Used Design Patterns

1. **Builder Pattern:**
   - Due to the many required and optional parameters that can be set to construct the Tile class, which lead to a large number of constructors. For this reason we used this pattern to unify it to 1 constructor and every time we need to construct a new Tile we call the builder to build the Tile object with setting optional parameters by need.
   - **Place in Code :**
     In Model.Tile- Builder,protected constructor and Init class used because Tile is superclass for subclasses “YellowTile” and “BlueTile “ so YellowTile and Blue Tile also contain builder Design pattern

2. **Factory Method Pattern:**
   - Since there are 5 different types/colors of tiles and and not all of them share the same class because some have additional required methods or fields (3 colors are from Tile class and there is YellowTile and BlueTile classes) using factory pattern helped with creating different colors tiles objects without worrying about what constructor to use in the different methods and what parameters should be set for each different color
   - **Place in Code :**
     Factory class:Colored Tiles Factory
     Used In: Model.Board.initiateBoardSecondaryColors()

3. **The Template Pattern:**
   - Soldier Piece and Queen Piece share part of the piece algorithms but in other parts they differ, so to spare us and the code from multiple checks every time a piece needs to take action we used **Template pattern** which translated in the code to abstract “Piece” class with methods for shared parts and abstract methods declaration that the extending classes “Soldier” and “Queen” had to implement for the different algorithms.
   - **Place in Code :**
     Abstract superclass: Model.Piece class
     Concrete subclasses: Model.Soldier, Model.Queen
     Used in : mainly in many methods of Model.Board and in some other separate places