro » dans le programme :

```
if (synchro==1) //si synchro égal 1 je regarde le type de buffer reçu
{
    printf("consomme |%s|\n",gbuffer);
    switch (
        Test si variable synchro est à 1

        break;
    }
    synchro=0; //j'ai traité le message réseau, donc je refais passer synchro à 0
}
```

*Fin de traitement « synchro » remis à 0*

L'application côté client contient une partie graphique qui a été réalisé à l'aide de la bibliothèque SDL. Cette bibliothèque nous permet avec différentes fonctions de gérer l'affichage graphique (images, boutons...) en affichant, créant ou supprimant des objets graphiques. Les clics de la souris seront également gérés par cette bibliothèque pour la sélection des boutons du jeu.

Ainsi, le bouton où il y a marqué « Connect » appelle la fonction « sendMessageToServer () » lorsqu'on appuie sur celui-ci. Cela envoie l'IP, le port et le nom du client. Le même principe est utilisé pour le bouton « Go » ou lorsqu'on clique sur les cases ou les noms des personnages.

Le client va ainsi envoyer 4 types de messages qui vont commencer chacun par des lettres différentes :

ENTETE	MESSAGE	TYPE DE MESSAGE
<b>C</b>	<i>IPClient, PORTClient, NAMEClient</i>	<u>Demande de connexion</u> IPClient = Adresse IP du Client PORTClient= Port du Client NAMEClient = Nom du joueur
<b>G</b>	<i>ID, PersoAccuse</i>	<u>Envoi d'une accusation</u> ID = ID du joueur PersoAccuse = Personnage Accusé
<b>O</b>	<i>ID, Objet</i>	<u>Envoi d'une demande d'une colonne Objet</u> ID = ID du joueur Objet = Objet sélectionné
<b>S</b>	<i>ID, JoueurSel, Objet</i>	<u>Envoi d'une demande de l'objet d'un joueur</u> ID = ID du joueur JoueurSel=ID du joueur sélectionné Objet = Objet sélectionné

De façon plus précise les messages envoyés par le client permettent de :

- C : le client va envoyer ses informations de connexions (adresse IP, numéro de port, nom)
- G : le client a trouvé la carte coupable non distribuée, il envoie alors son ID et la carte qu'il soupçonne :

```

    case 'G':
// RAJOUTER DU CODE ICI
sscanf(buffer,"%c %d %d", &com, &JoueurQuest, &AccuseJoueur);
printf("COM=%c JoueurQuest=%d AccuseJoueur=%d\n",com, JoueurQuest, AccuseJoueur);
if(AccuseJoueur == deck[12])
{
    tcpClients[JoueurQuest].gagne = 1;
    printf("Félicitation, %s a gagné !\n\n", tcpClients[JoueurQuest].name);
    sprintf(reply,"W %s", tcpClients[JoueurQuest].name);
    broadcastMessage(reply);
    close(newsockfd);
    close(sockfd);
    return 0;
}
else
{
    tcpClients[JoueurQuest].perdu = 1;
    templose = JoueurQuest;
    printf("Mince alors ! %s a perdu !\n\n", tcpClients[JoueurQuest].name);
    sprintf(reply,"P %s", tcpClients[JoueurQuest].name);
    broadcastMessage(reply);

    if((tcpClients[0].perdu == 1)&&(tcpClients[1].perdu == 1)&&(tcpClients[2].perdu == 1)&&(tcpClients[3].perdu == 1))//
    {
        sprintf(reply,"F");
        broadcastMessage(reply);
        close(newsockfd);
        close(sockfd);
        return 0;
    }
}
}

```

- O : le client demande à l'ensemble des joueurs, ceux qui possèdent un objet précis. Il envoie alors son ID ainsi que la ligne correspondant à l'objet. On parcourt ensuite le tableau « tableCartes » pour chaque joueur, en omettant la personne qui fait la demande. Si la personne détient au moins un exemplaire de cet objet, alors on attribue la valeur 100 à cette case. Au de marquer la valeur 100 sur une case, une étoile sera affichée :

```

    case 'O':
    // RAJOUTER DU CODE ICI

    sscanf(buffer,"%c %d %d", &com, &JoueurQuest, &ObjDemande);
    printf("COM=%c JoueurQuest=%d ObjDemande=%d\n",com, JoueurQuest, ObjDemande);
    templose = JoueurQuest;
    for(y = 0; y < 4; y++)
    {
        for(cp = 0; cp < 4; cp++)
        {
            if(cp == JoueurQuest)
            {
            }
            else
            {
                if(tableCartes[cp][ObjDemande] == 0)
                {
                    sprintf(reply,"V %d %d %d ", cp, ObjDemande, tableCartes[cp][ObjDemande]);
                    sendMessageToClient(tcpClients[y].ipAddress, tcpClients[y].port, reply);
                }
                else
                {
                    sprintf(reply,"V %d %d 100 ", cp, ObjDemande);
                    sendMessageToClient(tcpClients[y].ipAddress, tcpClients[y].port, reply);
                }
            }
        }
    }
}

