MC322/MC336
Segundo semestre de 2015

Laboratório 6

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

---

1 Objetivo

O objetivo deste laboratório será a familiarização com a classe Motor para o jogo LaMa (Lacaios & Magias).

2 Atividade

Nesta atividade o aluno deverá ler o Enunciado do Trabalho 2 e após isso começar a implementar sua classe MotorRA (arquivo MotorXXXXXXX.java).

Procedimentos para montar o projeto:

1. Crie um novo projeto no Eclipse.
2. Baixe o pacote **Lab6.zip** disponibilizado no Ensino Aberto.
3. Crie uma nova classe neste projeto chamada "MotorRAXXXXXXX"(substituindo “XXXXXX” pelo seu RA), no pacote default.
4. Copie o conteúdo do arquivo disponibilizado src/MotoRArXXXXXXX.java para ser o conteúdo da classe "MotorRAXXXXXXX"recém criada.
5. Altere neste arquivo todas as ocorrências de “XXXXXX” pelo seu RA (assim como o nome do arquivo e da classe).
6. Realize os mesmos procedimentos para a classe Main no arquivo disponibilizado src/Main.java.
7. Neste momento irão aparecer diversos erros e warnings no Eclipse.
8. 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.
9. Dê OK e feche as janelas. Neste momento já deve ser possível compilar o projeto sem erros.

Na classe **MotorRA** realize as seguintes tarefas:

1. Leia o enunciado do trabalho 2, especialmente o capítulo 3 que explica sobre a classe Motor abstrata. Perceba que a classe MotorRA herda da classe Motor abstrata e que estamos implementando todos os métodos abstratos dela e chamando seu construtor através do método *super()*.
2. Analise o código presente no método *executarPartida()*, tente descobrir como funciona para a classe MotorRA receber as Jogadas das classes Jogadores e processá-las.

3. Note que este MotorRA realiza corretamente a compra de cartas para os jogadores em cada turno, e também realiza a contagem de "dano após falta de cartas" (Regra 13 do Jogo, capítulo 2 do enunciado).

4. Note que este MotorRA não realiza nenhum tratamento de Jogadas, exceto a Jogada de Baixar Lacaio. Observe o exemplo em código.

5. A tarefa deste laboratório é realizar o tratamento da Jogada de Ataque de um Lacaio. Lembre-se que um lacaio pode atacar um outro lacaio ou o herói do oponente. Realize todas as mudanças que forem necessárias de acordo com a jogada (lacaio perdeu vida, lacaio morreu, herói perdeu vida, etc). Imprima mensagens correspondentes que expliquem o que está acontecendo no Jogo.

6. Após realizar o tratamento da Jogada de Ataque, comece agora a tratar as possíveis jogadas inválidas de uma jogada de Ataque. Para isso confira a Tabela 6 e o Capítulo 7 do enunciado. Para cada tipo de erro é necessário a criação de um objeto **GameStatus** e retornar este objeto do método *processarTurno()*.

Imprima mensagens de acordo com o que for acontecendo no Jogo. Se tiver dúvidas, pode utilizar como exemplo as mensagens de log do Campeonato do Trabalho 1, disponíveis no site do docente (`http://www.ic.unicamp.br/~fusberti`).

Como exemplo temos no código o tratamento da jogada de baixar lacaio, e também o tratamento do erro quando o Jogador tenta baixar uma carta lacaio que não possui em sua mão. Utilize-se deste exemplo para construir o tratamento das demais jogadas. Note que você precisará manipular algumas estruturas de dados, como as dos objetos *lacaios, lacaiosOponente*, etc. Se precisar lembrar dos atributos existentes na classe Motor, verifique a Tabela 1 do enunciado.

A atividade planejada para este laboratório é o tratamento **completo** (incluindo todos os erros) da Jogada de Ataque de Lacaio (erros: 2, 5, 6, 7, 8). Após terminar a atividade, ou acabar o tempo, preencha o formulário em comentário na parte de cima para facilitar a correção. Após isto, submeta a atividade.

3 Extra

Esta seção não precisa ser respondida formalmente e nem acarreta na nota do laboratório. Como atividade extra ficam alguns questionamentos avançados sobre o código:

- A classe UnoptimizedDeepCopy realiza "cópia profunda" dos objetos. Por que ela é utilizada ? Imagine uma situação onde um Jogador é malicioso e altera o atributo ataque de seus lacaios, por exemplo, dobrando os ataques deles. Se não houvesse "cópia profunda" isso poderia se tornar um problema para o Motor ?

- Note que utilizamos no código alguns métodos como *contains(), indexOf()*, como esses métodos funcionam para "comparar" duas cartas ? Pesquise na Internet e encontrará que pode ser feito através do método *equals()*. Este método foi implementado na classe Carta e atualmente considera duas cartas iguais se possuírem o mesmo ID.

- Repare no "truque" que foi utilizado para que o código implementado pudesse ser o mesmo tratando-se do turno do Jogador 1 ou do Jogador 2. No laboratório passado usávamos um *if()* para cada caso, o que aumenta o tamanho de código e atrapalha a leitura. É recomendável que procure-se utilizar truques assim durante a implementação do trabalho.

4 Observações

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

- Não submeta nenhum arquivo a não ser MotorRAXXXXXXX.java (onde XXXXXX é o RA do aluno).
- Seu programa deve compilar sem erros.
- **Preencha o formulário nos comentários na classe MotorRAXXXXXX**. Seja honesto, segundo sua opinião, quanto à completude da atividade. Esta informação serve apenas para ajudar na correção e não implica na sua nota. Não preencher o formulário acarretará em nota zero.

5 Submissão

Para submeter esta 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 **Lab6** e dentro desta subpasta submeta o arquivo fonte com a sua implementação. Ao submeter, marque a opção "Compartilhado com Formadores" na opção de compartilhamento.