Notes to the Model UML diagram

Team #6: Aspesi, Battiston, Carabelli

- The creation of decks, cells, players and games has been left out of the Model UML diagram since these aspects will be addressed in the Control component.
  In particular:
  - Cards and Cells (our name for the squares that make up rooms) will be created by *factory* classes when the server is starting up and at the start of a new match, respectively
  - Players and Games will be created by the classes in charge of the "multi-match" management.
- We chose to take a functional approach to the modeling of weapons and power cards' actions. Every action will be defined as a λ-function and assigned to the correct instance of the card by the *factory* class in charge of spawning the whole deck.
  Following the tutor's suggestion, we decided to extend the functional approach to the application of cards' effects on the Player, by providing a public method which takes a `PlayerLambda` functional interface as an argument.
  The `PlayerLambda` instructs the Player instance on how to apply effects to itself.
- Information regarding Players' position is intentionally redundant:
  - Players have a `Point: position` attribute, which gives the correct indexes to access the Cell where the player is within the Map class
  - Cells have a `List<Player>: pawns` attribute, which contains the reference to every Player staying on the cell

  Both of these are necessary to avoid useless elaboration in situation which could arise from the lack of any of the two. In particular:
  - Without the attribute in the Cell class, `getPosition()` would have to be called on any Player in the game to figure out possible targets on that spot
  - Without the attribute in the Player class, it would be necessary to probe all the 12 cells to find its position