Trabalho de INF1636

20/04/2022

Prof. Ivan Mathias Filho

Instruções para a 1ª iteração

O software para jogar Banco Imobiliário **tem de ser** organizado segundo a arquitetura MVC. Na 1ª iteração será iniciada a especificação e a codificação do componente Model dessa arquitetura. Esse componente será formado por uma ou mais classes que fornecerão uma API (Application Programming Interface) para que os demais componentes (View e Controller) solicitem, quando necessário, serviços ao componente Model.

Implementação do componente Model

O componente Model **tem de ser** um pacote Java. Todas as classes que representem elementos básicos do jogo, tais como o tabuleiro, os jogadores e as cartas, **têm de ser** declaradas como **não públicas**, para que não possam ser diretamente referenciadas pelas classes dos pacotes View e Controller.

Segundo o princípio da ocultação da informação, apenas as funções de um módulo que serão diretamente chamadas pelos demais módulos de uma aplicação – tendo em vista obter os serviços necessários à execução de suas tarefas – deverão ser públicas. Isto é, o pacote Model terá de oferecer uma ou mais classes públicas que disponibilizarão uma API para que as regras do jogo possam ser acessadas pelas classes externas a Model. **Essa API terá de ser implementada segundo os padrões de design Singleton e Façade, que serão abordados em aulas futuras**.

O componente Controller não começará a ser desenvolvido nesta 1ª iteração, mas é interessante, neste momento, apresentar algumas de suas características futuras.

Todo os procedimentos que concernem à realização de uma jogada serão controlados por classes do componente Controller. Isso pode ser exemplificado com um trecho do manual do **War** que lista as etapas que compõem uma jogada. Esse trecho diz o seguinte:

Cada jogador passa na sua vez, tanto na primeira como em todas as outras rodadas, pelas seguintes etapas, nesta ordem:

a) Receber novos exércitos e distribuí-los de acordo com a sua estratégia;

b) Se desejar, atacar os seus adversários;

c) Deslocar seus exércitos, se houver conveniência;

d) Receber uma carta, se fizer jus a isto.

Isto é, será o componente Controller quem controlará a execução das etapas referentes a uma jogada. Entretanto, para que as regras do jogo sejam cumpridas, o Controller terá de solicitar serviços ao componente Model. Vejamos o seguinte exemplo:

O jogador de cor preta deseja atacar o território Califórnia a partir do território Colômbia/Venezuela. Para dar prosseguimento ao ataque, o Controller terá de se certificar de que

1. O território Colômbia/Venezuela pertence ao jogador de cor preta;
2. O Território Califórnia não pertence ao jogador de cor preta;
3. O território Colômbia/Venezuela faz fronteira com o território Califórnia;
4. O território Colômbia/Venezuela tem pelo menos dois exércitos.

O exemplo acima fornece um caminho a ser seguido na implementação dos componentes Model e Controller. Ele deve ser visto como umas das possibilidades de organização dos componentes, e não como um modelo que tem de ser seguido.

Objetivo da 1ª iteração

O objetivo da 1ª iteração é criar um conjunto de classes, pertencentes ao componente Model, que implemente **TODAS** as regras que constam do manual de regras do Banco Imobiliário.

Quando os componentes View e Controller começarem a ser implementados surgirão, certamente, demandas por parte desses componentes que não foram inicialmente previstas. Sendo assim, o componente Model deverá permitir futuras alterações para atender a essas demandas.

Para tal, deve-se observar o princípio aberto/fechado, cuja formulação é atribuída a Bertrand Meyer, autor do livro **Object Oriented Software Construction**. Esse princípio diz o seguinte:

- Um módulo será dito aberto se ele ainda estiver disponível para extensão. Por exemplo, se for possível adicionar campos às estruturas de dados desse módulo, ou novos elementos ao conjunto de funções que executa.
- Um módulo será dito ser fechado se estiver disponível para uso por outros módulos. Isso pressupõe que o módulo tenha sido bem-definido. Isto é, que tenha uma descrição estável e que sua API abstraia suas características específicas.

O Processo de Desenvolvimento

Em termos de processo de desenvolvimento de software, é importante que o relatório da iteração diga o que foi implementado e testado; o que foi implementado, mas não foi totalmente testado; e o que não foi implementado no curso de uma iteração. Ele deve, também, relatar os problemas encontrados e os motivos pelos quais uma ou outra funcionalidade não foi implementada ou testada. Por último, esse relatório deve apontar as funcionalidades que serão implementadas na próxima iteração e os responsáveis por cada uma delas.

Um modelo de relatório de iteração está disponível na página da disciplina no EAD.

Testes Unitários

Testes unitários terão de ser realizados para testar unidades individuais de código fonte. Tais unidades podem ser funções, métodos, classes, módulos e etc. Eles têm por objetivo verificar se cada unidade atende corretamente à sua especificação.

Todas as classes contidas no Model têm de ser testadas individualmente. Para tal, use o framework **JUnit**, que foi apresentado nas transparências do **Capítulo 18**.

Os casos de teste terão de ser elaborados e documentados de acordo com as diretrizes expostas ao longo do Capítulo 18 e resumidas em suas duas últimas transparências.