# EP 1 — Onitama

Prof. Flávio Luiz Coutinho

ACH2003 — Computação Orientada a Objetos

## 1 Introdução

*Onitama* é um jogo de tabuleiro de 2 jogadores. Cada jogador reveza a vez para jogar cartas de movimento que irão mover suas peças. O objetivo é derrotar o adversário em uma disputa de artes marciais. Neste divertido jogo, jogadores precisam pensar em estratégias para qual movimento realizar na hora certa, defendendo seu Templo e seu Mestre. Saber quando ser ofensivo ou defensivo é a chave!


**Figura 1:** Imagens de `https://www.arcanewonders.com/product/onitama/`.

Por mais que seja muito divertido falar sobre jogos de tabuleiro, eu e você sabemos qual é realmente a melhor parte: o Enunciado do EP!

## 2 Elementos e regras do jogo

Uma partida do jogo Onitama ocorre em um tabuleiro de dimensão $5 \times 5$, e deve respeitar as seguintes regras/condições:

- O espaço central das linhas superior e inferior do tabuleiro é denominado *Templo*. Cada Templo está associado a um jogador, e iremos assumir que o Templo da linha superior terá a cor azul, enquanto o da linha inferior terá a cor vermelha.
- Cada jogador possui 5 peças: um Mestre e 4 alunos;

- Na configuração inicial do tabuleiro, todas as peças do jogador azul ocupam a linha superior do tabuleiro, enquanto as peças do jogador vermelho a linha inferior, com os mestres ocupando seus respectivos Templos;
- O Mestre funciona da mesma forma que as peças de aluno, a única diferença é que serve para condições de vitória;
- Cada jogador também possui duas cartas de movimento na mão;
- Uma carta de movimento é escolhida para ser a carta da mesa. Essa carta possui uma cor, que determinará o jogador que começará a partida;
- Ou seja, em cada partida, terá apenas 5 cartas em jogo: duas na mão de cada jogador e uma na mesa.
- Cartas de movimento indicam para quais posições uma peça pode se mover;
- Uma jogada consiste em "aplicar" uma carta a uma peça e movê-la para uma posição disponível de acordo com as possibilidades definidas na carta. Ao realizar um movimento, uma peça não pode se mover para fora do tabuleiro e nem para um espaço contendo uma peça da mesma cor;
- Caso uma peça se mova para um espaço contendo uma peça da cor do oponente, a peça que estava no espaço é eliminada;
- Após utilizar uma carta, o jogador deve trocar a carta recém-utilizada pela carta da mesa. Assim, a carta que antes era da mesa vai para a mão do jogador; e a carta recém-utilizada vai para a mesa;
- Após uma jogada, na qual é feito um movimento, o jogador passa a vez para seu oponente;
- O jogo termina quando uma destas condições de vitória é atendida:
  - Um dos Mestres é capturado;
  - Um dos Mestres ocupa o Templo adversário.

O livro completo com as regras do jogo é disponibilizado publicamente pelo seu editor (em inglês) em `https://www.arcanewonders.com/wp-content/uploads/2021/05/Onitama-Rulebook.pdf`.

## 2.1 Tabuleiro

O jogo começa com a configuração inicial do tabuleiro ilustrada pela Figura 2.

As posições do tabuleiro são indexadas da mesma forma como indexamos usualmente os elementos de uma matriz. O canto superior esquerdo possui coordenadas 0,0. Adotamos como convenção de que as peças azuis começarão o jogo na parte de cima do tabuleiro (linha 0), enquanto que as peças vermelhas começarão em baixo (linha 4).

## 2.2 Carta

A Figura 3 ilustra dois padrões de movimentos diferentes possibilitados por cartas de movimento.