```

- S : le client demande à un joueur précis s'il possède ou non un objet et combien il possède d'exemplaire de cet objet, il envoie alors son ID, le joueur à qui il demande et l'objet en question :

```

case 'S':
// RAJOUTER DU CODE ICI

sscanf(buffer,"%c %d %d %d", &com, &JoueurQuest, &joueurQuestionne, &ObjDemande);
printf("COM=%c JoueurQuest=%d joueurQuestionne = %d ObjDemande=%d\n",com, JoueurQuest, joueurQuestionne, ObjDemande);
templose = JoueurQuest;
for(y = 0; y < 4; y++)
{
    if(joueurQuestionne == JoueurQuest)
    {
    }
    else
    {
        sprintf(reply,"V %d %d %d ", joueurQuestionne, ObjDemande, tableCartes[joueurQuestionne][ObjDemande]);
        sendMessageToClient(tcpClients[y].ipAddress, tcpClients[y].port, reply);
    }
}

sprintf(reply,"V %d %d %d ", JoueurQuest, ObjDemande, tableCartes[JoueurQuest][ObjDemande]);
sendMessageToClient(tcpClients[JoueurQuest].ipAddress, tcpClients[JoueurQuest].port, reply);

break;

```

## B) Echanges Serveur vers Client

Ensuite le client doit pouvoir lire ce que le serveur lui envoie ou répond. Dans les trames reçues par le client un message avec une lettre en entête va préciser l'action que le programme devra exécuter. Ainsi, lorsque cet entête est reçu, le programme va par l'intermédiaire d'un « switch case » sélectionner l'action associée à celle-ci :

ENTETE	MESSAGE	TYPE DE MESSAGE
<b>I</b>	<i>ID</i>	Le joueur reçoit son ID
<b>L</b>	<i>Name1, Name2, Name3, Name4</i>	Envoie de l'ensemble des noms des joueurs en Broadcast
<b>D</b>	<i>C1, C2, C3</i>	Distribution des 3 cartes pour chacun des joueurs
<b>M</b>	<i>JoueurCourant</i>	Les joueurs reçoivent l'ID du joueur courant en Broadcast
<b>V</b>	<i>Ligne, Colonne</i>	Le joueur reçoit une valeur de tableCarte
<b>P</b>	<i>JoueurLoose</i>	Le joueur a perdu
<b>W</b>	<i>JoueurWin</i>	Le joueur a gagné
<b>F</b>		L'ensemble des joueurs ont perdu

De façon plus précise les messages reçus par le client permettent de :

- **I** : l'ID est reçu par le client au début de la partie.
- **L** : chaque joueur reçoit la liste de l'ensemble des joueurs connectés au serveur.
- **D** : chaque client reçoit ses 3 cartes.
- **M** : chacun des clients va recevoir le numéro du joueur courant. Ainsi, cela permettra l'affichage du bouton « GO ». Dans le cas où le client est le joueur courant, le bouton « GO » s'affiche. Dans le cas contraire celui-ci est masqué.
- **V** : chaque joueur va recevoir la valeur d'une case du tableau, dans lesquelles sont affichées le nombre de symbole que possèdent chaque joueur. Ainsi, après que tous les clients se soient connectés au serveur, chaque joueur va recevoir un certain nombre des 8 symboles en fonction des cartes qu'il recevra, ce qui complètera sa ligne du nombre de symboles. En outre, une condition if avec un compteur s'incrémentant permet de ne pas remplir les 8 cases qui ont déjà été rempli pour chaque joueur lorsqu'un joueur reçoit une réponse à une de ses questions.
- **P** : le client va recevoir le nom d'un perdant. Cela aura pour effet de mettre hors-jeu le client qui a perdu, mais l'application ne se fermera pas. Le perdant reste connecté car son client doit toujours pouvoir répondre aux questions des autres clients.
- **W** : chaque client va recevoir le nom du vainqueur. Ce nom sera affiché sur leur console, la variable « quit » sera ensuite mis à 1 afin de fermer l'application de chaque client.
- **F** : dans ce cas la partie se termine car l'ensemble des joueurs ont perdu. On passe la variable « quit » à 1 afin de sortir de la boucle « while () » et fermer l'application de chaque client.

