Programação Modular

TP1 - Banco Imobiliário

Cristiano Guimarães
Vitor Menezello

Belo Horizonte - Abril de 2017

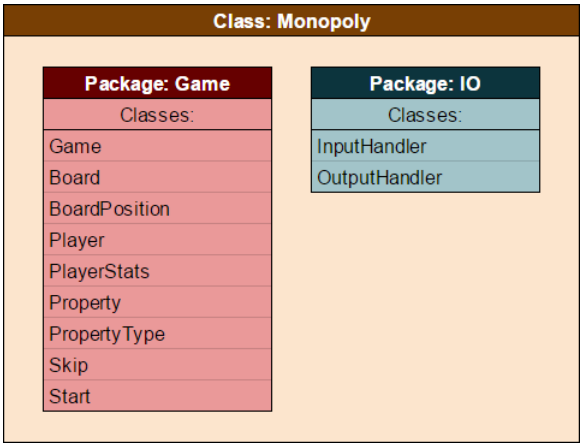
1 Introducão

Neste trabalho implementamos uma versão simples do jogo Banco Imobiliário para praticar os conceitos de Orientação a Objetos, como abstração, encapsulamento, herança e polimorfismo. O programa foi escrito em Java devido à sua simplicidade.

O programa lê de um arquivo texto as posições do tabuleiro e passa as informações lidas para a classe *Board* (Tabuleiro). Em seguida, um segundo arquivo texto, que contém o número de jogadores e cada jogada realizada, é lido e seus dados são escritos na classe *Player* (Jogador), e, com uma sequência de métodos, o jogo é criado e executado, gerando um relatório com as estatísticas pedidas pela especificação do problema.

2 Implementação

2.1 Classes

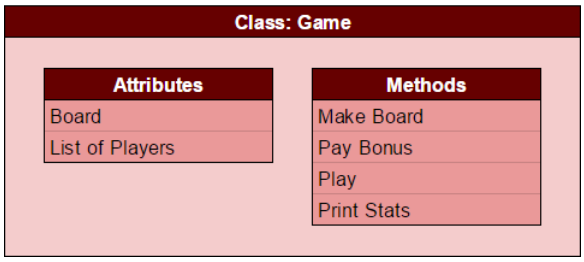


No pacote *Game* existem várias classes que formam a abstração de como o jogo é realmente composto. Há também um segundo pacote (*IO*) que lida com os arquivos de entrada e gera um relatório das jogadas.

2.1.1 Game

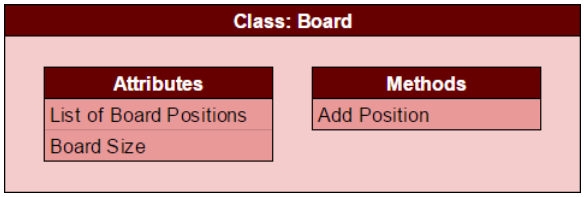
A classe *Game* possui um *Board* e uma lista de *Players*. Durante a execução do programa, o método de criação do tabuleiro é chamado e recebe como parâmetro o nome do primeiro arquivo de entrada, definido comumente como "tabuleiro.txt". Esse método cria uma instância de *InputHandler* para ler este arquivo.

Os dois próximos métodos são relacionados às jogadas lidas no segundo arquivo, "jogadas.txt". Nestes métodos, são criados os jogadores e todas as suas características, que são atualizadas a cada jogada, levando a um vencedor. Finalmente, no último método, é escrito no arquivo de saída, "estatisticas.txt", o resultado do jogo. Este último método utiliza uma instância de *OutputHandler*.



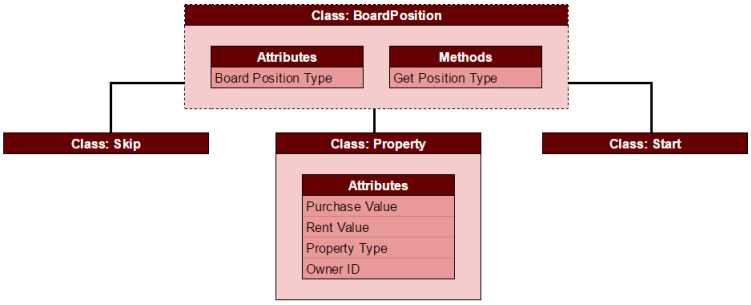
2.1.2 Board

Durante a criação do tabuleiro, o método que adiciona uma posição de tabuleiro é chamado até que ele esteja completo. A instância de *Board* passa a ter então um atributo que indica o tamanho do tabuleiro e uma lista de posições.



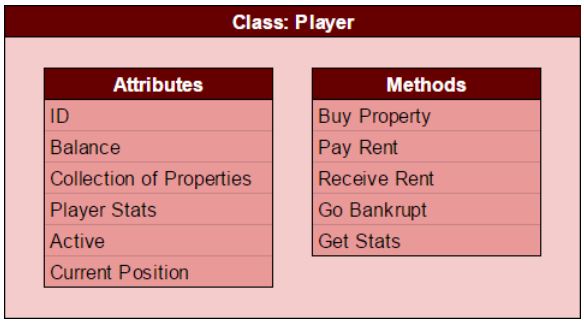
A classe abstrata *Board Position* possui um atributo simples que diz qual o tipo de posição que ela é. Existem três possibilidades: *Start*, *Skip* ou *Property*. Cada uma delas é uma classe que herda as características da classe *Board Position*.

