Trabalho MC322

Etapa 02

Jogo de Damas

O seu desafio aqui é escrever um conjunto de classes que simule o funcionamento de um jogo de damas. Este jogo tomará como base o jogo Resta Um e deve seguir duas premissas do mesmo:

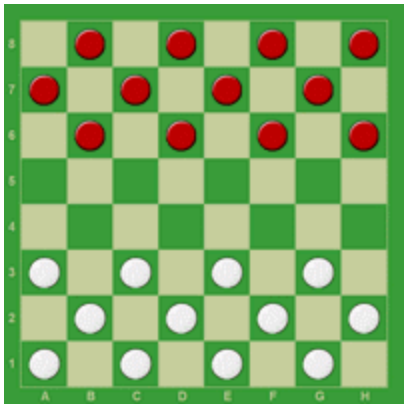
- defina pelo menos uma classe cujo objeto represente o tabuleiro;
- defina uma classe para representar cada tipo de peça do tabuleiro.

Como o jogo de damas tem dois tipos de peça - a peça comum (peão) e a dama - deve haver uma classe para cada uma delas. A requisição de movimento será feita ao tabuleiro, que repassa a requisição à peça. A peça verificará se o movimento é compatível com o seu tipo (peça comum ou dama) e se o movimento é possível no tabuleiro. Para verificar se o movimento é possível, a peça receberá do tabuleiro informações sobre o trajeto que deve percorrer.

Se a peça verificar que é possível, ela atualiza seu estado interno e retorna para o tabuleiro um boolean true que valida a movimentação pelo o tabuleiro. O tabuleiro executa o movimento, retirando a peça da origem e colocando no destino.

A verificação de captura pode ser feita pela peça ou pelo tabuleiro (você deve escolher), mas é o tabuleiro quem executa a remoção da peça.

- Tabuleiro:
  - o tabuleiro tem 64 casas (8 linhas e 8 colunas);
  - cada lado tem 12 peças, posicionadas conforme a ilustração:



- Lance:
  - o jogo está organizado em rodadas (lances) em que os oponentes se revezam em movimentos no tabuleiro;

- Movimento:
  - peças comuns se movem uma casa na diagonal para frente somente se a posição estiver livre e for o seu lance;
  - damas se movem qualquer número de casas na diagonal para frente ou para trás, contanto que a diagonal esteja desimpedida e seja o seu lance.
- Transformação em dama: quando uma peça comum atinge o lado oposto do tabuleiro ela se transforma em uma dama.
- Captura da oponente:
  - quando uma peça é capturada ela sai do tabuleiro;
  - cada vez que uma peça comum tem uma peça da cor oponente na sua casa vizinha diagonal (para frente ou para trás), seguida de uma casa vazia na mesma diagonal (destino), ela deve capturar a oponente se for seu lance - a captura envolve o movimento para o destino;
  - cada vez que uma dama tem uma peça da cor oponente (peça alvo) na sua diagonal (para frente ou para trás), havendo pelo menos um espaço livre na mesma diagonal depois da peça alvo (destino), ela deve capturar a oponente se for o seu lance - o caminho entre a origem e destino do movimento da dama só pode ter a peça alvo e a dama se move para o destino na captura;
  - mais de uma captura pode ser feito no mesmo lance.

Os movimentos a serem jogados no tabuleiro serão recebidos a partir da chamada de um método de um objeto, cuja classe já está codificada (veja detalhamento abaixo).

Tabuleiro:

O estado inicial do tabuleiro e seu novo estado depois de cada movimento deve ser mostrado no console na forma de caracteres, como ilustrado a seguir:

```
8 - p - p - p - p
7 p - p - p - p -
6 - p - p - p - p
5 - - - - - - - -
4 - - - - - - - -
3 b - b - b - b -
2 - b - b - b - b
1 b - b - b - b -
  a b c d e f g h
```

