First, I will discuss about the some design decisions of my design.

For each Player class, it has an Action class, which means the Action class will record everything about the move of the Player of this round. This decision will lower the coupling of the diagram, because if Game class owns the Action class, its change will have effect on both player and action, but if we assign the action to the player, the change of game will only effect on the player. Then I assign all the validation work to the Board class and I forward the word dictionary to it too. The reason why I do this is: Board class has a matrix of Location, so we could get any information from Board class, like whether there is a tile on one location or if there is a special tile we may activate when we place the tiles. So it's very convenient and it's the responsibility of Board to make sure whether the action is ok. By doing this, I increase the cohesion of the design because I distribute the responsibilities correctly. Moreover, I use decorator pattern for the Location class in the design. That's because the Location class should have two major properties, they are Multipler Interface (Like double word or triple letter) and SpecialTile Interface, and each property has an expansion to sub properties, and because we may add more sub properties for SpecialTile in the future, we need to build the structure dynamically, so a decorator is the best solution for this. When we want to implement some effects from the Location, we just call the corresponding methods from the two interfaces.

Then for the validation part of the action, I don't implement lots of other validations like checking whether the new added tiles for the action are in one line or if the the locations of the new added tiles are overlapped with the existing tiles. I think they are the responsibilities of UI, the UI will have some restrictions like letting the player choose the start location first then choose the direction, so the players are forced to place the tiles in one line. And the UI for the board only opens those locations with no tiles, thus making it impossible for placing a new tile on an existing tile. And the selected start locations are only restricted to the positions near an existing tile (except for the first action), which makes the new word adjacent to the existing tiles. So for the UI, I think it may not only give a representation of Board, it should give some help to the normal operation of the game.

In the end, I will give short descriptions of the class I have made:

(1) Game class: the class which directly interact with players and other functional classes, it's like a dispatcher that can distribute the responsibilities to other parts.

(2) Controller class: the class that controls the order of the players, it can tell the Game class who the current player is and who's the next turn. And it can also implement skip() method for players or special tiles.

(3) Player class: the class represents the properties of a player, like having action or exchanging tiles.

(4) TilePackage class: the class represents the package of available tiles, it can give random number of tiles to refill the inventory of player or exchange tiles for the player.

(5) Board class: the class that has all the location information for the game board, it owns a matrix of Location objects. So it implements almost all the validation methods and placing work.

(6) Location class: the class stores all the location information in it. Moreover, it owns a tile object and two different interfaces, one is the Multipler interface and another is the SpecialTile interface. So it's a decorator which can apply different methods to the word or the action players have made.

(7) Action class: the class that represents the move of the players, it stores the start location, direction, tiles and special tile information of the current move of the players. It only related to Player class and will be validated by the Board class.

(8) Tile class: the class that represents the information of regular tile, like the letter and the value.
(9) Multipler interface: the interface represents the multipler cells on the board, it will have an effect on the value calculation of the word.

(10) SpecialTile interface: the interface that represents the properties of the special tile, different special tile has different effecting way. Some like Negative-points and Boom that will have an effect on the calculation of the value calculation of the words, and others like Reverse-player-order and OneMoreOrder will have an effect on the way of controlling of the Controller. So I think for the activation of the special tiles, I have better know it just after the action has been validated by the Board.

(11) Word class: the class that storing all the information of the words we have created from the validated action. So we will easily know how many scores the player has made for his/her action. The word class will be effected by the Multipler class.

(12) SpecialTileStore class: the class that sells special tiles to players.