### III) Le programme SH13.c et Server.c

#### A) Le programme server.c

Notre programme « server.c » est composé de plusieurs fonctions permettant d'initialiser le jeu. Ainsi ces différentes fonctions sont :

- printDeck() qui permet d'afficher sur la console le deck entier.
- melangerDeck() afin de mélanger les cartes aléatoirement. On doit inclure pour cela la bibliothèque <time.h> et on ajoute la ligne « srand(time(NULL)) ; ».
- createTable() va permettre d'attribuer 3 cartes à chaque joueurs.

Le serveur communique avec les clients/joueurs avec le protocole TCP. Il utilise par ailleurs plusieurs fonctions. printClients() va afficher l'adresse IP, le port et le nom de chaque joueur dans la console du serveur. findClientByName() reçoit en paramètre d'entrée un pointeur sur chaîne de caractères qui représente un nom. Le serveur va alors chercher le client qui correspond à ce nom. sendMessageToClient() va permettre d'envoyer un message à un client précis.

Le serveur va communiquer avec les 4 clients avec le protocole TCP. Plusieurs fonctions vont être utilisées par « server.c ». Ces fonctions sont :

- « printclient() » permettant d'afficher l'adresse IP, le port et le nom de chaque joueur dans la console du serveur :

```
0: localhost 32001 math
1: localhost 32002 jerem
2: localhost 32003 tintin
3: localhost 32004 thomas
```

- « findClientByName() » permettant de trouver un client par son nom :

```
int findClientByName(char *name)
{
    int i;

    for (i=0;i<nbClients;i++)
        if (strcmp(tcpClients[i].name,name)==0)
            return i;
    return -1;
}
```

Pour communiquer, le serveur a deux moyens :

- il peut envoyer un message à un client en particulier. Pour réaliser cela, nous utilisons la fonction « sendMessageToClient() ».
- il peut envoyer un message à l'ensemble des clients à l'aide de la fonction « Broadcastmessage() ».



Afin de gérer les connexions entre le serveur et le client, nous utilisons des fonctions du protocole TCP :

- la fonction « socket() » permettant de définir le mode TCP et connect.
- la fonction « bind() » permettant de spécifier le mode TCP ou UDP.
- la fonction « accept() » qui accepte la connexion.
- les fonctions « read() » et « write() » qui lisent ou écrivent dans un socket.

L'exécution du serveur se fait à l'aide de la commande suivante :

```
root@kali:~/Bureau/SH13Mathieu/SH13_11_06_19# ./server 32000
```

Commande : `$. ./server <Numéro de port Server>`

<Numéro de port Server> sera le port sur lequel les clients se connecteront au serveur.

En se lançant le serveur utiliser les fonctions vu précédemment pour initialiser les cartes :

```
root@kali:~/Bureau/SH13Mathieu/SH13_11_06_19# ./server 32000
0 Sebastian Moran
1 irene Adler
2 inspector Lestrade
3 inspector Gregson
4 inspector Baynes
5 inspector Bradstreet
6 inspector Hopkins
7 Sherlock Holmes
8 John Watson
9 Mycroft Holmes
10 Mrs. Hudson
11 Mary Morstan
12 James Moriarty
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
8 John Watson
0 Sebastian Moran
1 irene Adler
10 Mrs. Hudson
12 James Moriarty
6 inspector Hopkins
2 inspector Lestrade
5 inspector Bradstreet
7 Sherlock Holmes
3 inspector Gregson
11 Mary Morstan
4 inspector Baynes
9 Mycroft Holmes
01 01 02 00 00 01 01 02
02 01 00 01 00 01 01 01
01 01 02 02 01 00 01 00
00 01 01 02 02 01 00 00
```

Le serveur attend ensuite la connexion d'un client, qu'il détectera lors de l'envoi d'un message avec une entête « C » correspondant à une demande de connexion. Dès lors qu'un client se connecte, son ID est cherché, puis le client envoie un message personnel avec une entête « I », afin de lui communiquer son ID. Un message avec une entête « L » est également envoyé à tous les clients afin de montrer qui est actuellement connecté.

