APLICAÇÃO DO JOGADOR

ALEXANDRE TOMMASI
SILAS RODRIGUES
GABRIEL PIVOTO
LUIZ HENRIQUE PINA
MATHEUS AUGUSTO DE FARIA

Design Patterns

Criação:

- Singleton

Estrutural:

- Adapter
- Composite

Comportamental:

- Observer
- Command

Criação

Singleton

- O padrão Singleton garante que entidades como Partida ou Menu tenham apenas uma instância ativa, centralizando o controle de criação e acesso. Isso evita conflitos, assegura consistência no estado e oferece um ponto único de acesso para gerenciar essas instâncias de forma eficiente.

Estrutural

Adapter

- Função dele é transformar o JSON recebido do backend (outro grupo) no formato necessário para criar e exibir as cartas no frontend, assim trabalhar com objetos padronizados

```
{
    "card":{
        "foto": "string",
        "descricao": "string",
        "tipo":"string",
        "poder":"integer"
    }
}
```

```
public class Card
{
    public string Foto { get; set; }
    public string Descricao { get; set; }
    public string Tipo { get; set; }
    public int Poder { get; set; }

    public Card(string foto, string descricao, string tipo, int poder)
    {
        Foto = foto;
        Descricao = descricao;
        Tipo = tipo;
        Poder = poder;
    }

    public override string ToString()
    {
        return $"Foto: {Foto}, Descricao: {Descricao}, Tipo: {Tipo}, Poder: {Poder}";
    }
}
```

Composite

- Função dele é gerenciar e estruturar os menus e submenus, de forma que fique simples e intuitiva.
- Organizar objetos em estruturas de árvore representando as hierarquias dos objetos

Comportamental

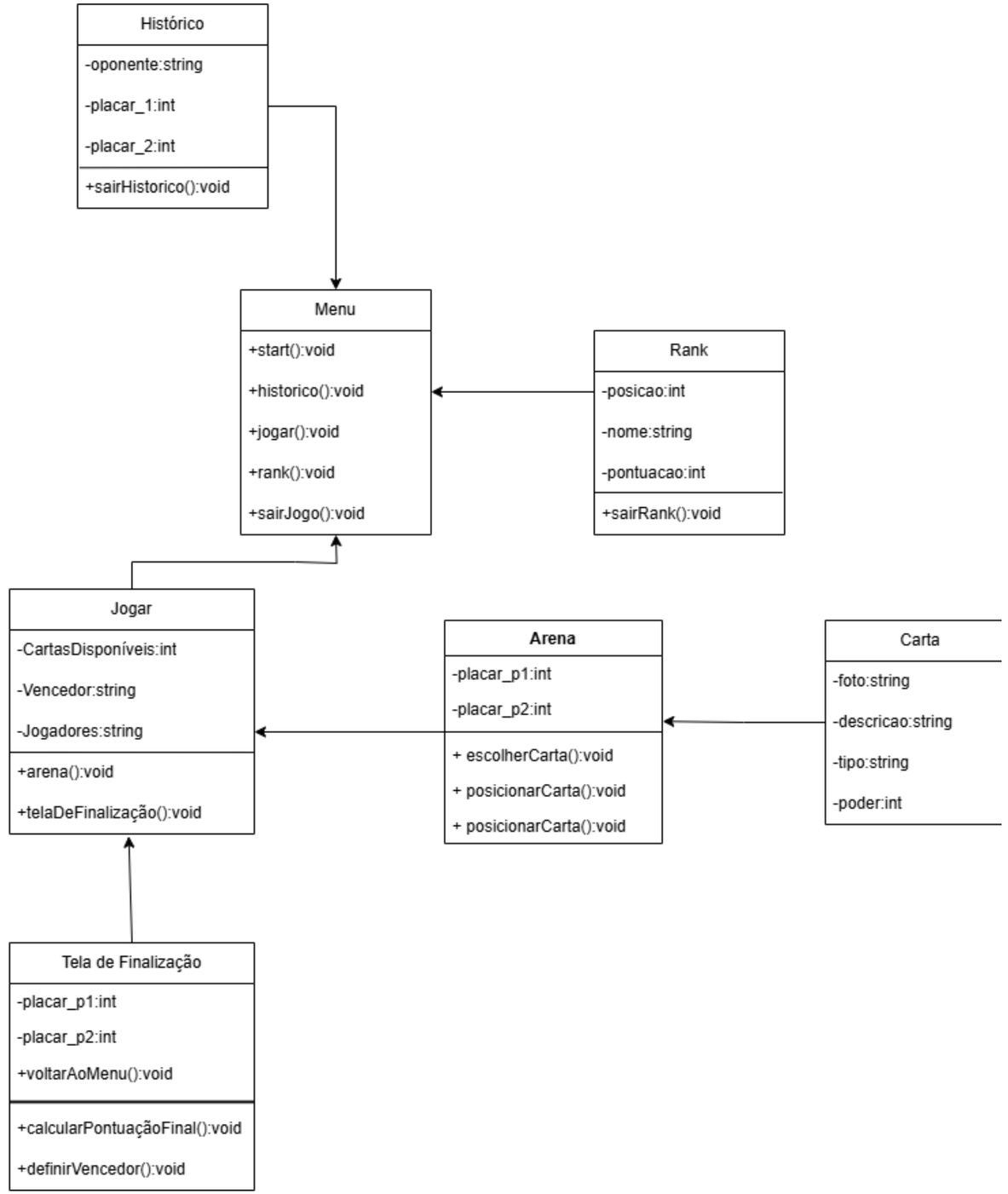
Observer

- O Observer é um padrão onde um objeto (sujeito) notifica automaticamente outros objetos (observadores) sobre mudanças no seu estado.
- Pode ser útil na classe Partida para notificar mudanças de estado, como o fim de uma partida ou atualização de pontuações. Assim, as instâncias de Jogador poderiam ser notificadas quando um evento ocorre.

Command

- O Command encapsula uma solicitação como um objeto, permitindo que comandos sejam executados, enfileirados ou desfeitos, separando a solicitação da execução.
- Aplicável na classe Menu para encapsular ações como iniciarPartida() e solicitarHistorico() em comandos. Isso permite a adição de novas ações ao menu sem modificar sua estrutura.

UML de Classes



Conclusão

- Singleton: Garante instâncias únicas para classes cruciais, como Partida e Menu, centralizando o controle e evitando inconsistências no estado do sistema.
- Observer: Facilita a comunicação entre Partida e Jogador, notificando eventos como fim de jogo ou atualizações de pontuação de forma automática e eficiente.
- Command: Encapsula ações no Menu (e potencialmente em outras partes do sistema), tornando-as flexíveis e escaláveis, com suporte para execução, desfazer e extensão de funcionalidades.
- Adapter: Converte dados JSON recebidos do backend em objetos utilizáveis no frontend, padronizando a integração e permitindo a manipulação consistente de Cartas.
- Composite: Estrutura menus e submenus em uma hierarquia clara e gerenciável, garantindo uma navegação intuitiva e modularidade no design.

MUITO OBRIGADO!