MC322/MC336
Segundo semestre de 2015

Laboratório 3

Professor: Fábio Luiz Usberti (fusberti@ic.unicamp.br)
PED: Rafael Arakaki (rafaelkendyarakaki@gmail.com)

---

1 Objetivo

O objetivo deste laboratório será a familiarização com as classes e métodos disponibilizados no Trabalho 1 e a criação de um pequeno Jogador com procedimentos bem específicos.

2 Atividade

Nesta atividade o aluno deverá criar uma pequena classe Jogador restrita que irá executar jogadas específicas.

Para facilitar, crie um novo projeto no Eclipse chamado “Lab3”. Siga as instruções abaixo para importar os arquivos .class e a documentação para seu projeto.

Procedimentos para montar o projeto:

1. Crie um novo projeto no Eclipse.
2. Baixe o pacote **Motor-T1-V2.zip** disponibilizado no Ensino Aberto.
3. Crie uma nova classe neste projeto chamada "Main", no pacote default (não especificar nada no campo ’package’).
4. Copie o conteúdo do arquivo disponibilizado src/Main.java para ser o conteúdo da classe "Main"recém criada.
5. Crie uma nova classe neste projeto chamada "JogadorRAxxxxxx"(substituindo “xxxxxx” pelo seu RA), no pacote default.
6. Copie o conteúdo do arquivo disponibilizado src/JogadorRAxxxxxx.java para ser o conteúdo da classe "JogadorRAxxxxxx"recém criada.
7. No arquivo "Main.java"do seu projeto, altere as linhas 68 e 131 e substitua “xxxxxx” pelo seu RA.
8. Neste momento irão aparecer diversos erros e warnings no Eclipse.
9. Clique com botão direito no seu projeto, selecione as propriedades, vá em "Java Build Path -> Add External Class Folder", e adicione a pasta "bin"disponibilizada neste pacote.
10. Clique na setinha ao lado esquerdo do item que foi criado (logo abaixo da frase "JARs and class folders on the build path").
11. Escolha a opção Javadoc location, navegue até a pasta "doc"disponibilizada neste pacote e selecione-a.
12. Dê OK e feche as janelas.

Então você deve ter os arquivos Main.java e JogadorRAxxxxxx.java no projeto. Note que deve-se renomear a classe do jogador **JogadorRA**xxxxxx substituindo "xxxxxx" pelo seu RA (renomeie o arquivo também). A partir destas mudanças já deverá ser possível compilar e executar o projeto, porém a classe JogadorRA não realizará nenhum movimento e irá perder o jogo para a classe Jogador Aleatório.

Neste laboratório será requerido que o aluno implemente alguns procedimentos bem específicos. **Atente-se a não fugir das especificações contidas neste enunciado**. Implementações que funcionarem porém não seguirem as especificações receberão nota no máximo parcial na correção da atividade.

O seu Jogador deve criar, em ordem, no ArrayList<jogada> que é devolvido durante o método processarTurno(), as seguintes Jogadas:

1. Se **houver mana disponível** para alguma carta de Magia, deverá utilizar esta carta de magia. Se a magia for de alvo, deverá mirar no herói do oponente (qualquer carta magia que a mana permita). Utilize no máximo uma magia.

2. Se **houver mana disponível** para alguma carta de Lacaio, deverá baixar à mesa esta carta de Lacaio (qualquer carta lacaio que a mana permita). Baixe no máximo um lacaio.

3. Por fim, deverá atacar com todos os lacaios **vivos** na mesa e que **não foram baixados neste turno** no herói do oponente.

Note que só se deve realizar os procedimentos de utilizar magia e baixar lacaio apenas uma vez por turno, portanto este Jogador nunca deve baixar duas cartas lacaios ou utilizar duas magias no mesmo turno (note que isto é um requisito desta atividade e não uma regra do jogo). Outro padrão para este Jogador é que este prioriza Magias em detrimento de uso de Lacaios.

Se for realizado alguma Jogada ilegal no jogo (segundo as regras) o Motor irá dispirar uma *exception* com um erro correspondente. Faz parte desta atividade criar um código que somente faça jogadas válidas, em caso de dúvidas consulte as regras do Jogo no enunciado do trabalho 1.

3 Exemplos

Nesta seção se encontram exemplos de diversas jogadas os quais você poderá se basear.

Exemplo de Jogada para baixar um lacaio à mesa:

```
// card = objeto Carta do lacaio que quero baixar
// null = nao ha alvo nesta jogada
Jogada lac = new Jogada(TipoJogada.LACAIO, card, null);
minhasJogadas.add(lac);
```

jogadaLacaio.java

Exemplo de Jogada para utilizar magia de alvo, mirando no herói do oponente:

```
// card = objeto Carta da magia (de alvo) que quero usar
// null = para mirar no heroi do oponente
Jogada mag = new Jogada(TipoJogada.MAGIA, card, null);
minhasJogadas.add(mag);
```

jogadaMagia1.java

Exemplo de Jogada para utilizar magia de área:

```
// card = objeto Carta da magia (de area) que quero usar
// null = nao ha alvo especifico nesta jogada
Jogada mag = new Jogada(TipoJogada.MAGIA, card, null);
minhasJogadas.add(mag);
```

jogadaMagia2.java

Exemplo de Jogada para atacar com um lacaio no herói do oponente:

```java
// card = objeto Carta de meu lacaio de ataque
// null significa que o lacaio esta atacando o heroi do oponente.
Jogada atk = new Jogada(TipoJogada.ATAQUE, card, null);
minhasJogadas.add(atk);
```

jogadaAtacar2.java

Exemplo de como percorrer as cartas da mão em ordem, e baixar à mesa o primeiro Lacaio que a Mana permitir:

```java
int minhaMana = primeiroJogador ? mesa.getManaJog1() : mesa.getManaJog2();
for(int i = 0; i < mao.size(); i++){
    Carta card = mao.get(i);
    if(card.getTipo() == TipoCarta.LACAIO && card.getMana() <= minhaMana){
        Jogada lac = new Jogada(TipoJogada.LACAIO, card, null);
        minhasJogadas.add(lac);
        minhaMana -= card.getMana();
        System.out.println("Jogada: Baixei um lacaio: "+ card);
        mao.remove(i);
        i--;
        break;
    }
}
```

droparUmLacaio.java

Exemplo de utilização da Mesa para verificar quais lacaios estão vivos e prontos para atacar:

```java
ArrayList<Carta> lacaiosMeus = primeiroJogador ? mesa.getLacaiosJog1() : mesa.
    getLacaiosJog2();
for(int i = 0; i < lacaiosMeus.size(); i++){
    Carta meuLacaioVivo = lacaiosMeus.get(i);
    System.out.println("Um lacaio vivo descido em turnos anteriores: "+ meuLacaioVivo);
}
```

exemploMesa2.java

4 Observações

Atente-se às seguintes observações antes de submeter sua atividade:

- Não submeta nenhum arquivo a não ser JogadorRAxxxxxx.java (onde xxxxxx é o RA do aluno).
- Seu programa deve compilar sem erros ou *warnings*.
- Sua classe Jogador deve operar conforme as especificações. Classes que não seguirem as especificações receberão **no máximo** nota parcial na atividade.

5 Submissão

Para submeter desta atividade utilize a página da disciplina no Ensino Aberto. Utilize o recurso de portfólio para submeter a atividade. Para isso, crie uma subpasta dentro de seu portfólio com o nome **Lab3** e dentro desta subpasta submeta o arquivo fonte com a sua implementação. Ao submeter, marque a opção "Compartilhado com Formadores" na opção de compartilhamento.