Setup

1. Creare una GameView serializable
2. Creare il metodo ***setChangedAndNotifyObservers(Event)*** in Observable, che accorpa setChanged e notifyObservers

> Gli Observer si basano su un concetto semplice: quando chiamo addObserver, viene effettuato un link tra Observer e Observable grazie a una lambda. Da li' in poi, quando l'Observer viene notificato, viene chiamata la funzione all'interno della lambda. Le enum servono per distinguere la notify in piu' varianti.

Da model a view

1. Rendere Board, Shelf e Game Observable, con un enum di eventi (Game.Event) come contrassegno
   - UPDATED_BOARD
   - UPDATED_SHELF
   - UPDATED_CURRENT_PLAYER
   - UPDATED_CHAT
2. Aggiungere ***setChangedAndNotifyObservers(Game.Event)*** dopo ogni metodo che va a modificare uno di quegli attributi, dando come argomento Game.Enum
3. Aggiungere un metodo di register in ServerImpl, che con addObserver effettua un link tra ServerImpl (Observer) e la funzione che va chiamata quando l'Observer legge qualcosa: ***client.update(...)***
4. Nel costruttore di ClientImpl, chiamare la funzione register del server, cosi' da effettuare il link alla creazione della ClientImpl

> In pratica, quando modifico un attributo, e quindi chiamo ***setChangedAndNotifyObservers(Event)***, l'Observer si attiva e a sua volta chiama ***client.update(...)***

5. Creare un metodo ***client.update(...)*** che genera una GameView e la passa alla TextualUI chiamando ***textualui.update(...)***
6. Creare un metodo ***textualui.update(...)*** che fa uno switch case sulla enum e stampa quello che e' cambiato

Da view a model

1. Rendere TextualUI observable, con un enum di eventi (TextualUI.Event) come contrassegno
    - TILE_SELECTION
    - COLUMN_SELECTION
    - TILE_INSERTION
2. Creare i metodi che prendono gli input dall'utente e aggiungere ***setChangedAndNotifyObservers(TextualUI.Event)*** dopo ogni metodo che identifica uno di quegli eventi, dando come argomento TextualUI.Enum
3. Effettuare un link tra TUI e ClientImpl aggiungendo addObserver nel costruttore di clientImpl, avente all'interno della lambda server.update(this, arg)

> Percio', quando l'observer rileva un cambiamento, chiama ***server.update(...)*** che a sua volta chiama ***controller.update()***

4. Aggiungere un metodo in Engine chiamato update che fa uno switch case sulla enum chiama le funzioni corrispondenti del controller