Apresentação do Jogo - Tente Não Jubilar!

Gabriel Bonfim Silva de Moraes - 216111
Leandro Ponsano Corimbaba - 239084

Introdução

O Tente Não Jubilar é uma versão do Monopólio (ou Banco Imobiliário) adaptada para o mundo universitário, em que o objetivo é acumular créditos fazendo aulas e provas. Seus adversários também deverão parar nos institutos da Unicamp para fazer as aulas que você já fez, gerando mais créditos! Vence o indivíduo que conseguir 2000 créditos para passar de ano sem jubilar.

Boa sorte tentando sobreviver a UNICAMP!

Visão Geral

- **Número de jogadores:** 2 ou 3
- **Objetivo:** ganhar créditos (moeda do jogo)
- **Tabuleiro:** composto de 33 casas, com 2 bifurcações
- **Movimentação:** cíclica no sentido horário, 2 dados (2 a 12 casas)
- **Tipos de Célula:** Início, Atraso, Institutos, Empresas e Sorte Ou Revés
- **Após movimentação:** sofre efeito específico da célula

Efeito - Células Especiais

- **Atraso:** jogador fica 2 turnos (Exame) ou 1 turno (Bandeco) sem jogar
- **Início:** jogador ganha 10 créditos
- **Sorte Ou Revés:** situação aleatória, podendo ganhar ou perder créditos

Efeito - Propriedades

- Possuem **custo e aluguel** específicos
- Podem ser **compradas** com créditos
- **Institutos:** jogador é questionado sobre a área do conhecimento do instituto, podendo ganhar ou perder créditos. Paga o preço do aluguel
- **Empresa:** Paga o preço das ações da empresa x quantidade de casas andadas
- **Exemplos:**
  - Dac: custo 300, ações 50
  - FEEC: custo 250, aluguel 80

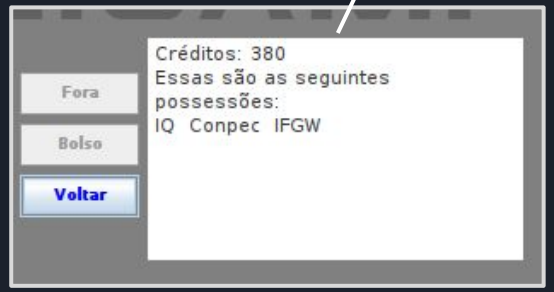
Fim do Jogo

- **Créditos iniciais:** 1000
- **Jubilar:** perder todos os créditos - eliminado do jogo!
- **Obter diploma:** atingir 2000 créditos - vencedor do jogo!

Interface Gráfica (GUI)

Painel de Mensagens

Início



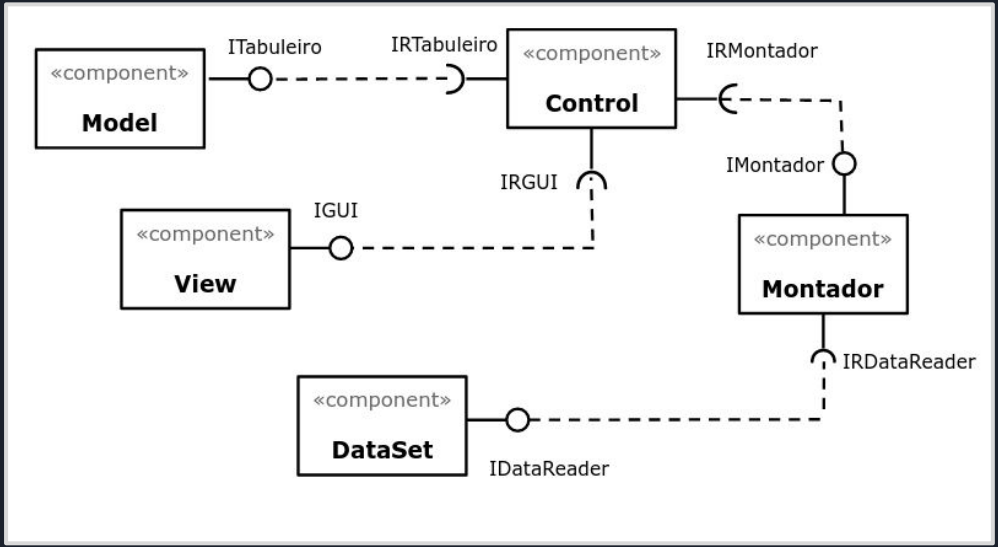
Peça (Jogador)

