Design Patterns

1. **Façade Pattern**

Facade pattern hides the complexities of the system and provides an interface to the client using which the client can access the system. This type of design pattern comes under structural pattern as this pattern adds an interface to existing system to hide its complexities.

We create a central class with the name Move Validation (façade class) that has the objects of all the Validations (Skip validation , ID validation , Distance Validation)

1. **Factory Pattern**

This type of design pattern comes under creational pattern as this pattern provides one of the best ways to create an object.

In Factory pattern, we create object without exposing the creation logic to the client and refer to newly created object using a common interface.

We create the Tile Factory class that help us to create the appropriate object to every Tile by checking the cols and rows to this Tile

By using this design pattern we encapsulate the creation of every tile and how we decide which color and value to get

1. **prototype pattern**

Prototype pattern refers to creating duplicate object while keeping performance in mind. This type of design pattern comes under creational pattern as this pattern provides one of the best ways to create an object.

This pattern involves implementing a prototype interface which tells to create a clone of the current object. This pattern is used when creation of object directly is costly. For example, an object is to be created after a costly database operation. We can cache the object, returns its clone on next request and update the database as and when needed thus reducing database calls.

We made the MakeCopy methods that return the copy to the object for in the Tile class and the subclasses to get the copy from every subclass

And that help us to copy the Tiles without to create new one and to get the same attributes to the old object