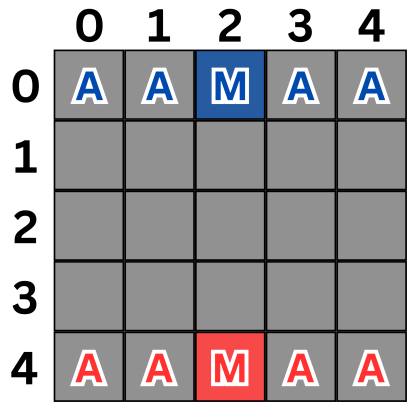


**Figura 2:** Tabuleiro na configuração inicial do jogo.

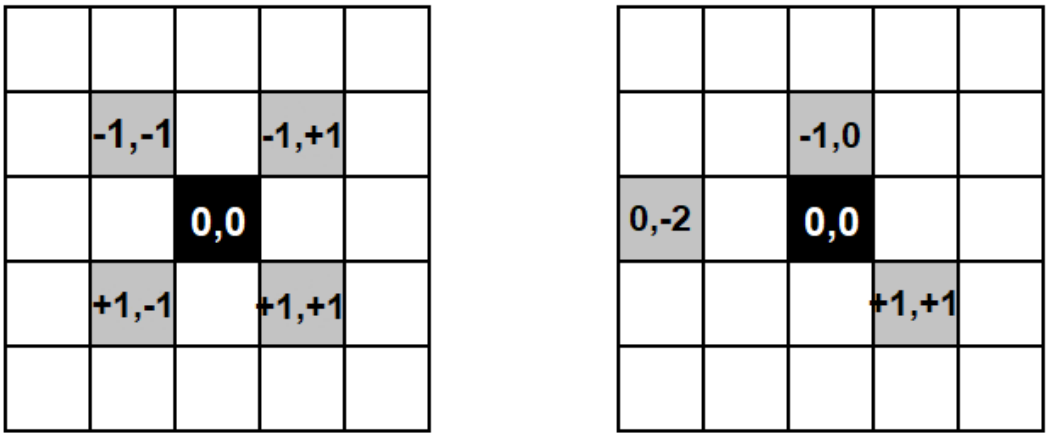


**Figura 3:** Exemplos de movimentações possíveis para uma peça dada por duas cartas de movimento.

O centro (em preto) representa a posição atual da peça que esta carta está movendo. As outras posições marcadas (em cinza) indicam a posição para onde a peça pode se mover em relação ao centro pintado de preto.

O jogo original possui 16 cartas de movimentações ao todo. Sem perda de generalidade, para simplificar o trabalho de codificação, iremos trabalhar com um baralho restrito a 8 cartas de movimento. As cartas estão indicadas na Figura 4.

# 3 Enunciado

Ao ler a emocionante descrição acima sobre Onitama, você se empolgou muito mesmo e decidiu utilizar seu vasto conhecimento de Orientação a Objetos para implementar as regras do jogo em Java: que divertido!

Junto com o enunciado, uma implementação incompleta do jogo foi disponibilizada. O objetivo do EP é completar a implementação do jogo, entendendo a documentação do código incompleto e fornecendo a implementação dos métodos faltantes (marcados com **TODO**) e da classe `GameImpl` que implementa a interface Game. Usando a implementação da interface, será possível jogar Onitama em Java, ou seja, sua implementação deve garantir que as regras são seguidas ao chamar os métodos da interface **`Game`**.


**Figura 4:** Imagem de `https://www.ultraboardgames.com/onitama/game-rules.php`.

`GameImpl` deve conter 3 construtores:

- Um construtor que recebe nenhum parâmetro;
- Um construtor que recebe duas Strings como parâmetro, representando os nomes dos jogadores de Peças Vermelhas e Azuis, **respectivamente**;
- Um construtor que recebe duas Strings como parâmetro, representando os nomes do jogador de Peças Vermelhas e Azuis, **respectivamente**. Adicionalmente, o construtor deve receber também um Array de Cards. Este Array de Cards terá um Deck totalmente novo, com cartas que não estão nesse enunciado. Sua implementação deve selecionar 5 cartas aleatórias desse novo deck passado como parâmetro do construtor, ao invés de usar o deck criado no método `createCards()`

Sua implementação será usada para a simulação de uma partida e o seu funcionamento será usado para a avaliação deste EP.

**Atenção**: Para este EP, não é necessário implementar o jogo em si ou um algorítimo que jogue o jogo, mas sim implementar as regras do jogo, ou seja, a interface **`Game`** e todos os métodos das classes fornecidas pelo enunciado.

