Documentação

1. Singleton: TeamRepository

Aplicabilidade: O padrão Singleton é aplicado quando é necessário garantir que uma classe tenha apenas uma instância e fornecer um ponto global de acesso a essa instância. No caso do ***TeamRepository***, ele é utilizado para garantir que todas as partes do sistema acessem e manipulem o mesmo conjunto de dados das equipes e pilotos.

Justificativa: A utilização do Singleton em ***TeamRepository*** é justificada pela necessidade de manter uma única fonte de verdade para os dados das equipes e pilotos. Como diversas partes do sistema precisam acessar e modificar esses dados, o Singleton garante que todas as operações sejam realizadas na mesma instância, evitando inconsistências.

2. Builder: TeamBuilder

Aplicabilidade: O padrão Builder é útil quando a construção de um objeto complexo é realizada por meio de um processo passo a passo. Ele permite a criação de diferentes representações de um objeto com o mesmo código de construção.

Justificativa: A utilização do Builder em ***TeamBuilder*** é justificada pela necessidade de criar objetos ***Team*** com várias etapas de configuração (adição de pilotos e carros). O Builder torna o processo de criação mais legível e flexível, permitindo a construção de equipes completas de maneira controlada.

3. Padrão Estrutural: Facade - SeasonManager

Aplicabilidade: O padrão Facade é usado para fornecer uma interface simplificada para um conjunto de interfaces em um subsistema. Ele define uma interface de nível mais alto que torna o subsistema mais fácil de usar.

Justificativa: A utilização do Facade em ***SeasonManager*** é justificada pela necessidade de simplificar a interação com o subsistema de gerenciamento de temporada. ***SeasonManager*** encapsula a complexidade das operações relacionadas à temporada (adicionar equipes, registrar corridas, atualizar especificações de carros) e fornece uma interface simplificada para o usuário.

4. Padrão Comportamental: Observer - CarSpecificationObserver

Aplicabilidade: O padrão Observer é utilizado para definir uma dependência entre objetos, de modo que, quando um objeto muda de estado, todos os seus dependentes são notificados e atualizados automaticamente.

Justificativa: A utilização do Observer em ***CarSpecificationObserver*** é justificada pela necessidade de monitorar as alterações nas especificações dos carros e gerar atualizações de notícias. O ***CarSpecificationObserver*** é registrado no ***TeamRepository*** e é notificado sempre que há uma atualização nas especificações dos carros, permitindo a geração de mensagens de notícias apropriadas.