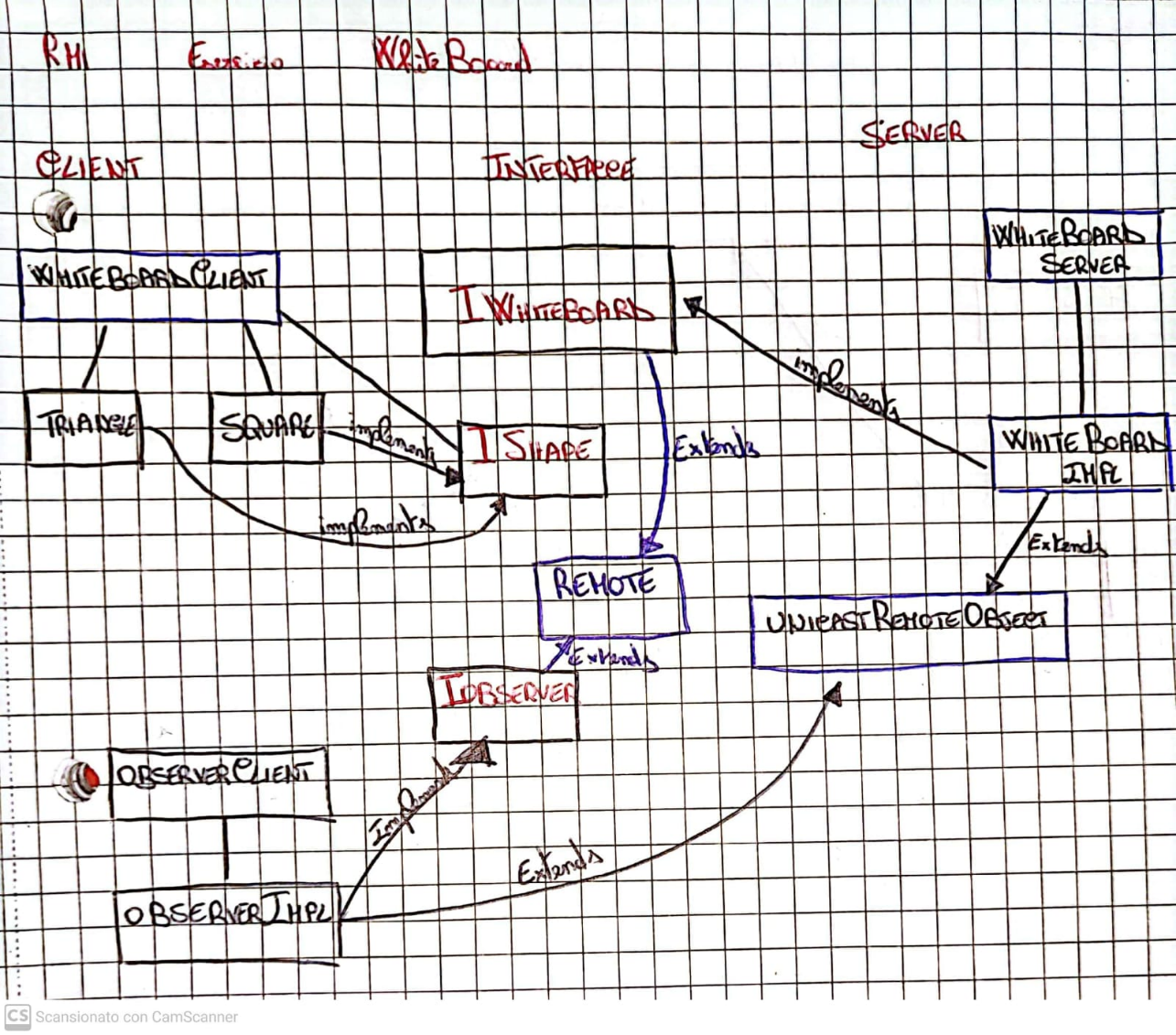
**Class Diagram**



Consideriamo una **whiteBoard**, su cui un **Client** (WhiteboardClient) può inviare (**draw**) delle **forme** (Shape).  
Dato che stiamo inviando come **parametro** un **oggetto** tramite una **chiamata** sfruttando RMI, **shape** deve estendere la classe **Serializable**.