Botão "Rolar Dados" ou "Passar Turno"

Botão "Informações do Jogador"

Campo para Entrada de Texto

Pattern MVC: Model-View-Control



- **Model:** engloba as células, tabuleiro e jogador
- **View:** interface gráfica
- **Control:** controle dos turnos e rodadas

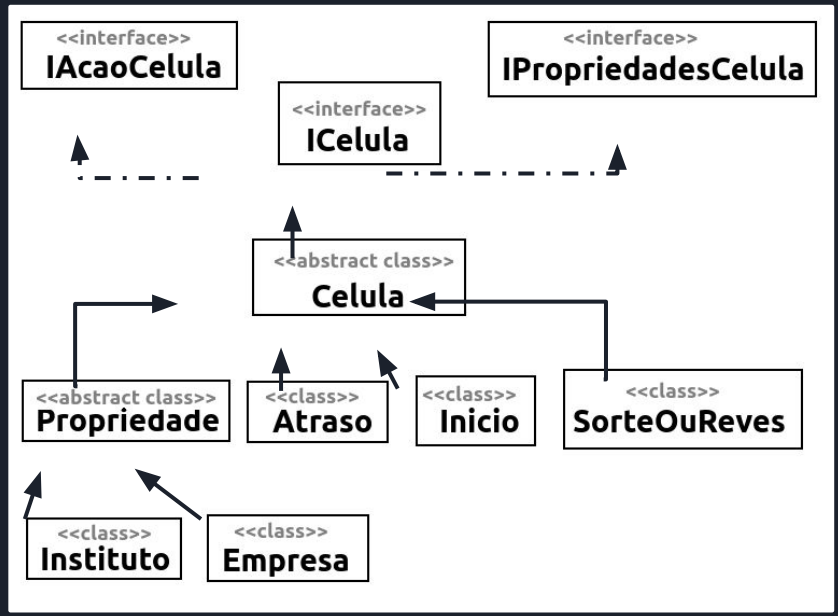
Outros componentes:

- **Montador:** instancia células e monta o tabuleiro
- **DataSet:** Lê arquivos de entrada e passa informações para o Montador

Componente Model - Células
Herança e Polimorfismo

```java
public void montarTabuleiro() {
    guardarTabuleiro();
     (...)
    String tipo = tabuleiro[k][2];
    if (tipo.equals("pi")) {
        String nome = tabuleiro[k][3];
        String arquivo = "data/" + nome + ".csv";
        guardarQuestoes(arquivo);
        tab[i][j] = new Instituto();
    }
    else if (tipo.equals("pe")) {
        tab[i][j] = new Empresa();
    }
    else if (tipo.equals("sr")) {
        tab[i][j] = new SorteOuReves();
    }
    else if (tipo.equals("ex")) {
        tab[i][j]= new Atraso(2);
    }
    else if (tipo.equals("ba")) {
        tab[i][j]= new Atraso(1);
    }
    else if (tipo.equals("nn")) {
        tab[i][j] = new Atraso(0);
    }
    else if (tipo.equals("in")){
        tab[i][j]= new Inicio();
    }
```

- **Tabuleiro:** armazena matriz de células (classe Celula), mas as células são instanciadas pelo montador de acordo com seu tipo