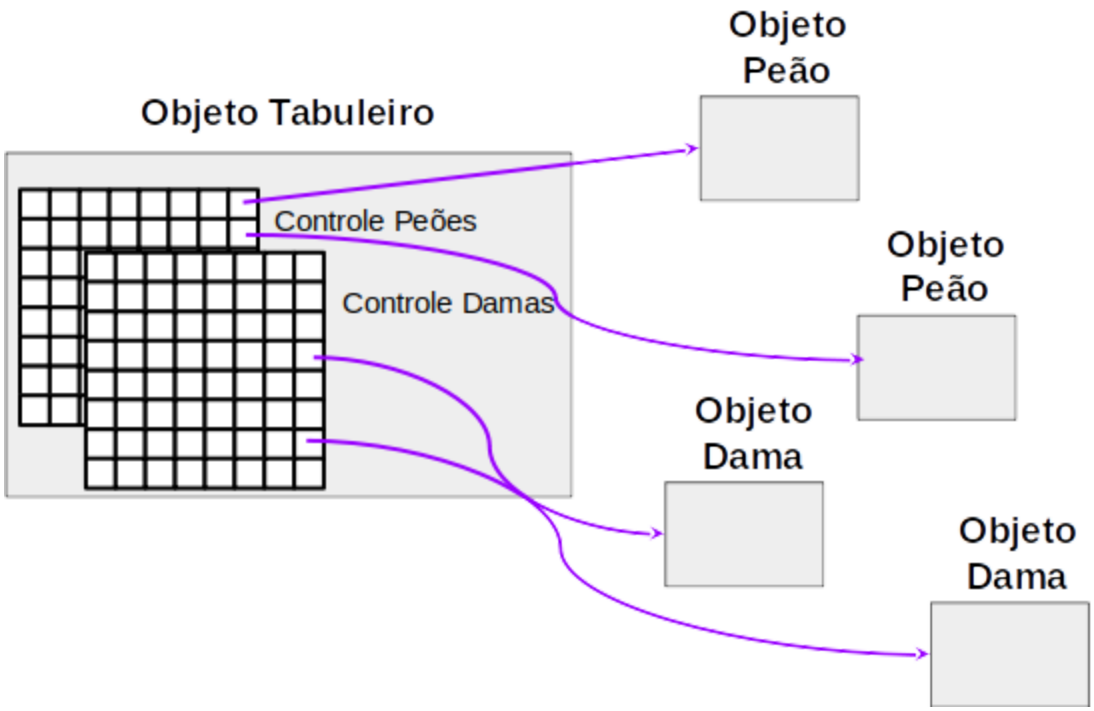
Os caracteres minúsculos p representam as peças pretas, b são as brancas e - é uma casa vazia. Utilize as maiúsculas (P ou B) para representar as damas.

A ilustração também mostra o estado inicial do tabuleiro.

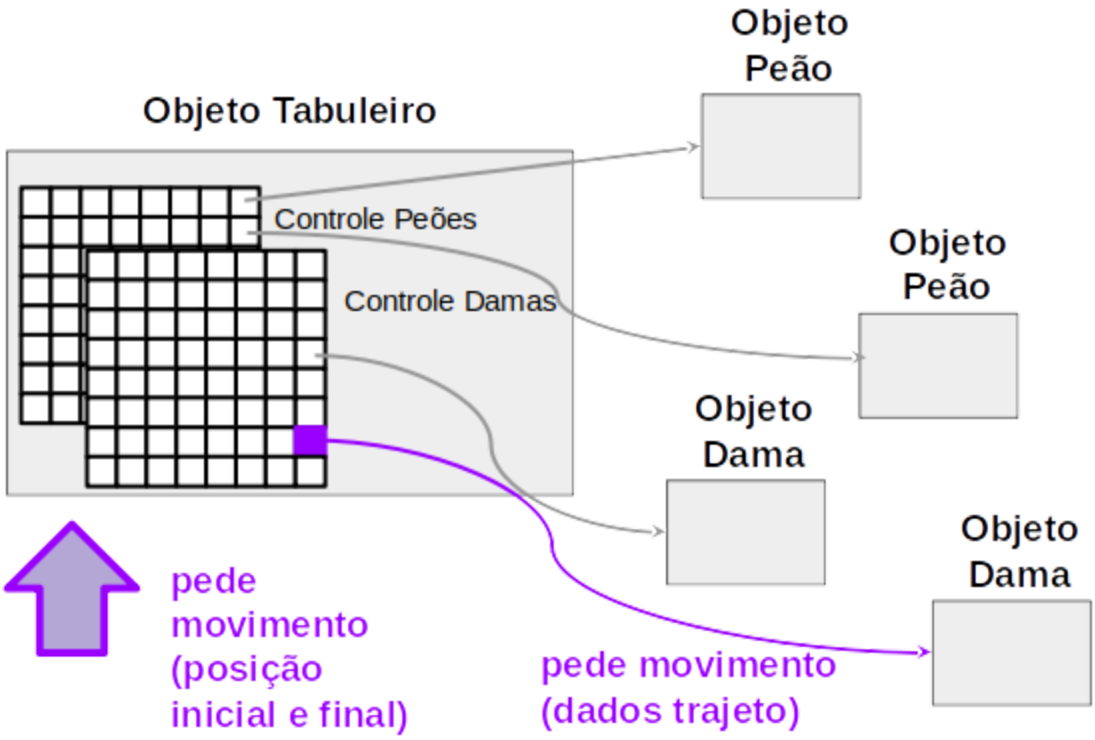
O tabuleiro deverá ser apresentado em caracteres da forma mostrada acima. As posições do tabuleiro serão descritas utilizando as colunas de **'a'** até **'h'** e as linhas de **1** até **8**, onde uma posição é dada combinando coluna com linha, por exemplo, **c4** consiste na coluna **c** e na linha **4**.

As três figuras a seguir mostram a arquitetura proposta e como o tabuleiro se comunica com a peça.

