

**INF01120 - Técnicas de Construção de Programas**  
**Turma A (2025/2)**

# MatchUp

**Grupo 1**

Bruno Andrade Ramos  
Marcelo Gonda Stangler  
Matheus Cirio  
Tobias Cadoná Marion (Facilitador)

## Mudanças em Relação à Etapa Anterior

Houve três mudanças grandes em relação ao definido na etapa anterior:

1. **Remoção da classe “Place”:** Durante o desenvolvimento, percebemos que essa classe estava meio “jogada”, sem trazer nenhuma contribuição que justificasse sua permanência. Entretanto, numa versão futura do programa, nós pensaríamos em achar uma maneira de retornar com essa classe, de forma que ela complemente a aplicação;
2. **Uso de Generics:** Ao projetarmos o núcleo do nosso sistema de torneios, enfrentamos o desafio de manter uma alta coesão em um domínio complexo. Precisávamos que as regras matemáticas de pareamento (como Sistema Suíço ou Mata-mata) evoluíssem de forma independente das entidades que participam delas (Times ou Pessoas). Essa abordagem, baseada apenas em herança simples sem Generics, nos levaria rapidamente a um cenário de alto acoplamento e explosão combinatória de classes, no qual para cada tipo de competidor, teríamos que implementar versões específicas de cada algoritmo. Para resolver isso, adotamos o uso de Generics (`<T extends Competitor>`), permitindo isolar completamente a “mecânica do torneio” da “natureza do competidor”, tornando o modelo mais coeso, mais reutilizável;
3. **Utilização de Factory Methods:** Para garantir a integridade do domínio, impedindo, por exemplo, que um `Tournament<Competitor>` misture Times e Pessoas na mesma competição, utilizamos Factory Methods estáticos combinados a um construtor privado. Apesar de isso nos obrigar a criar e manter métodos de fábrica explícitos na classe `Tournament` para cada novo tipo suportado, ela é muito mais segura do que a alternativa e muito mais reutilizável. O resultado final é uma arquitetura onde o custo de extensão é baixo e a segurança de tipos é garantida pelo compilador, blindando as regras de negócio contra estados inválidos;
4. **Setters do Match:** Na implementação final, todos os setters foram removidos e substituídos pelo método único: “`public void updateResult(int newScoreA, int newScoreB, EventStatus newStatus)`”. Nele, foi possível concentrar a lógica de não permitir resultados empatados, placares negativos e a verificação/alteração do estado da partida.

---

## Implementação

### Visão geral da arquitetura:

A aplicação foi desenvolvida seguindo uma arquitetura clara e modular, separando responsabilidades entre:

- **Models:** Regras de negócio e entidades do domínio. Contém todas as classes do domínio, responsáveis por representar o estado e aplicar as regras do torneio.
  - **Competitor:** Interface base que define o contrato mínimo para qualquer competidor, permitindo que torneios funcionem tanto com pessoas quanto com times usando generics;

- **Person:** Representa participantes individuais;
- **Team:** Representa participantes individuais;
  - **Métodos:** addPlayer, removePlayer, getPlayers;
- **Match:** Classe que representa uma partida.
  - **Atributos Principais:** Competidores (A e B), Pontuação (A e B), Status;
  - **Método Principal:** updateResult. Concentra as lógicas que foram citadas na etapa anterior.
- **Tournament:** Classe principal do domínio;
  - **Atributos Principais:** Nome, Lista de Participantes, Lista de Rodadas, Status, Pairing;
  - **Método Principal:** generateNextMatches (delega a geração de rodadas para o pairing selecionado).
- **Services:** Lógica de aplicação e orquestração, intermediário entre Models e Views;
  - **Pairing:** Interface que define a estratégia de geração de rodadas;
    - **Tipos:** League (Pontos Corridos), Knockout (Mata-Mata) e Swiss (Suíço).
  - **League (Pontos Corridos):** Gera todas as rodadas de um round-robin (simples ou duplo) de uma só vez, de forma que o mesmo jogador/time não se enfrente;
  - **Knockout (Mata-Mata):** Gera rodadas eliminatórias, com um número de participantes que seja potência de 2. Só gera a próxima rodada quando a atual está 100% finalizada;
  - **Swiss (Suíço):** Gera rodadas em formato suíço, com um número de participantes igual a 16 ou 32 (esta decisão foi tomada devido a elevada complexidade de controlar e organizar com outros números de participantes).
  - **TournamentService:** Cria torneios, persiste (armazenando em memória) e retorna torneios para a interface;
  - **CompetitorService:** Mesma ideia do **TournamentService**, mas para competidores;
- **View (JavaFX):** Interface gráfica e interação com o usuário. Responsável por todas as telas do sistema, desde a criação de um novo torneio até a criação de jogadores/times. Controlada pelo respectivo **controller**, que é quem “reage” às interações do usuário e outros comportamentos do sistema.