Puis, lorsque les 4 joueurs se sont connectés, le serveur envoie deux messages personnels pour chaque client :

- un message avec comme entête « D » pour lui distribuer ses cartes.
- un second message avec comme entête « V » pour lui donner la ligne qui correspond dans « tableCartes ».

Enfin, le serveur envoie un message en « Broadcast » à tous les clients avec comme entête « M » afin que chaque client connaisse le joueur courant. Ainsi, chaque joueur pourra jouer chacun son tour.

## B) Le programme SH13.c

**L'exécution du programme client se fait à l'aide de la commande suivante :**

```
root@kali:~/Bureau/SH13Mathieu/SH13_11_06_19# ./sh13 localhost 32000 localhost 32004 thomas
```

Commande : `$ ./sh13 <IP server> <Port server> <IP Client> <Port Client> <Nom du joueur>`

<IP server> = correspond à l'adresse IP du serveur, ici nous sommes en local d'où « localhost »

<Port server> = correspond au numéro de port déclaré au lancement de l'application serveur

<IP Client> = correspond à l'adresse IP du client, ici nous sommes en local d'où « localhost »

<Port Client> = correspond au numéro de port du Client

<Nom du joueur> = correspond au nom du joueur choisi

Exemple de lancement de 4 clients :

- `$ ./sh13 localhost 32000 localhost 32001 thomas`
- `$ ./sh13 localhost 32000 localhost 32002 mathieu`
- `$ ./sh13 localhost 32000 localhost 32003 stephane`
- `$ ./sh13 localhost 32000 localhost 32004 nathan`

### L'IHM avant connexion :

Ensuite lorsque la commande est lancée une IHM se lance, cette IHM à son lancement contient par défaut :

- Le nom du joueur courant dans le titre de la fenêtre, ex : « **Player Name : tintin** »
- Le bouton « **Connect** » qui permet au client de se connecter au serveur
- Un tableau contenant en ligne du haut les objets et en colonne à gauche l'ensemble des joueurs dont lui. Ce tableau permet d'avancer dans l'enquête de la carte manquante.
- Enfin, une liste des noms des cartes potentiellement coupables est affichée dans un tableau en dessous. On peut cocher les suspects à écarter avec une croix rouge.

Player Name : tintin

CONNECT








5	5	5	5	4	3	3	3


	Sebastian Moran	
	Irene Adler	
	Inspector Lestrade	
	Inspector Gregson	
	Inspector Baynes	
	Inspector Bradstreet	
	Inspector Hopkins	
	Sherlock Holmes	
	John Watson	
	Mycroft Holmes	
	Mrs. Hudson	
	Mary Morstan	
	James Moriarty	














### L'IHM après connexion :

Une fois que le premier joueur est connecté, il recevra le nom des joueurs qui se connectent après lui au fur et à mesure :

Player Name : tintin


	 5	 5	 5	 5	 4	 3	 3	 3
tintin								
jerem								
-								
-								

	Sebastian Moran	
	Irene Adler	
	Inspector Lestrade	
	Inspector Gregson	
	Inspector Baynes	
	Inspector Bradstreet	
	Inspector Hopkins	
	Sherlock Holmes	
	John Watson	
	Mycroft Holmes	
	Mrs. Hudson	
	Mary Morstan	
	James Moriarty	

### L'IHM en jeu :

Une fois que les 4 joueurs sont connectés, les cartes sont distribuées entre eux de manière aléatoire. Ces cartes sont affichées à droite et le client peut également observer sa ligne où son prénom est marqué, remplis de chiffres représentant le nombre de symboles qu'il possède :



Player Name : tintin

tintin	1	1	2	0	0	1
jerem						
thomas						
math						

Character List:

- Sebastian Moran
- Irene Adler
- Inspector Lestrade
- Inspector Gregson
- Inspector Baynes
- Inspector Bradstreet
- Inspector Hopkins
- Sherlock Holmes
- John Watson
- Mycroft Holmes
- Mrs. Hudson
- Mary Morstan
- James Moriarty

