Universidade de São Paulo - USP
Instituto de Ciências Matemáticas e de Computação - ICMC
Departamento de Ciências de Computação - SCC
Bacharelado em Ciências de Computação - BCC
Disciplina de Programação Orientada a Objetos - SSC103

Prof. Márcio Eduardo Delamaro

Trabalho prático - POO

Fernando Moura Leite Vendrameto (9875973)

Guilherme de Pinho Montemovo (9779461)

Gustavo Sutter Pessurno de Carvalho (9763193)

Leonardo Moreira Kobe (9778623)

Sumário

1 Introdução

Neste projeto foi implementado uma versão online, com interface gráfica, do clássico jogo de tabuleiro Monopoly. As regras foram modificadas particularmente para esta versão.
Para implementação foi utilizado o Java 8. O projeto foi compilado e testado no Sistema Operacional Linux (Ubuntu 16.04 e Manjaro 17.10).

2 Objetivos

Foi proposto o desenvolvimento de uma plataforma de interação entre dois ou mais jogadores. Além de desenvolver um jogo de Banco Imobiliário, também foi decidido implementar os seguintes tópicos:

- Interface Gráfica de Jogador
- Interface Gráfica de Servidor
- Inteligência Artificial

3 Descrição do Software

3.1 Classes principais

- Servidor: Essa classe contém toda a lógica do jogo. É ela que administra os turnos, e executa as ações do jogador, por meio do método estática *main*. Também é responsável pela conexão dos clientes com o servidor e pelo envio de informações à eles, por meio de métodos do objeto Servidor.
- Cliente: Essa classe mostra o tabuleiro na tela para os jogadores, armazena algumas informações básicas sobre o jogador e recebe informações do servidor, como mensagens para o *log*, mensagens para atualização da GUI e listas de compráveis para o jogador escolher na hipoteca ou na construção e venda de casas.
- MainGUI:

Além dessas, há também uma classe abstrata para os espaços do tabuleiros, que é estendida pelos compráveis (propriedades e companhias) e pelos jogáveis (início, vá para a prisão, sorte ou revés); uma classe para o jogador; uma para o banco, com métodos estáticos; uma classe que implementa as ações dos jogáveis; duas classes para os dados; uma que carrega os dados das propriedades, escritos em um arquivo, para o jogo e cinco classes para implementar partes da GUI como o plano de fundo, o log de mensagens, a lista de bens, a árvore de propriedades de cada jogador e o tabuleiro.

3.2 Funcionamento do Jogo

O jogo começa depois que todos os jogadores estiverem conectados e digitarem seus respectivos nomes. A ordem de jogo é a ordem da conexão dos jogadores com o servidor. O número máximo de jogadores permiditos (somando players reais e bots) é 6.

Os jogadores começam com $1500 cada. No início da rodada o jogador rola os dados e anda no tabuleiro o valor deles. Dependendo do espaço em que ele cai, ocorrem algumas ações.

Se for um espaço comprável (propriedade ou companhia) o jogador pode comprá-lo, se ainda não tiver dono. Caso contrário, o jogador deve pagar aluguel ao dono da mesma. Se for um espaço de ação (Início, Sorte ou Revés, Imposto de Renda, Vá para a Cadeia), o jogador sofre o efeito da ação e segue em seu turno.

Concluída a parte do espaço o jogador pode tomar mais algumas ações, que são: Hipotecar, Construir Casa, Vender Casa e Encerrar Rodada. As três primeiras são feitas em cima das propriedades do jogador. A última finaliza sua rodada e passa a vez para o próximo jogador. Há também a opção de sair do jogo, que é tratada como se o jogador tivesse falido.

Ao ir para a cadeia o jogador deve ficar lá por três rodadas e não pode realizar nenhuma ação, porém continua recebendo aluguel se outro jogador cair em uma de suas propriedades.

Um jogador vai à falência quando tem que pagar (aluguel ou em alguma ação) e não tem dinheiro para isso, mesmo que ele possua propriedades. Todas suas propriedades são devolvidas ao banco e qualquer casa construída nelas é demolida.

O jogo termina quando todos os jogadores vão à falência menos um, que é o ganhador da partida.