As classes fornecidas que auxiliarão no desenvolvimento do jogo são:

- **`Card`**: Classe que contem informações sobre cada cartas.
- **`Color`**: `Enum` contendo todas as possíveis cores do jogo.
- **`Game`**: Interface que contém métodos que serão chamados para a execução do jogo.
- **`Piece`**: Classe que contém informações sobre as peças de jogo.
- **`Player`**: Classe que contém informações e ações básicas relacionadas aos jogadores.
- **`Position`**: Classe usada para definição de estrutura de posições e movimentos do jogo.
- **`Spot`**: Classe contendo ações e informações sobre cada espaço (quadrado) no tabuleiro.
- **`OnitamaGameException`**: Exceção geral para qualquer regra burlada.
- **`IllegalMovementException`**: Exceção para quando se tenta fazer um movimento inválido.
- **`IncorrectTurnOrderException`**: Exceção para quando se tenta jogar fora do seu turno.
- **`InvalidCardException`**: Exceção para quando se tenta jogar uma carta que não está na mão do jogador.
- **`InvalidPieceException`**: Exceção para quando se tenta mover uma peça que não está em jogo.

Todas essas classes possuem métodos que devem ser implementados. Mais detalhes sobre o que cada método deve fazer podem ser encontrados na documentação das classes.

## 3.1 O que é esperado

- Implementação da interface **`Game`** com o nome `GameImpl.java`;
- Implementação de todos os métodos (a não ser que explicitamente tidos como opcionais) nas classes fornecidas conforme descrito na documentação gerada com o `javadoc`;
- Implementação de atributos nas classes fornecidas de modo que façam sentido no contexto de cada uma;
- Implementação das exceções (*exceptions*) fornecidas;
- Um arquivo texto chamado `LEIAME.txt` com o nome e número USP dos autores e com instruções de como compilar e executar o seu programa em um terminal de linha de comando.

## 3.2 O que é permitido

- Criar novas classes e métodos de auxílio e uso interno da classe;
- Criar uma classe para execução (*Main*) das funcionalidades;
- Utilizar conteúdos ainda não vistos em aula (como estrutura de dados, por exemplo);
- Criar testes unitários utilizando JUnit (veja a Seção 6).

### 3.3 O que não é permitido

- Alterar a assinatura dos métodos e classes fornecidas (modificadores de acesso, nome, parâmetros, tipo de retorno, etc);
- Utilizar bibliotecas de terceiros que exigem instalação adicional.

## 4 Avaliação

A avaliação levará em consideração os seguintes pontos:

- Funcionamento geral do jogo;
- Clareza de código (código de qualidade é código legível!).

Será avaliado o código implementado na classe `GameImpl` (que implementa a interface **`Game`**) e todos os métodos que estão faltando nas classes passadas no enunciado. Você pode criar novas classes e implementar outros métodos auxiliares, mas a avaliação será feita com o uso dos métodos especificados pelo enunciado.

Casos de plágio ou, de forma mais geral, qualquer desrespeito ao *Código de Ética da Universidade de São Paulo*[1] acarretarão em nota 0 (zero) para todos os integrantes do grupo, além das demais consequências previstas pelo Regime Disciplinar[2] da universidade.

## 5 Grupos

Este Exercício-Programa deve ser feito em grupos de 2 ou 3 alunos.

## 6 Bônus

Exercícios-Programa entregues com um conjunto de testes unitários usando JUnit[3] poderão receber até 1 ponto adicional de bônus (a depender da qualidade e abrangência dos testes).

## 7 Entrega

A entrega deve ser feita no eDisciplinas (`https://edisciplinas.usp.br/`) **até às 23h59 do dia 11 de junho de 2023** por apenas um dos alunos do grupo.

Entregue todos os arquivos necessários para compilar o seu EP em um arquivo compactado `.tar.xz`, `.tar.gz` ou `.zip`.

Junto com os arquivos-fonte, inclua um arquivo texto chamado `LEIAME.txt` com o nome e número USP dos autores e com instruções de como compilar e executar o seu programa em um terminal de linha de comando. O seu programa **não pode** depender de uma IDE, editor ou qualquer programa externo ao JDK para ser compilado e executado.

[1] `https://leginf.usp.br/?resolucao=resolucao-no-4871-de-22-de-outubro-de-2001-3`
[2] `http://www.prg.usp.br/wp-content/uploads/manual_disciplinar_web2.pdf`
[3] https://junit.org

Se o seu programa implementar o bônus descrito na Seção 6, inclua a biblioteca do JUnit (arquivos `.jar`) no arquivo compactado. Assegure-se que as instruções sobre como compilar e executar presentes no `LEIAME.txt` incluam a biblioteca do JUnit no `CLASSPATH`.