

<<Interface>>
CellObserver

- PLAYER: Player - counter: int + canMoveUp: boolean + canWinPerimetric: boolean + Turn(p: Player) + start(): void

+ evolveTurn (w: Worker) : void + move (w: Worker) : void + build (w: Worker) : void + win (w: Worker) : boolean + lose(w:Worker) : boolean

Hephaestus Artemis Demeter Minotaur Prometheus Charon Apollo Atlas Hestia + evolveTurn(Worker w) : void + move(Worker w) : void + limitWin(Worker w) : void moveSwap(Worker opposite) : void + move(Worker w) : void - move(Worker w) : void + move(Worker w) : void + move(Worker w) : void verifyRestriction() : void + move(Worker w) : void + move(Worker w) : void + build(Worker w) : void + move(Worker w) : void + move(Worker w) : void + build(Worker w) : void + build(Worker w) : void + move(Worker w) : void + move(Worker w) : void + move(Worker w) : void + build(Worker w) : void + build(Worker w) : void + move(Worker w) : void + move(Worker w) : void + build(Worker w) : void + build(Worker w) : void + win(Worker w) : boolean + build(Worker w) : void + build(Worker w) : void + build(Worker w) : void + win(Worker w) : boolean + win(Worker w) : boolean + build(Worker w) : void + build(Worker w) : void + build(Worker w) : void + win(Worker w) : boolean + win(Worker w) : boolean + build(Worker w) : void + win(Worker w) : boolean + win(Worker w) : boolean + win(Worker w) : boolean + secondMove(Worker w) : void + win(Worker w) : boolean qui fare override di move qui fare override di win qui override build qui override move

ho messo il metodo "applyRestriction()": fa sì che venga settato a 1 un attributo in worker che non è il worker del player possessore della carta Athena, attributo chiamato ad esempio "hasMoved"



metterei un attributo booleano 0/1 su ogni worker per vedere se la sua vittoria è limitata o meno