3.2.1 Como jogar

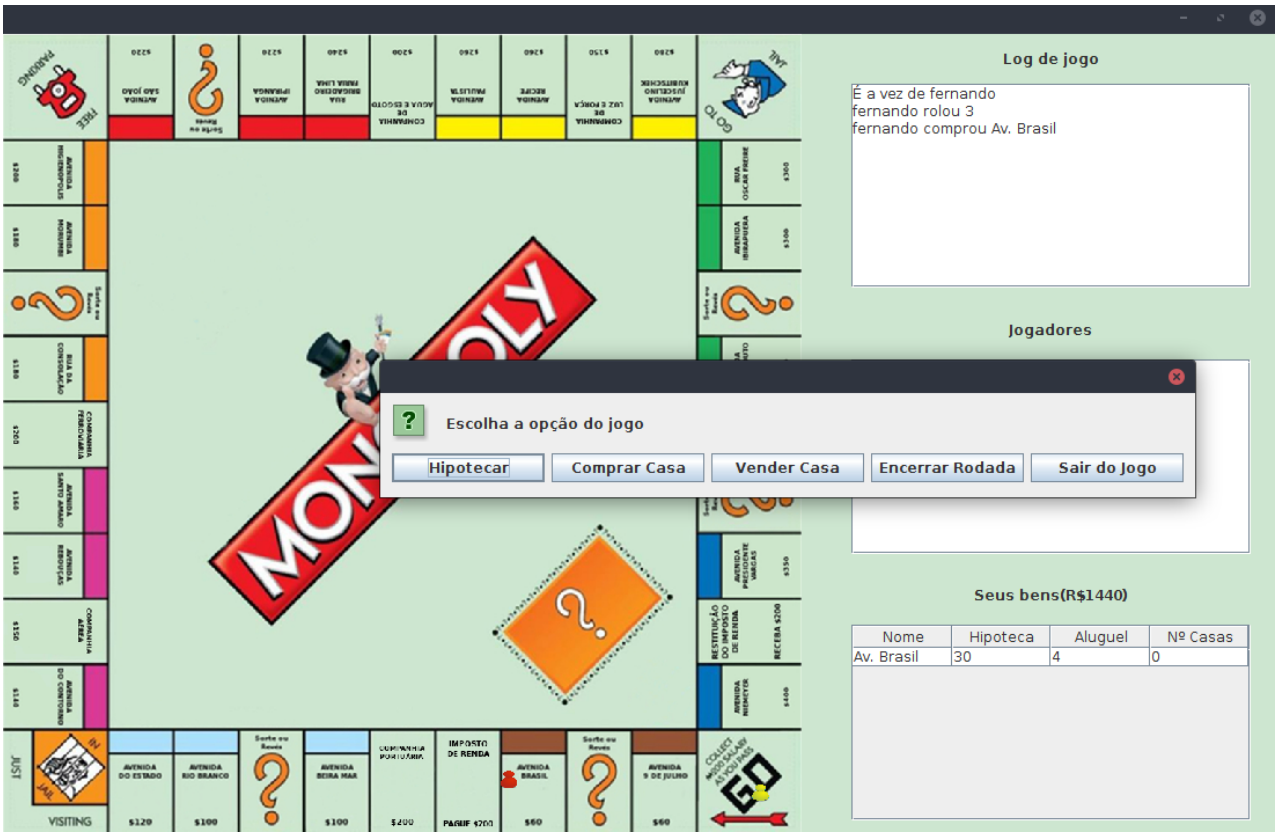
Primeiro é necessário inicializar o servidor, rodando a partir do arquivo "Servidor.java". O usuário que roda o servidor tem, então, que digitar o endereço de IP do servidor na forma X.X.X.X, a porta que será aberta para a conexão o número de jogadores que participarão da sessão e o número de *bots*. Então, cada jogador deve executar, a partir do arquivo "Cliente.java", o programa, digitar o IP e a porta aberta do servidor, seu nome e então o jogo em si começa. A partir desse ponto a execução é bem intuitiva, e os jogadores que estão esperando seu turno podem acompanhar os movimentos dos outros por meio de um *log* de ações na parte superior direita da GUI.

Observe que, por causa da implementação do servidor, o envio aos clientes é sequencial, então as telas só aparecem para o próximo jogador após o atual efetuar suas ações.

3.3 Interface Gráfica

A interface é, na maior parte do tempo, como na Figura 1. Apenas as telas de inicialização aceitam entradas digitadas pelo usuário, e essas entradas são tratadas para que não ocorram erros. Durante o resto da interação o usuário só utiliza botões e listas para escolher que ações tomar, o que facilita tanto a jogabilidade quanto o tratamento dessas ações. A lateral direita da tela é composta por um *log* com as ações de todos jogadores, uma árvore com todos jogadores e suas propriedades e uma tabela com os bens que o usuário tem.

Figura 1: Tela de opções de jogo, que aparece depois que o jogador rola os dados



3.4 Rede

A rede foi montada de forma que todas as ações do jogo sejam feitas pelo servidor, com o cliente tendo o papel apenas de escolher qual ação deve ser executada e, se necessário, escolher uma propriedade entre as suas para realizar essa ação. Como a comunicação é constante e diversa entre servidor e cliente optamos por não utilizar *threads* para ter sincronia entre os jogadores, já que nessa implementação sequencial eles ficam todos no mesmo ponto do jogo.

4 Principais Bibliotecas Utilizadas

- java.util.ArrayList: usada em várias classes, quando julgamos ser necessário usar uma lista
- java.util.HashMap: usada para mapear as listas usando uma chave, quando necessário
- java.io.ObjectInputStream e java.io.ObjectOutputStream: usadas para entrada e saída de objetos no servidor e no cliente
- java.net.ServerSocket e java.net.Socket: usadas para fazer as conexões servidor-cliente
- java.io.Serializable: usada para serializar e tornar possível o envio de objetos que nós criamos
- java.awt e javax.swing: usadas para fazer a interface gráfica

5 Contribuição dos Membros

O código foi trabalhado por todos os membros do grupo de forma distribuída, de modo que as tarefas foram divididas de acordo com sua extensão, portanto a divisão percentual de contribuição ficou em 25% para cada integrante.