Componente Model - Classes Abstratas e Sobrecarga de Métodos

```java
public abstract class Celula implements ICelula {
    protected String direcao, tipo, nome;

    public void definirPropriedades(String[] props, String[][] questoes) {
        this.direcao = props[1];
        this.tipo = props[2];
        this.nome = props[3];
    }

     (...)

    public abstract Pagamento efeito(IJogador jogador, int casas, String[][] sr, IGUI gui);
}

public abstract class Propriedade extends Celula {
    protected int aluguel, custo;
    protected String dono;

    public void definirPropriedades(String[] props, String[][] questoes) {
        super.definirPropriedades(props, questoes);
        this.custo = Integer.parseInt(props[4]);
        this.aluguel = Integer.parseInt(props[5]);
        this.dono = null;
    }

}
```

- **Método efeito:** abstrato, pois cada herdeira de célula tem sua implementação

- **Método definirPropriedades:** sobrecarregado

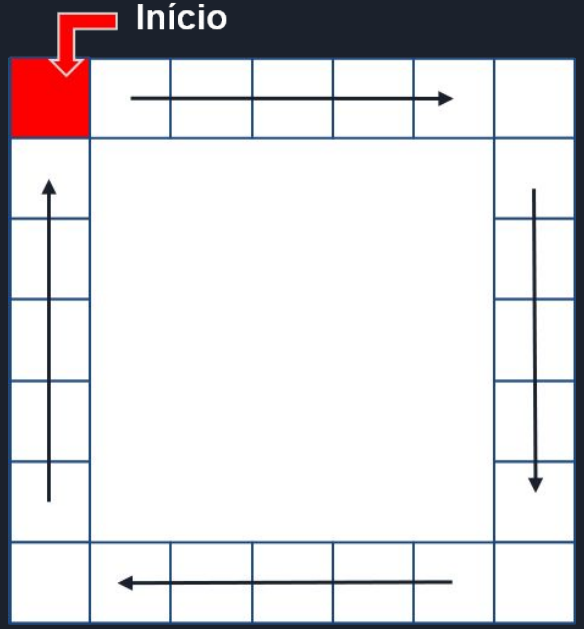
Componente Control - Conexão entre componentes por meio de Interfaces Requeridas

```java
public class Main {

    public static void main(String[] args) {
        IControle controle = new Controle();
        IGUI gui = new GUI();
        IMontador montador = new Montador();
        ITabuleiro tabuleiro = new Tabuleiro();
        IDataReader dataReader = new DataReader();
        montador.connect(dataReader);
        gui.connect(controle);
        controle.connect(gui);
        controle.connect(montador);
        controle.connect(tabuleiro);
        controle.executarJogo();
    }
}
```

- **Montador:** requer serviço do dataReader
- **GUI:** requer serviço do controle
- **Controle:** requer serviços da gui, do montador e do tabuleiro

Evolução, Tentativas e Mudanças de Rota



- **Inicialmente:** não havia perguntas e o tabuleiro era mais simples (ao lado). Após conselho dos monitores, aumentamos a complexidade do jogo
- Começamos sem definir um design de componentes, o que **dificultou muito** o andamento do projeto
- Somente após a adoção do **Pattern MVC** e definição dos componentes o jogo tomou forma
- **Com o tempo,** percebemos que o Jogador não deveria ser um componente, e sim parte do Model
- Para efetuar pagamentos (de um jogador a outro), antes utilizávamos um ArrayList<Object>, porém era necessário a realização de casts. **Mudamos de rota** e decidimos criar uma classe que representasse o Pagamento

Lições Aprendidas, Erros e Planos Futuros

- **A maior lição aprendida foi:** antes de tudo é necessário fazer um planejamento de cima para baixo
- **Deixamos a desejar:** plano de exceções e animação dos dados
- **Projetos futuros:** refatoração com novos patterns aprendidos (como Factory Pattern), adição de jogadores e outros features: empréstimos, negociações, hipotecas etc
- A experiência de implementar do zero um projeto próprio foi extremamente divertida e cativante, pretende-se usar os conhecimentos aprendidos em aula para criar novos projetos

Para rodar o jogo:

- **`git clone`** https://github.com/asimov1239/Tente-Nao-Jubilar.git
- Abra o projeto no Eclipse ou similar
- Entre em `src/control/Main.java`
- Execute (aperte Run Main.java) 

Obrigado e Divirtam-se!
Tente Não Jubilar