Os imóveis são as classes que possuem mais atributos, armazenando valores como o tipo de propriedade, valor de compra e aluguel e dono.



2.1.3 Player

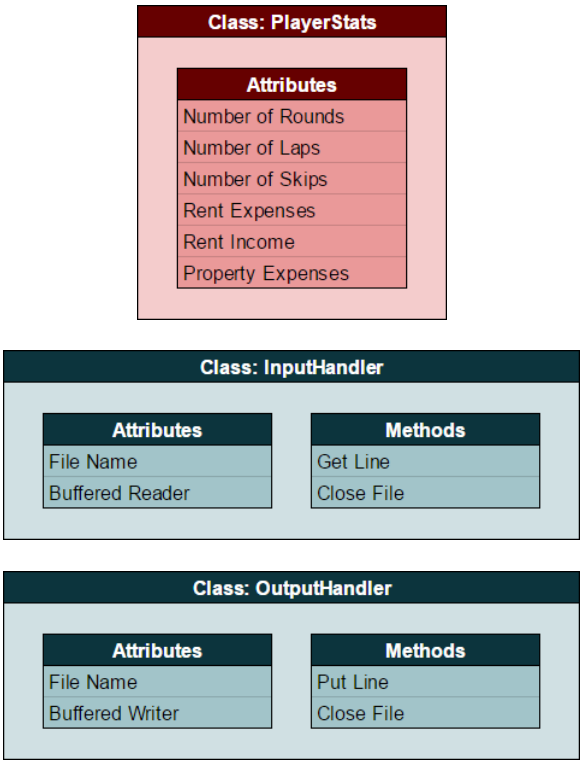
A outra classe importante do jogo é a classe jogador. Cada jogador possui um ID que o representa, seu saldo, uma coleção de propriedades, descritas acima, uma flag que indica que ele ainda está jogando e sua posição no tabuleiro. O atributo *Player Stats* é uma classe separada, e é utilizada principalmente para gerar as estatísticas finais. Os métodos principais da classe *Player* são as de comprar propriedade, pagar ou receber aluguel e ir à falência.



Em *Player Stats* armazenamos o número de rodadas, voltas no tabuleiro e turnos pulados durante o jogo, as despesas e receitas com aluguel e os gastos com compras de propriedades.

2.1.4 Input e Output Handlers

As classes *InputHandler* e *OutputHandler* possuem função similar, por isso seus atributos e métodos são muito parecidos. A principal diferença é que a classe que lida com a entrada possui um leitor de arquivo e um método que retorna uma linha lida, e a que lida com a saída possui um escritor em arquivo que escreve uma linha neste.



2.2 Fluxo do Programa

A classe principal do programa, Monopoly, possui um fluxo simples. Primeiro é criada uma instância da classe *Game* e em seguida são chamados os seus respectivos métodos, listados na descrição da classe. Após a criação do tabuleiro, o jogo ocorre com a leitura das jogadas. Por último é gerado um arquivo com os resultados. As regras do jogo Banco Imobiliário proposto estão claras na especificação do trabalho prático.

Durante a implementação do código, foi decidido que quando um jogador não possuir o dinheiro para pagar o aluguel de uma propriedade na qual ele está, o seu saldo é deduzido para que ele fique com valor negativo, deixando claro nas estatísticas que o jogador faliu.

3 Testes

O primeiro teste foi realizado com os arquivos disponibilizados no Moodle. Para garantir a corretude e funcionalidade do programa, foi criado mais um teste, que está descrito logo abaixo, incluindo os dois arquivos de entrada "tabuleiro.txt" e "jogadas.txt" e a saída "estatisticas.txt". Os arquivos utilizados estarão disponíveis junto com o código.

tabuleiro.txt

```
8
1;1;1
2;2;3;2;500;50
3;3;2
4;4;3;1;300;80
5;5;3;3;400;50
6;6;2
7;7;3;5;300;30
8;8;3;4;200;10
```

jogadas.txt

```
10%3%500
1;1;3
2;2;4
3;3;1
4;1;2
5;2;2
6;3;2
7;1;4
8;2;3
9;3;1
DUMP
```

estatisticas.txt

```
1:3
2:1-1;2-1;3-0
3:1-450;2-350;3--240
4:1-250;2-0;3-0
5:1-0;2-250;3-0
6:1-800;2-400;3-500
7:1-1;2-0;3-0
```

4 Conclusão

O trabalho, apesar de simples, cumpre a sua função de ajudar a sintetizar os conceitos de Orientação a Objetos, fazendo com que os alunos implementem seus códigos com um pensamento diferente do que haviam fazendo até então.

Isso gera algumas dificuldades, principalmente ao começar a criar uma abstração que represente corretamente o problema utilizando objetos, e também quanto ao uso correto de modificadores de acesso para melhor encapsulamento, mas essas dificuldades foram superadas e o problema proposto foi resolvido de maneira clara, eficiente e simples.

Algumas pequenas dificuldades surgiram também por falta de familiaridade com a linguagem, mas estas foram resolvidas facilmente com pesquisas e leituras de tutoriais. As referências para os sites utilizados estão logo abaixo.

5 Bibliografia

Tutorials Point: Java - Files and I/O