PREPARAZIONE PARTITA

1) Primo giocatore avvia la partita -> sceglie numero giocatori (numPlayer), viene istanziato il MatchState, inserisce nickname, viene inserito player in array

2) Attesa prox giocatore

3) Secondo giocatore entra in partita e inserisce nickname, sceglie due caselle non occupate (necessaria verifica + istanziare due builder + metti isOccupied true) e viene inserito nell'array -> addPlayer

4) Attesa prox giocatore

5) Terzo giocatore (uguale)

6) Viene assegnata una carta ad ogni giocatore PlayerModel (con assignCard pesca casualmente in deck -> da modificare se numPlayer=3 con removeCard3Players) e viene istanziata la corrispondente GodCard da getGod. La carta pescata si elimina dal deck con RemoveCard

INIZIO PARTITA

1) nextPlayer fa partire primo giocatore, lo chiama e si attiva la sua carta

2) TURNO*

3) Quando ha finito il turno (mossa+costruzione -> hasMoved = hasBuild = true) chiama endTurn

4) endTurn chiama nextPlayer e inizializza hasMoved = hasBuild = false

5) nextPlayer cambia il currentPlayer e fa partire il suo turno con beginTurn()

CONDIZIONI FINE PARTITA

Ad ogni turno devo controllare che:

• Un giocatore vince (hasWon -> se true chiama endMatch)

• Se l'array di giocatori rimane con un elemento

PossibleMoves: devo controllare il notMoveUp (se true per colpa di Atena non posso salire di livello)

TURNO*

1) Controllo se ho perso con hasLost (se per entrambi i builder sono disabled OR PossibleMoves=0)

1-1) se ho perso chiamo cleanAndRemovePlayer, mi elimino dall'array di giocatori e chiamo nextPlayer

1-2) se un solo builder è disabled, obbligo il giocatore a muovere l'altro

1-3) altrimenti faccio scegliere al giocatore quale vuole muovere

MOVE(BuilderModel, CellModel) : posso muovere se hasMoved==false

2) Il giocatore sceglie la casella in cui spostarsi (CellModel)

3) Verifico che sia in possibleMoves del builder

4) Cambio cella al builder

5) Metto hasMoved true

6) chiamo hasWon per verificare se ho vinto

BUILD(BuilderModel, CellModel) : posso costruire solo se hasMoved==true

7) Il giocatore sceglie la casella in cui costruire

8) Verifico che sia in possibleBuilds del builder e che il pezzo di costruzione che ci serve sia disponile nell'array remainingPieces

9) Cambio il level della cella (addLevel in CellModel) -> se non posso costruire HO PERSO (mi cancello dall'array dei giocatori)

10) Passo a checkPieces l'intero del livello della costruzione che hai fatto, così lui lo toglie dall'array di pezzi disponibili "remainingPieces"

11) hasBuild true

12) controllo se sono bloccato -> se true, disabilito builder

APOLLO

1) modifico hasLost (se PossibleMoves == PossibleSwitch == 0)

2) "

3) Verifico che sia in possibleMoves o in PossibleSwitch del builder

4) " + se PossibleSwitch cambio casella al builder dell'altro giocatore

5) "

ARTEMIDE

1) Aggiungo due attributi: la cella da cui parto (startingCell) e un boolean secondMove

2) Faccio la move -> hasMoved true

3) Può fare un'altra move SSE hasMoved==true && secondMove==false && verifico che la cella in cui si vuole muovere sia in possibleMoves e diversa da startingCell -> secondMove=true

4) Costruisce -> hasBuild = true

ATENA

1) Inizializzo notMoveup=false

2) MOVE

3) Controllo se è salita di livello : se si, notMoveUp=true

4) BUILD

ATLAS

1) Modifico la BUILD

DEMETRA

1) Aggiungo due attributi: la cella su cui costruisce la prima volta (startingCell) e un boolean secondBuild

2) Logica uguale ad Artemide

EFESTO

1) Aggiungo due attributi: la cella su cui costruisce la prima volta (startingCell) e un bollean secondBuild

2) Logica ugaule ad Artemide (devo costruire per forza sulla startingCell)

MINOTAURO

1) Modifico hasLost (se PossibleMoves == PossibleSwitch == 0 + controllo sulla casella successiva di PossibleSwitch in quella direzione è libera e senza cupola)

2) Controllo se la casella scelta è in PossibleMoves o PossibleSwitch (con casella successiva libera): in quest'ultimo caso cambio la cella del builder dell'avversario

3) IDEA: decidere se aggiungere un metodo PossibleSwitchMinotauro in BuilderModel o fare il controllo su PossibleSwitch direttamente nell'hasLost della carta

PAN

1) Modifico hasWon (aggiungo il caso se scende di due o più livelli)

PROMETEO

1) Controllo che notMoveUp==true oppure PossibleLevelUp==0 : in questo caso si può attivare il potere

2) Aggiungo attributo firstBuild=false

3) Prima volta che costruisci controllare che hasMoved==firstBuild==false

4) Metto firstBuild=true

5) MOVE

6) BUILD