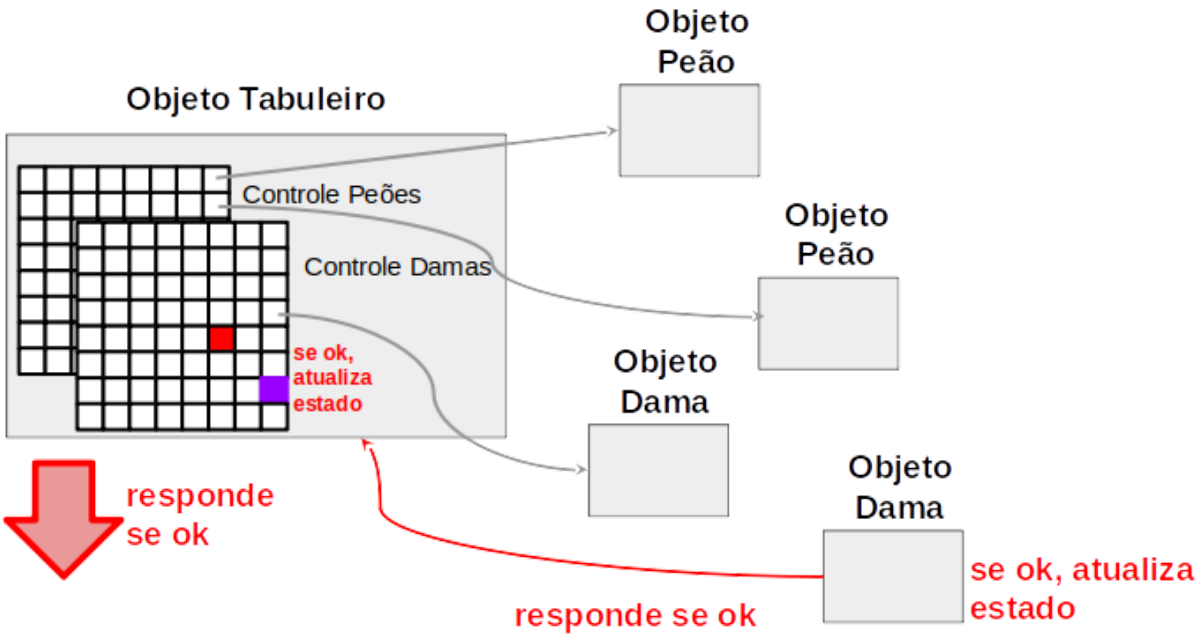
Esta é a arquitetura geral conforme descrito anteriormente. Considerando que será usada uma matriz para se controlar a posição dos objetos e há duas classes, sugere-se uma matriz para controlar os peões e outra para as damas. O estado do tabuleiro é a sobreposição das duas matrizes.



O tabuleiro recebe a solicitação de movimentação, verifica qual a peça envolvida e monta o trajeto da origem para o destino. A forma como o trajeto deve ser implementado fica a seu critério. O tabuleiro envia a solicitação de movimento para a peça (de acordo com a classe).



A peça verifica se o movimento é possível. Se for, atualiza seu estado. Ela retorna para o tabuleiro a informação se é possível. Se for possível, o tabuleiro atualiza seu estado e dá o retorno para quem solicitou o movimento.



Entrada do Programa:

A entrada do programa será um arquivo `.csv` contendo todos os comandos a serem executados pelo jogo. Cada comando consistirá de uma posição inicial (posição da peça a ser movida) e uma posição final (posição para onde a peça selecionada será movida). No arquivo `.csv` os comandos serão separados por vírgulas, ou seja, cada comando, contendo posição inicial e final, será separado por vírgula do próximo. Está sendo disponibilizado uma classe (CSVReader) para ler esse arquivo .csv e retornar a entrada pronta num vetor de String, onde cada posição desse vetor consiste da posição inicial e da posição final separadas por ":".

Exemplo de uma posição do vetor contendo as entradas: **f4:d4**. Nesse exemplo, a posição inicial é a coluna **f** e a linha **4** e a posição final é a coluna **d** e a linha **4**.

A entrada usará o mesmo CSVReader da etapa anterior do trabalho.

Saída do Programa:

A saída do programa deve ser mostrada no console e segue o mesmo princípio da etapa anterior do trabalho. Deve ser mostrada a posição inicial (source) e final (target) da peça que vai ser movimentada na rodada, bem como o estado do tabuleiro após a movimentação. Antes da primeira movimentação mostre o estado inicial do tabuleiro.

Além da classe CSVReader, deverá ser criada a classe **AppDama** que estará dentro do pacote *mc322.lab05*. Essa classe deve conter um método estático chamado *executaJogo* que terá como único parâmetro uma String contendo o caminho do arquivo .csv a ser lido. O método irá imprimir no console os estados do jogo (como descrito anteriormente) e retornar um vetor de Strings contendo o estado do tabuleiro a cada etapa do movimento (como explicado no laboratório anterior).

Entrega

A entrega deve ser realizada em duas etapas:

- No checkpoint você deve zipar o código e enviar via Classroom.
- A entrega final será via Github, conforme template.