UFMG/ICEx/DCC - Programação modular
Trabalho prático 1 - *Banco Imobiliário Modular*

Pedro Paulo Valadares Brum
Rafael Grandsire de Oliveira

18 de abril de 2017

1 Introdução

Neste trabalho prático tivemos que implementas um jogo(Banco Imobiliário) sequindo os princípios da Orientação a Objetos e utilizando a linguagem Java. Primeiramente, o programa implementado deveria contruir um tabuleiro do Banco Imobiliário com base no arquivos de entrada tabuleiro.txt. Cada posição do tabuleiro possui um imóvel associado a ela. Cada imóvel possui um valor de compra e uma taxa de aluguel. Conforme o jogo vai progredindo esses imóveis podem ser comprados ou alugados pelos jogadores.

As jogadas dos participantes do jogo Banco Imobiliário estão no arquivo de entrada jogadas.txt. Esse arquivo possui as instruções do jogo e cada linha representa um ação de um jogador. Com esse arquivo podemos simular o jogo e gerar as estatísticas do jogo.

Além disso, deríamos implementar as regras de negócios básicas do jogo e tratar casos de exceção simples.

Como o jogo deveria ser implementado seguindo-se os princípios da Orientação a Objeto, espera-se que o programa tenha uma boa abstração do problema, que utilize de forma inteligente o encapsulamento, que use corretamente o princípio da herança e que aplique o polimorfismo de forma a melhorar o código. Espera-se também um programa bem modularizado, com a separação em conjuntos de módulos, classes com independência de funcionamento, separação de resposabilidades, entre outras características. Além disso, espera-se também o uso do princípio das mensagens que propõe a comunicação entre objetos, envio e recebimento de mensagens e utilização de contrado firmado entre as partes.

2 Implementação

2.1 Banco Imobiliário

Este módulo é o módulo principal e representa o jogo Banco Imobiliário. Nele instanciamos objetos das classes *Tabuleiro* e *Jogada* utilizando os arquivos de entrada tabuleiro.txt e jogadas.txt, respectivamente. Com o objeto da classe *Jogada* instanciado chamamos o método *simular* passando como parâmetro o tabuleiro construído e o arquivo onde serão escritas as estatísticas do jogo.

2.2 Intrução

Neste módulo foi implementado a classe *Instrucao* que representa as instruções dos jogadores. Ela foi declarada como public, pois ela pode precisar ser utilizada em qualquer lugar do programa. Essa classe possui os atributos *id*, *idJogador* e *valorDado*.

Atributos:

id - Identificador da linha no arquivo(1 até n) - *private*.

idJogador - Identificador do jogador - *private*.

valorDado - Valor do dado de seis faces, que o jogador tirou na jogada - *private*.

Os atributos da classe são declarados como *private*, e dessa forma, só podem ser acessados por métodos da mesma classe.

Métodos:

$formatInstrucao$ - *public static* : Caso a linha lida do arquivo jogadas.txt tenha apenas "DUMP"retorna um par (null, null). Caso contrário, retorna um par (id, Instrução). Sendo que instrução possui um id, o id do jogador associado à instrução e o valor retirado no dado.

Instrucao - Construtor da classe Instrucao - *public*.

getId - permite o acesso ao atributo id - *public*.

getIdJogador - permite o acesso ao atributo idJogador - *public*.

getValoDado - permite o acesso ao atributo valorDado - *public*.

isDump - seta objeto instrucao como null - *public static*.

Os métodos foram declarados como *public* e , portanto, podem ser acessados de qualquer outra parte do código. Os métodos $formatInstrucao$ e *isDump* foram declarados

também como *static* e, portanto, são acessados diretamente pela classe, são resolvidos em tempo de compilação, não podem ser sobrescritos e só podem acessar atributos e métodos *static*.

2.3 Jogada

Neste módulo foi implementado a classe *Jogada* que representa o jogo. Ela foi declarada como public, pois ela pode precisar ser utilizada em qualquer lugar do programa. Essa classe possui os atributos *qtIntrucoes*, *instrucoes*, *qtJogador*, *jogadores* e *jogadoresPositivos*.

