INF1636 – Relatório de Acompanhamento de Iteração

Jogo – Banco Imobiliário

1. Participantes

Breno Varela de Oliveira (2010309)

Daniel Stulberg Huf (1920468)

Tulio Gomes Vuolo (1920306)

2. Iteração

1ª Iteração

Data de início: 25/04/2022

Data de Encerramento: 08/05/2022

3. Tarefas Planejadas

Nessa primeira iteração, foi implementado o corpo majoritário do componente **Model**, segundo a arquitetura MVC, para o projeto do jogo Banco Imobiliário.

4. Organização das Tarefas

O componente Model foi implementado a partir de um pacote Java, no qual suas classes representaram elementos do jogo. Ao longo da execução dessa iteração, Breno esteve encarregado principalmente de desenvolver as transações bancárias entre jogadores e o banco, com compras e vendas de títulos, bem como testar essas movimentações dentro da Main, onde é simulado uma versão primitiva do Banco Imobiliário através do terminal e sem perguntas ao usuário. Daniel foi responsável por criar os elementos básicos do jogo, como por exemplo o tabuleiro com as casas, as cartas de Sorte ou Revés, e os dados do jogo. Tulio foi encarregado de testar e executar a primeira versão primitiva do jogo, e criar casos de teste unitário (JUnits) para as principais classes que foram desenvolvidas.

5. Tarefas Executadas

A seguir, será descrito o modo como foram implementadas as regras do jogo, classe por classe.

- **Banco**

Classe que representa o banco no Jogo Imobiliário. O banco possui uma **conta** bancária estática, inicializada com um saldo de 200 000 no início do jogo, através de seu construtor. Foi criado um método auxiliar estático que retorna a conta bancária do banco.

- **CasaNegociavel**

Subclasse da classe CasaTabuleiro, a casa negociável representa as casas passíveis de compra, venda e pagamento de taxas. Cada casa negociável possui um **preço** de compra/venda pré-estabelecidos, além de um jogador **titular**. No início do jogo, o titular de todas as casas negociáveis é inicializado como null, mas pode ser alterado posteriormente através do método **transfereTitular**. Também foram criados métodos auxiliares para retornarem os valores dos atributos das instâncias dessa classe.

- **CasaTabuleiro**

Classe abstrata que representa uma casa genérica do tabuleiro do Banco Imobiliário. Cada casa possui um **nome**, que pode ser acessado através de um método auxiliar get.

- **Companhia**

Subclasse da classe CasaNegociavel, a companhia representa as casas de companhia do jogo. Uma companhia, além de ter as características herdadas de sua superclasse, possui uma **taxa**, que, multiplicada pelo numero de dados lançados pelo jogador que parou em tal companhia (já previamente possuída por um titular), é retornada através do método **getTaxa**.

- **ContaBancaria**

Classe que representa a conta de um jogador ou do banco, é através dela em que são processadas todas as operações financeiras do jogo. Uma conta bancária possui um **saldo**, que pode ser alterado através de métodos de **depósito** e **saque**. Além disso, podem ser **transferidos** valores entre uma conta bancária corrente e uma conta bancária de destino.

- **Dado**

Classe que representa um dos dados do jogo. O único método que um dado executa é ser **lançado**, retornando um valor aleatório entre 1 e 6.

- **ExcecaoDinheiroInsuficiente**

Classe que define a exceção de dinheiro insuficiente, a ser levantada caso um jogador não possua saldo suficiente em sua conta bancária para realizar alguma operação financeira ao longo do jogo.

- **Jogador**

Classe que representa um jogador do jogo. Sobre o jogador, devemos saber a sua **posição** no tabuleiro, o **número de repetições** que está jogando a mesma rodada (caso ele lance números repetidos nos dados), o **número de rodadas preso** (caso o jogador esteja preso em dado momento), a sua **conta bancária**, se ele **está na prisão** em dado momento, e se ele **possui o cartão de saída livre da prisão**.

Para a classe jogador, implementamos, além dos métodos auxiliares de get, os métodos **prende**, **solta, ganhaSaidaLivre, compra, paga, rolarDadoseMover.** Esses três últimos métodos serão descritos com mais detalhes a seguir:

1. **compra**
   Para efetuar uma compra, ocorre uma tentativa de transferência de valor entre a conta do jogador que deseja comprar a casa comercializável (recebida como parâmetro) e a conta do banco, seguida de uma transferência de titular do banco para o jogador corrente. Essa tentativa pode não ser efetuada, caso o jogador não possua saldo suficiente para tal.

2. **paga**
   Método desenvolvido para tratar o pagamento de alugueis ou taxas de uma casa comercializável. Caso um jogador pare em um terreno ou companhia que possua dono, o mesmo deve pagar o preço de aluguel / taxa do local recebido como parâmetro, segundo as condições atuais em que o terreno se encontra. Será feita uma tentativa de pagamento, que se não for bem-sucedida, deverá ser tratada caso a caso, para cada jogador.

