Padrões do Projeto Banco Imobiliário

Todo o código está dividido em pastas com seus respectivos padrões, possibilitando uma melhor organização e entendimento de onde os padrões foram implementados. Controller / Entidades / Fábrica / Interfaces- Controller / Interface-Entidade. Segue abaixo a justificativa do porque foi usado esses padrões.

Fábrica: Com o padrão Factory Method pode ser encapsulado o código que cria objetos. É muito comum ter classes que instanciam classes concretas e essa parte do código normalmente sofre diversas modificações, portanto nesses casos usamos um Factory Method que encapsula esse comportamento de instanciação. Com o uso do Factory Method tem o código de criação em um objeto ou método, evitando assim a duplicação e além disso tem um local único para fazer manutenção. O padrão também possibilita um código flexível e extensível para o futuro.

Controller: A camada de controle é responsável por intermediar as requisições enviadas pelo View com as respostas fornecidas pelo Model, processando os dados que o usuário informou e repassando para outras camadas. Numa analogia bem simplista, o controller operaria como o ‘’maestro de uma orquestra’’ que permite a comunicação entre o detentor dos dados e a pessoa com vários questionamentos no MVC. No projeto, a classe que ele (CONTROLLER) vai ter a sua própria entidade, fazendo operações com ela, como criando o tabuleiro. Na hora do método ser criado ele usa o padrão Fábrica para criação . Todos do Método controller são padrão singleton, pois só são instanciados uma vez.

Singleton: Garante que uma classe tenha uma instância no programa e forneça um ponto de acesso global para a mesma. Protege a instância com encapsulamento. No projeto, todos os métodos fábrica são Singleton, pois só precisa de uma instância, todas as classes estão separados por pastas, separando os padrões que usam fábrica/singleton.

Strategy: O Padrão Strategy sugere que se produza uma família de classes para cada variação do algoritmo e que se forneça para a classe hospedeira uma instância de Strategy para a qual ela delegou em tempo de execução. No projeto está sendo usado na hora de criar as cartas de título de propriedade, nas classes Cartas, dentro da Fábrica Cartas, pois vai facilitar nos cálculo dos aluguéis do título de propriedade. Também será usado esse padrão para criar as cartas de sorte e reverse.

Decorator: Agrega responsabilidades adicionais a um objeto dinamicamente. Os decorator fornecem uma alternativa flexível ao uso de subclasses para extensão de funcionalidades. No projeto, o padrão Decorator foi usado na criação das posições do tabuleiro, que são duas, com cartas(título de propriedade) e sem cartas(posições de ponto de partida, parada livre, prisão e vá para a prisão).