---

## Testes

De forma geral, os testes foram bem úteis e nos ajudaram a detectar rapidamente falhas na lógica e na implementação da aplicação. A seguir, estão listados os classes que foram testadas e quais foram os testes realizados:

- **CompetitorStatsText:** Inicialização com 0s, cálculo correto tanto para vitórias quanto para derrotas e desempate por saldo de gols;
- **MatchTest:** Criação de forma correta para os dois tipos de competidores, validação do resultado (não pode empates e nem pontuação negativa), alterações disponíveis enquanto o jogo está sendo realizado e atribuição correta da pontuação;

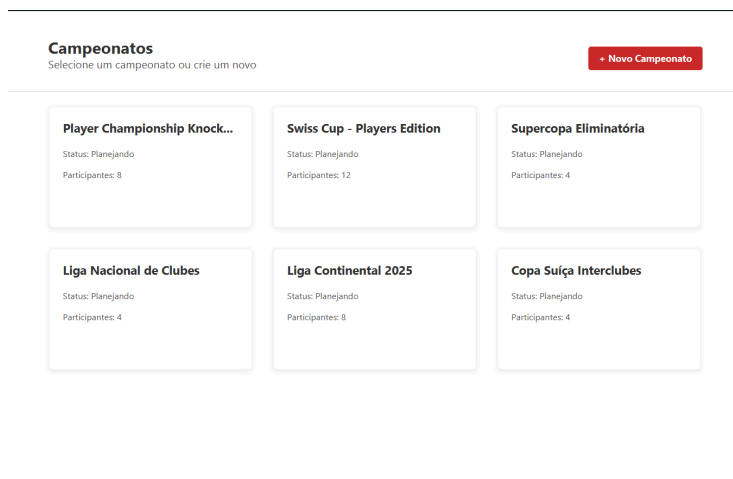
- **TournamentTest:** Criação correta para ambos os tipos de competidores, validação da função generateNextMatches e atualização correta de status;
- **CompetitorServiceIMTest/TournamentServiceTest:** Validação de adição, remoção e atualização de competidores/torneios e retorno correto de todos eles;
- **KnockoutTest/LeagueTest/SwissTest:** Validação da quantidade de jogadores permitidas por cada um deles, além da geração das rodadas e criação de novas rodadas de forma correta e apenas se todas as partidas da rodada atual foram finalizadas;
- **LeagueStandingCalculatorTest:** Atualização das estatísticas apenas no momento correto, ou seja, quando a partida é finalizada;

---

## Executável

### Menu Inicial:

Visão geral dos campeonatos existentes e botão para a criação de novos campeonatos



---

### Novo Campeonato:

Criação de um novo campeonato, com o preenchimento de suas informações básicas

**Novo Campeonato**  
Preencha os dados do novo campeonato

Nome do Campeonato:

Tipo de Competição:  Formato:  Tipo de Competidor:

Data de Início:

Gerenciar Jogadores:

Adição de jogadores ao campeonato, com o preenchimento de suas informações básicas. É possível adicionar jogadores já presentes em outros campeonatos

Gerenciar Participantes

Gerenciar Jogadores

Nome: CPF: Telefone: Nascimento:

Adicionar Novo

Nome	CPF	Telefone	Nascimento
Não há conteúdo na tabela			

Jogador existente: Adicionar Remover Selecionado

0 jogadores adicionados

Cancelar Finalizar

Gerenciar Times:

Adição de times do campeonato, com a adição de jogadores. É possível adicionar times e jogadores (“Gerenciar Jogadores”) já presentes em outros campeonatos

Gerenciar Participantes

Gerenciar Times

16 times adicionados

Nome do time: Adicionar Novo

Time	Qtd.	Jogadores
1	1	Letícia Duarte
2	0	
3	0	
4	0	
5	0	
6	0	
7	0	
8	0	
9	0	

Time existente: Adicionar Remover Selecionado

Gerenciar Jogadores Cancelar Finalizar

Editar Partida:

Atribuição do resultado das partidas

×

Editar Partida

1

-

0

+

×

2

-

0

+

Status

A confirmar

Em andamento

Finalizada

Cancelar

Salvar

Editar Times:

Ao clicar no lápis, ao lado do número de participantes, abre uma tela onde é possível editar os jogadores cadastrados no campeonato (antes do seu início)

Copa Suíça Interclubes

Status: Planejando

INFORMAÇÕES GERAIS

4 participantes

Gerar Rodadas

Voltar aos Campeonatos

Editar Participantes

Gerenciar Times

4 times adicionados

Nome do time: 

Adicionar Novo

Time	Qtd.	Jogadores
Sunset Tigers	3	Alice Johnson, Bob Martinez, Carla Ribeiro
North Dragons	3	Daniel Costa, Evelyn Tanaka, Frank O'Connor
Iron Wolves	3	Gustavo Lima, Hana Suzuki, Igor Petrov
Crimson Phoenix	3	Julia Mendes, Kevin Hart, Letícia Duarte