Atributos:

qtInstrucoes - quantidade de instruções presentes no arquivo de entrada - *private final*.

intrucoes - vetor de instruções - *private final*.

qtJogadores - quantidade de jogaores no jogo - *private final*.

jogadores - vetor de jogadores - *private final*.

jogadoresPositivos - jogadores ativos no jogo(que jogam a rodada) - *private*.

Os atributos da classe são declarados como *private*, e dessa forma, só podem ser utilizadas detro da própria classe. Os atributos *qtInstrucoes*, *instruces*, *qtJogadores* e *Jogadores* foram declaradas como *private final* e, portanto, são utilizadas como constanstes, ou seja, para cada objeto instanciado, possuem sempre o mesmo valor.

Métodos:

simular - *public*: recebe o tabuleiro do Jogo, o nome do arquivo de saída, simula o jogo com base nas instruções presente no arquivo de entrada jogadas.txt e escreve as estatísticas no arquivo de saída.

Jogada - Construtor da classe - *public*.

getQtInstrucoes - permite o acesso ao atributo qtInstrucoes - *public*.

getJogadores - permite o acesso ao atributo jogadores - *public*.

debug - imprime o jogadores - *public*.

decrementarJogadores - decrementa o atributo jogadoresPositivos - *public*.

Os métodos foram declarados como *public*, pois podem ser utilizados em qualquer local do programa.

2.4 Jogador

Neste módulo foi implementado a classe *Jogador* que representa os jogadores. Ela foi declarada como public, pois ela pode precisar ser utilizada em qualquer lugar do programa. Essa classe possui os atributos *id*, *posicao*, *valor*, *qtJogadas*, *qtVoltas*, *qtPassaVez*, *aluguelRecebido*, *alguelPago*, *gastoImoveis*.

Atributos:

id - id do jogador - *private final*.

posicao - posição do jogador no tabuleiro - *private*.

valor - valor em dinheiro que o jogador possui - *private*.

qtJogadas - quantidade de jogadas registradas - *private*.

qtVoltas - quantidade de voltas dadas no jogo pelo jogador - *private*.

qtPassaVez - quantidade de vezes que o jogador passou a vez - *private*.

aluguelRecebido - valor de aluguel recebido - *private*.

aluguelPago - valor de alugel pago - *private*.

gastoImoveis - valor gasto com imóveis - *private*.

Os atributos da classe são declarados como *private*, e dessa forma, só podem ser utilizadas detro da própria classe. O atributo *id* foi declarada como *private final* e, portanto, é utilizado como constanstes, ou seja, para cada objeto instanciado, possui sempre o mesmo valor.

Métodos:

andar - *public*: faz o jogador passado como parâmetro andar no tabuleiro. Caso o jogador pare em uma posição e o imóvel dessa posição esteja disponível e ele tiver dinheiro para comprar, o imóvel é adquirido pelo jogador. Caso o imóvel não esteja disponível e ele tenha dinheiro para pagar aluguel, o imóvel é alugado pelo jogador. Se o jogador não tiver dinheiro para aluguel no caso em que o imóvel não está disponível, ele é eliminado.

Jogador - Construtor da classe - *public*.

descontarValor - decrementa o valor que o jogador possui - *public*.

creditarValor - incrementa o valor que o jogador possui - *public*.

setPosicaoInicial - seta a posição que o jogador está - *public*.

eliminar - elimina o jogador - *public*

isEliminado - seta posição do jogador como 0 - *public*.

getId - permite o acesso ao atributo id - *public*.

receberAluguel - incrementa o atributo aluguelRecebido e credita a quantia no que o jogador possui - *public*.

pagarAluguel - incrementa o atributo aluguelPago e desconta a quantia no que o jogador possui - *public*.

gastarImovel - incrementa o atributo gastoImoveis e desconta a quantia no que o jogador possui - *public*.

getValor - permite o acesso ao atributo valor - *public*.

getQtJogadas - permite o acesso ao atributo qtJogadas - *public*.

getQtVoltas - permite o acesso ao atributo qtVoltas - *public*.

getQtPassaVez - permite o acesso ao atributo qtPassaVez - *public*.

getAluguelRecebido - permite o acesso ao atributo aluguelRecebido - *public*.

getAluguelPago - permite o acesso ao atributo aluguelPago - *public*.

getGastoImoveis - permite o acesso ao atributo gastoImoveis - *public*.

toString - retorna o id, a posição e o valor do jogador - *public*.