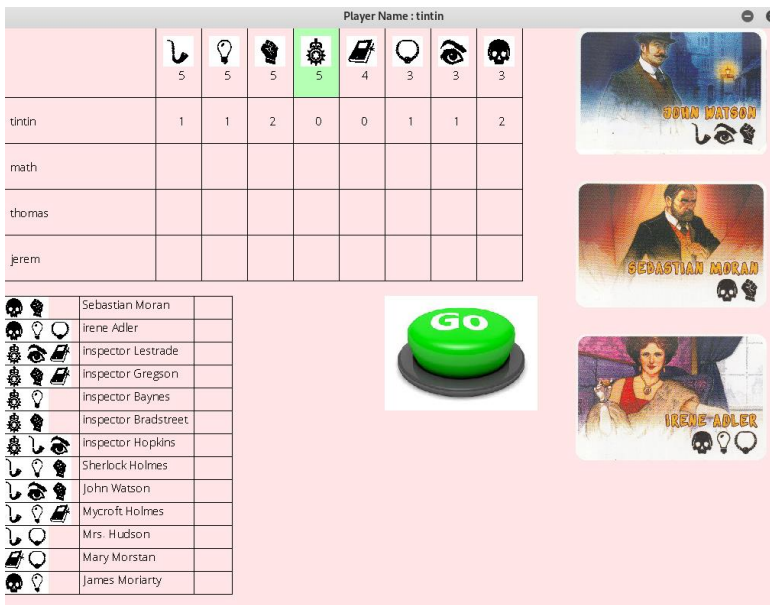
GO button

Action possible pour le joueur qui peut jouer :

#### 1) Qui à ce type d'item ?

- Le joueur peut sélectionner l'objet d'une des colonnes qui passe alors en vert
- Le joueur click sur GO
- Le serveur émet un broadcast de la réponse à tous les autres joueurs

Exemple : Tintin demande qui a au moins un item couronne ?



Player Name : tintin

tintin	1	1	2	0	0	1
math						
thomas						
jerem						

Character List:

- Sebastian Moran
- Irene Adler
- Inspector Lestrade
- Inspector Gregson
- Inspector Baynes
- Inspector Bradstreet
- Inspector Hopkins
- Sherlock Holmes
- John Watson
- Mycroft Holmes
- Mrs. Hudson
- Mary Morstan
- James Moriarty











GO button

Résultat de l'appui sur le bouton GO :













Sur l'IHM de Fred la colonne est complète avec tintin sa colonne est complète et on observe un astérisque dans la case des autres joueurs. Ce qui signifie que math, thomas et jerem possède au moins une fois l'item du collier.

A savoir que l'IHM des autres joueurs ne voient pas si tintin possède au moins un item collier ou non.

IHM de jerem après le tour de tintin :

Player Name : jerem									
									
tintin	5	5	5	5	4	3	3	3	
math				*					
thomas				*					
jerem	0	1	1	2	2	1	0	0	









	Sebastian Moran	
	Irene Adler	
	Inspector Lestrade	
	Inspector Gregson	
	Inspector Baynes	
	Inspector Bradstreet	
	Inspector Hopkins	
	Sherlock Holmes	
	John Watson	
	Mycroft Holmes	
	Mrs. Hudson	
	Mary Morstan	
	James Moriarty	

## 2) Combien as-tu de ce type d'item ?


























- Le joueur sélectionne l'item dans une colonne et celle-ci passe alors en vert
- Le joueur sélectionne également le nom du joueur et celui-ci passe en rouge
- Le joueur clique sur le bouton GO
- Un message de type Broadcast est envoyé à tous les clients

Exemple : Math demande combien d'item poing possède Tintin ?

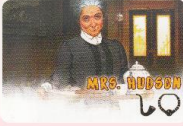
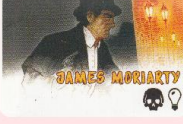
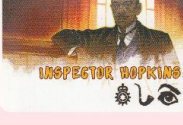
Player Name : math

	 5	 5	 5	 5	 4	 3	 3	 3
tintin			2					
math	2	1	0	1	0	1	1	1
thomas				*				
jerem				*				

 	Sebastian Moran	
 	Irene Adler	
 	Inspector Lestrade	
 	Inspector Gregson	
 	Inspector Baynes	
 	Inspector Bradstreet	
 	Inspector Hopkins	
 	Sherlock Holmes	
 	John Watson	
 	Mycroft Holmes	
 	Mrs. Hudson	
 	Mary Morstan	
 	James Moriarty	