Time existente: 

Adicionar

Remover Selecionado

Gerenciar Jogadores

Cancelar

Finalizar

### Pontos Corridos - Rodadas:

### Apresentação de todas as rodadas do campeonato

Campeonato Pontos

Corridos

Status: Em andamento

INFORMAÇÕES GERAIS

4 participantes

Rodadas

Classificação

< Anterior

Próxima >

Rodada 1

1

-

Planejando

2

-

3

-

Planejando

4

-

Voltar aos Campeonatos

**Pontos Corridos - Classificação:**

### Apresentação da classificação geral do campeonato

Campeonato Pontos Corridos

Status: Em andamento

INFORMAÇÕES GERAIS

4 participantes

Rodadas

Classificação

Classificação / Resultados

#	Time	PTS	J	V	D	GP	GC	SG
1	2	0	0	0	0	0	0	0
2	3	0	0	0	0	0	0	0
3	1	0	0	0	0	0	0	0
4	4	0	0	0	0	0	0	0

Voltar aos Campeonatos

Suíço - Rodadas:

Apresentação de todas as rodadas do campeonato

**Campeonato Suíço**  
Status: Em andamento

INFORMAÇÕES GERAIS

16 participantes

Voltar aos Campeonatos

Rodadas

Classificação

< Anterior

Próxima >

Rodada 1

1

10

-

Planejando

11

12

-

Planejando

13

14

-

Planejando

15

16

-

Planejando

2

3

-

Planejando

Suíço - Classificação:

Apresentação da classificação geral do campeonato

**Campeonato Suíço**  
Status: Em andamento

INFORMAÇÕES GERAIS

16 participantes

Mostrar Resultados

Voltar aos Campeonatos

Rodadas

Classificação

Classificação / Resultados

3 – 0

4

14

3 – 1

5

6

13

3 – 2

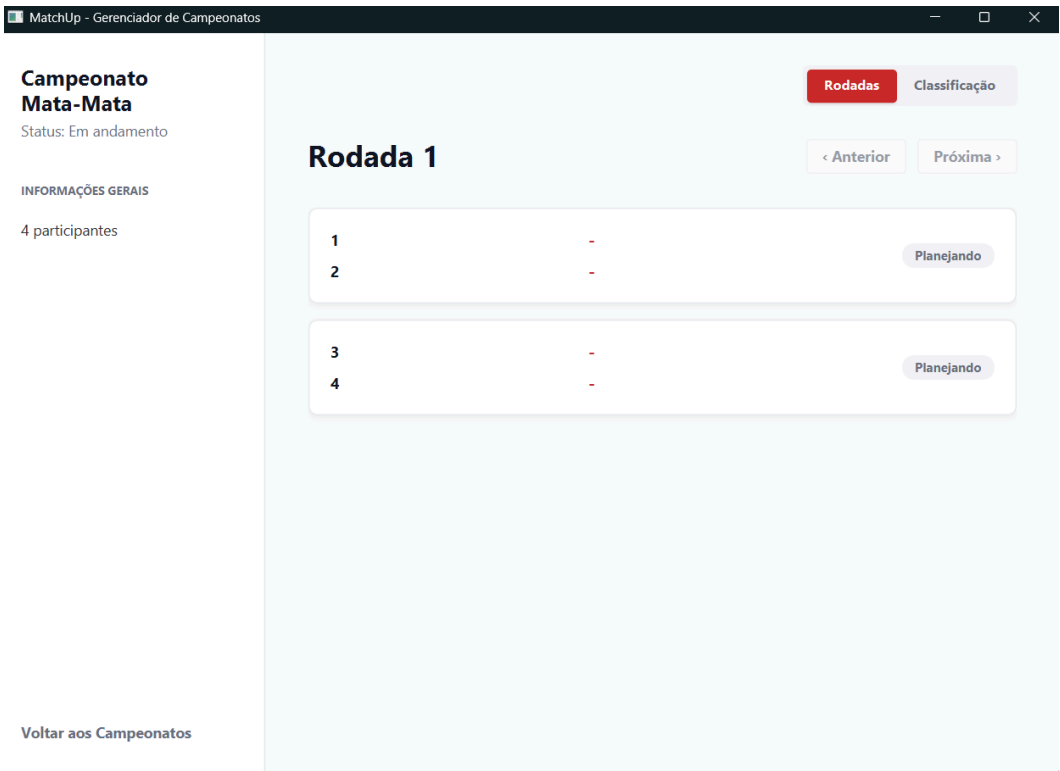
8

12



Mata-Mata - Rodadas:

Apresentação de todas as rodadas do campeonato



Mata-Mata - Classificação:

Apresentação da classificação geral do campeonato