Os métodos foram declarados como *public*, pois podem ser utilizados em qualquer local do programa.

2.5 Tabuleiro

Neste módulo foi implementado a classe *Tabuleiro* que representa o tabuleiro do jogo Banco Imobiliário. Ela foi declarada como public, pois ela pode precisar ser utilizada em qualquer lugar do programa. Essa classe possui os atributos *qtPosicoes*, *posicaoInicial* e *posicao*.

Atributos:

qtPosicoes - quantidade de posições no tabuleiro - *private final*.

posicaoInicial - posição inicial do tabuleito - *private final*.

posicao - vetor de posições - *private final*.

Os atributos da classe são declarados como *private final*, e dessa forma, só podem

ser utilizadas detro da própria classe. Além disso, elas são utilizadas como constanstes, ou seja, para cada objeto instanciado, elas possuem sempre o mesmo valor.

Métodos:

Tabuleiro - Construtor da classe - *public*.

getQtPosicoes - permite o acesso ao atributo qtPosicoes - *public*.

getPosicaoInicial - permite o acesso ao atributo posicaoInicial - *public*.

getPosicao - permite o acesso ao atributo posicao - *public*.

Os métodos foram declarados como *public*, pois podem ser utilizados em qualquer local do programa.

2.6 Posição

Neste módulo foi implementado a classe *Posicao* que representa as posições do tabuleiro. Ela foi declarada como public, pois ela pode precisar ser utilizada em qualquer lugar do programa. Essa classe possui o somente um atributo: *id*.

Atributos:

id - id da posição - *private final*.

O atributo *id* foi declaraco como *private final*, e dessa forma, só pode ser utilizado detro da própria classe. Além disso, ele é utilizado como constanste, ou seja, para cada objeto instanciado, ele possui sempre o mesmo valor.

Métodos:

formatPosicao - *public static*: utilizada pela classe Tabuleiro para formatar a posição recebida do arquivo de entrada tabuleiro.txt. Caso o identificador da posição seja igual a 1 ela representa a posição *Start*. Caso seja igual a 2 representa uma passe a vez. Caso seja igual a 3 representa um imóvel. Se a posição tiver um imóvel é feita a coleta das informações do imóvel: tipo, valor e taxa.

Posicao - Construtor da classe - *public*

isInicial - verifica se a posição passada por parâmetro é inicial - *public static*.

Os métodos foram declarados como *public*, pois podem ser utilizados em qualquer local do programa. Os métodos *formatPosicao* e *isInicial* foram declarados também como *static* e, portanto, são acessados diretamente pela classe, são resolvidos em tempo de compilação, não podem ser sobrescritos e só podem acessar atributos e métodos *static*.

2.7 Imóvel

Neste módulo foi implementado a classe *Imovel* que representa os imóveis do jogo. Ela foi declarada como public, pois ela pode precisar ser utilizada em qualquer lugar do programa. Essa classe possui os atributos *tipo*, *valor*, *taxa* e *dono*. Essa classe é uma subclasse da classe *Posicao*. Logo, a classe *Imovel* herda os membros que podem ser acessíveis foram da classe *Posicao* e acrescenta novos membros. Dessa forma, utiliza-se o princípio da herança. O uso da herança permite compartilhar similaridades, preservar as diferenças e proporciona maior legibilidade do código existente.

Atributos:

tipo - tipo do imóvel - *private final*.

valor - valor do imóvel - *private final*.

taxa - taxa do imóvel - *private final*.

dono - id do dono do imóvel - *private*.

Os atributos da classe são declarados como *private*, e dessa forma, só podem ser utilizadas detro da própria classe. Os atributos *tipo*, *valor* e *taxa* foram declaradas como *private final* e, portanto, são utilizados como constanstes, ou seja, para cada objeto instanciado, eles possuem sempre o mesmo valor.

Métodos:

Imovel - Construtor da classe - *public*.

getTipo - permite o acesso ao atributo tipo - *public*.

isComprada - retorna se o imóvel possui dono - *public*.

comprar - compra o imóvel - *public*.

devolver - devolve o imóvel - *public*.

getDono - permite o acesso ao atributo dono - *public*.

getValor - permite o acesso ao atributo valor - *public*.

getTaxa - permite o acesso ao atributo taxa - *public*.