Per il resto, abbiamo degli **observer** che si attaccano al **subject** (Whiteboard) e vengono notificati ogni volta che il **client** aggiunge una forma alla whiteboard.  
Quest’ultima è un oggetto i cui metodi saranno invocabili da remoto, dunque stiamo parlando di rendere **remoto** il metodo **addShape**().  
La **definizione** del servizio remoto è stata implementata per **ereditarietà**.

Ho perso ore per capire che i metodi invocabili da remoto devono **throware** l’eccezione **RemoteException** ovviamente.

**IWhiteboard**

E’ l’interfaccia remota del **Server**, invocabile dal **Client**.  
Dunque ci saranno i metodi **addShape**(IShape s) e **getShapes**(), e infine ritroviamo pure **observerAttach**(Observer obs) fondamentale per il concetto di callback.

E’ un’interfaccia che **estende** l’interfaccia **Remote**, dunque questi metodi possono presentare l’eccezione **RemoteException**.

Conseguentemente ritroviamo:

**WhiteBoardImpl**  
Ci permette di definire i servizi remoti per **ereditarietà**.  
Troviamo **2 liste**, una per gli **observer** e uno per il **contenutoBoard**.   
Consideriamo che il costruttore deve invocare anche **super**() ovviamente, per invocare il costruttore di **UnicastRemoteObject** (throwando RemoteException).

Troviamo i **metodi** per gli **observer**, dunque **observerAttach**(Observer obs) e il **metodo privato** **notifyAllObserver** che richiamerà tramite il meccanismo delle callback il metodo **notifyObserver**.

Infine ritroviamo il **metodo addShape**(IShape s) invocabile lato client.  
Dato che inviamo come **parametro** un oggetto, questo deve extendere l’interfaccia **Serializable**.

**Server**  
Semplicemente la **registrazione** del **servizio** nel **registro RMI**, in modo che invochi un processo in ascolto di richieste da parte del client.

**IShape**

Dato che deve essere passato come parametro di un’invocazione **RMI** deve estendere la classe **Serializable**.  
Sarà implementato dalle classi **Square** e **Triangle**.

**IObserver**

Offre il metodo di **callback** notifyObserver, dunque deve estendere la classe **Remote** e ovviamente throware la RemoteException.

**ObserverImpl**

L’implementazione di questa interfaccia è fornita da **ObserverImpl**, che estende UnicastRemoteObject e implementa l’interfaccia **IObserver** in modo da poter definire per **ereditarietà** un servizio remoto.

Ha come dato privato una **remoteWhiteboard**, che sarebbe il riferimento all’oggetto **whiteboard** presente sul server. Questo viene costruito mediante un parametro passato nel costruttore, consultando il **registro RMI**, e viene utilizzatto per vedere quali forme sono all’interno della lavagna sfruttando il metodo **getShapes**.

Questo è ovviamente un caso particolare, non è detto che ogni volta nel costruttore dell’observer devo passargli il riferimento all’oggetto remoto (subject).

**ObserverClient**

Non andrà a registrare il **servizio remoto** offerto dagli observer, dato che questi sono utilizzati per le callback.  
Dunque **consulta RMI registry** per ricavare il **riferimento** all’oggetto remoto **whiteboard**, servizio offerto lato server, lo **passa** nel costruttore dell’observer e poi **effettua** l’attach.

**CLIENT**

Ricava il **riferimento** al servizio remoto offerto dal server, dunque ottiene il **riferimento** alla **whiteboard** e invia delle **forme** sfruttando il servizio remoto **addShape**(s).