3. **rolarDadoseMover**
   Método que trata o lançamento dos dados e a movimentação do jogador no tabuleiro. São tratadas todas as situações possíveis, isto é, se o jogador está na prisão ou não, se os dados lançados foram números iguais, além de processar quantas rodadas o jogador está preso, ou se está repetindo lançamentos em uma mesma rodada. A depender de cada caso, o jogador irá se movimentar de acordo com a situação em que se encontra.

- **Passagem**

Subclasse da classe CasaTabuleiro, a passagem representa as casas do tabuleiro em que não há operação de compra ou venda. No entanto, uma passagem possui um **valor** atribuído a ela, que poderá ser pago ou depositado ao jogador que passar pela casa.

- **SorteReves**

Subclasse da classe CasaTabuleiro, ela representa o baralho das cartas de Sorte ou Revés. No início de cada jogo, a lista de **cartasSorteReves** será instanciada e embaralhada. A longo do jogo, o método **pickSorteReves** poderá ser chamado, onde será identificada a carta que está na **posição corrente**, isto é, a carta de cima do baralho. A depender do número da carta, o jogador que a tira poderá pagar ou receber algum valor, ou mesmo receber uma carta de **saída livre da prisão**, que se tornará indisponível aos outros jogadores até que ela seja utilizada pelo jogador que a tirou, e colocada de volta no baralho.

- **Tabuleiro**

Classe que organiza as posições de cada casa no tabuleiro e as instancia passando seus devidos nomes e respectivos valores.

- **Terreno**

Subclasse da classe CasaNegociavel, o terreno representa as casas de imóveis do jogo. Um terreno, além de ter características herdadas de sua superclasse, possui um **valor de aluguel** a depender do **número de casas e hotéis** construídos no momento. Ele também possui um **custo para construir uma casa ou um hotel.**

- **TrataPosicao**

Classe onde serão tratadas as possibilidades de acontecimentos quando um jogador para em determinada casa. A depender do tipo de cada casa, um jogador deverá pagar taxas, comprar imóveis, receber dividendos, construir casas ou hotéis, tirar uma carta de sorte ou revés, etc.

- **Main**

Classe onde são instanciados os jogadores, dados e o tabuleiro do Banco Imobiliário. Em seguida, são simuladas algumas rodadas do jogo, com os resultados sendo exibidos no terminal.

É importante salientar que tal classe ainda será bastante modificada e provavelmente retirada do pacote Model. Além disso, as funções de compra e venda ainda estão ocorrendo de forma automática, sem antes perguntar ao jogador da vez o que ele deseja fazer. Tais detalhes serão corrigidos ao longo das próximas iterações.

- **Testes Unitários**

Todas as classes expostas acima foram devidamente implementadas. As principais classes foram individualmente testadas através do framework JUnit versão 4.

- **BancoTest** → Verifica o saldo inicial do banco no jogo.
- **CasaNegociavelTest** → Testa a transferência de titularidade entre jogadores.
- **ContaBancariaTest** → Testa a transferência de valores entre jogadores.
- **DadoTest** → Testa a geração de valores dentro e fora do intervalo desejado para o dado.
- **JogadorTest** → Testa ida do jogador para a prisão, testa se o jogador está se movimentando ao rolar os dados, testa se seu saldo é inicializado corretamente no início do jogo.
- **TabuleiroTest** → Testa o acesso à casa correta do tabuleiro.
- **TrataPosicaoTest** → Testa o tratamento das posições do tabuleiro de 0-39.

6. Próxima Iteração

6.1. Tarefas que faltou implementar

- Compra de casas e hotéis por parte de um jogador que já possua um certo imóvel, segundo as regras especificadas **(delegada para Daniel).**
- Desenvolvimento da situação de dinheiro insuficiente, a partir da qual o jogador escolherá um imóvel que possua e irá vende-lo por 90% de seu preço, já incluído o valor das casas e hotéis do mesmo. Caso não tenha mais nenhum imóvel ou companhia, o jogador irá à falência **(delegada para Breno).**
- Finalização do jogo, calculando o valor final de cada jogador **(delegada para Tulio).**

É importante salientar que tais funcionalidades serão implementadas para a próxima iteração, mas estarão passíveis de modificação na medida em que surgirem novas demandas por parte dos componentes View e Controller.

6.2. Tarefas planejadas

Para a próxima iteração, será implementado o componente View através de um novo pacote. Nesse pacote, serão desenvolvidas as funcionalidades de visualização em tempo real de itens como os pinos dos jogadores, as casas do tabuleiro, a exposição de cartas, e uma tabela com o saldo atual de cada jogador.