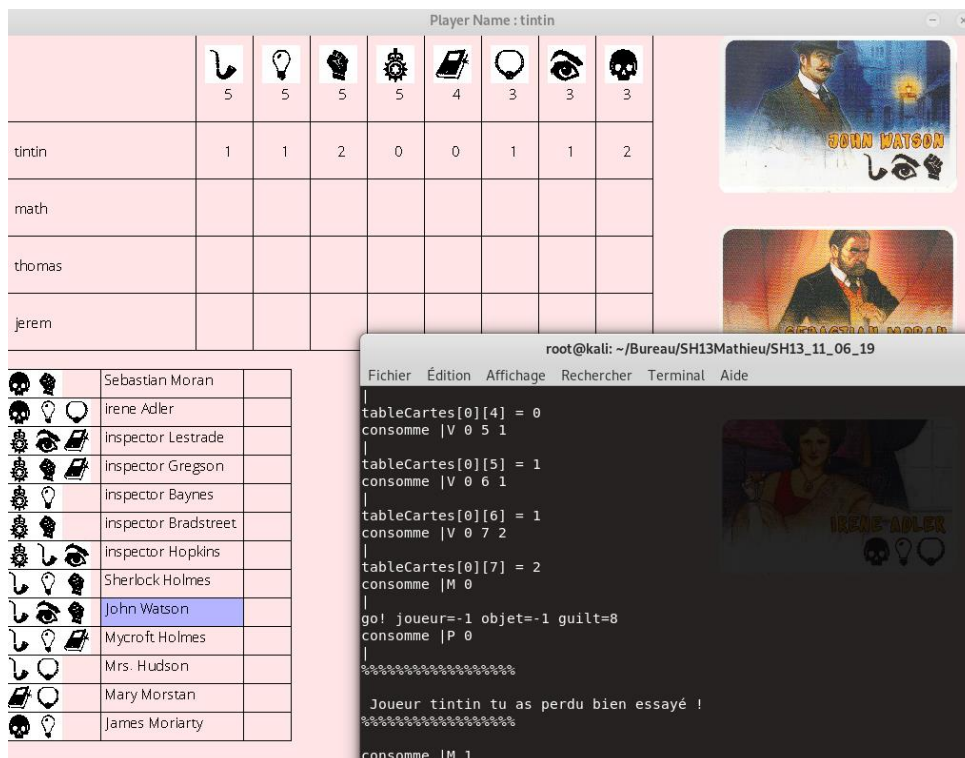
Tintin envoie alors comme réponse 2. Seul Math verra sur son écran le nombre d'item poing que possède Tintin.

### 3) Désignation de la carte coupable (manquante)








Le but est de faire une accusation pour gagner la partie.

- Le joueur doit sélectionner un nom de suspect, dès lors celui-ci passe en bleu
- Le joueur click sur le bouton GO
- Le serveur émet un broadcast de la réponse à tous les autres joueurs














Exemple mauvais coupable : Tintin désigne « John Watson » :



Player Name : tintin

							
tintin	1	1	2	0	0	1	1
math							
thomas							
jerem							

	Sebastian Moran	
	Irene Adler	
	Inspector Lestrade	
	Inspector Gregson	
	Inspector Baynes	
	Inspector Bradstreet	
	Inspector Hopkins	
	Sherlock Holmes	
	<b>John Watson</b>	
	Mycroft Holmes	
	Mrs. Hudson	
	Mary Morstan	
	James Moriarty	

```

root@kali: ~/Bureau/SH13Mathieu/SH13_11_06_19
Fichier Édition Affichage Rechercher Terminal Aide
|
tableCartes[0][4] = 0
consomme |V 0 5 1
|
tableCartes[0][5] = 1
consomme |V 0 6 1
|
tableCartes[0][6] = 1
consomme |V 0 7 2
|
tableCartes[0][7] = 2
consomme |M 0
|
go! joueur=-1 objet=-1 guilt=8
consomme |P 0
|
=====
Joueur tintin tu as perdu bien essayé !
=====
consomme |M 1
  
```

Ce n'était malheureusement pas le coupable. Tintin aura toujours son client ouvert pour envoyer des informations aux autres joueur mais ne pourra plus jouer.



[illegible]

Page 16 sur 16