Os métodos foram declarados como *public*, pois podem ser utilizados em qualquer local do programa.

2.8 Decisões de Manutenabilidade

Muitas informações como o ID dos objetos e até mesmo o tipo do imóvel não pareciam ter utilidades específicas para a resolução do problema proposto. Porém, é importante que o trabalho seja feito prevendo eventuais mudanças necessárias e também facilitando a manutenção futura. Deste modo, optou-se por armazenar esses dados de maneira precavida.

3 Testes

Exemplo de caso teste:
tabuleiro.txt: indica como será o tabuleiro do jogo.

```
6
1;1;2
2;2;1
3;3;3;4;100;10
4;4;3;1;200;20
5;5;3;2;300;90
6;6;2
```

A primeira linha do arquivo represento o número de posições que o tabuleiro deverá ter. As outras linha descrevem cada uma das posições. Nesse casa o tabuleiro do jogo possui 6 posições e a posição inicial é a de número 2.

- Posição 1 - Passe a vez.
- Posição 2 - *Start*.
- Posição 3 - hotel de valor 100 e com aluguel igual a 10.
- Posição 4 - residência de valor 200 e com aluguel igual a 40.
- Posição 5 - comércio de valor 300 e com aluguel igual a 270.
- Posição 6 - Passe a vez.

jogadas.txt: informa as jogadas de cada participante do jogo.

```
9
1;1;1
2;2;2
3;3;3
4;1;2
5;2;1
6;3;1
7;1;1
8;2;4
9;3;1
DUMP
```

A primeira linha do arquivo possui o número de instruções de jogada, o número de participantes do jogo e o valor em dinheiro que cada jogadors possui no início do jogo. Nesse caso, o jogo possui 9 instruções, 3 participantes e cada jogador começa com 300 reais.

• A primeira linha descreve que o jogador 1 tirou 1 no dado. Logo, o jogador 1 vai para posição 3 e compra o hotel. Assim, o jogador 1 fica com um saldo igual a 200.

• A segunda linha descreve que o jogador 2 tirou 2 no dado. Logo, o jogador 2 vai para posição 4 e compra a residência. Assim, o jogador 2 fica com um saldo igual a 100.

• A terceira linha descreve que o jogador 3 tirou 3 no dado. Logo, o jogador 3 vai para posição 5 e compra o comércio. Assim, o jogador 3 fica com um saldo igual a 0.

• A quarta linha descreve que o jogador 1 tirou 2 no dado. Logo, o jogador 1 vai para posição 5. Como o imóvel da posição já possui dono e o valor do aluguel é maior do que 200, o jogador 1 sai o jogo.

• A quinta linha descreve que o jogador 2 tirou 1 no dado. Logo, o jogador 2 vai para posição 5. Como o imóvel da posição já possui dono e o valor do aluguel é maior do que 100, o jogador 2 sai o jogo. Como o jogador 1 já tinha saído do jogo, o jogador 3 ganha a partida.

Simulação do jogo utilizando os arquivos tabuleiro.txt e jogadas.txt. Os jogadores são representados pelo indicativos J1, J2 e J3 representando o Jogador 1, Jogador 2 e Jogador 3, respectivamente.

Posição tabuleiro	1	2	3	4	5	6
tipo	2	1	3(4)	3(1)	3(2)	2
Rodada 1			J1	J2	J3	
Rodada 2					J1-J2-J3	

estatisticas.txt: dados estatísticos computados para o jogo simulado com base nos arquivos tabuleiro.txt e jogadas.txt.

```
1:2
2:1-0;2-0;3-0
3:1-200.00;2-100.00;3-0.00
4:1-0.00;2-0.00;3-0.00
5:1-0.00;2-0.00;3-0.00
6:1-100;2-200;3-300
7:1-0;2-0;3-0
```

A primeira linha do arquivo corresponde ao número de rodadas que o jogo teve.

4 Conclusão

Tendo em vista o problema apresentado, pode-se dizer que a implementação deste trabalho foi desafiadora, porém bastante interessante. Através deste trabalho pudemos ter uma maior noção de como utilizar os princípios da Orientação a Objetos. Os resultados dos testes estão de acordo com o que era esperado, e sendo assim, pode-se dizer que o trabalho foi bem-sucedido.