Relatório

Simulador de Investimentos com Correção pela Inflação

Vinícius André Veiga Souza Gonçalves *Data: 24/06/2025*

Objetivo

Muitos investidores iniciantes não consideram o impacto da inflação sobre seus investimentos. Esse simulador demonstra, na prática, como a inflação pode reduzir significativamente os ganhos reais ao longo do tempo.

Este projeto foi criado com o objetivo de **comparar o rendimento entre** investimentos em ativos do mercado financeiro, como:

1. **Renda fixa** (**Tesouro Selic**);

2. **Renda variável** (**PIBB11**);

3. **Renda variável nos EUA** (**IVVB11**);

4. **Cripto ativo** (**Bitcoin**).

Além disso, ele foi criado também para **observar os efeitos da inflação sobre os investimentos em determinados períodos pré estabelecidos (5, 10 e 15 anos)**, elevando o nível de comparação entre os ativos.

Para que fosse realizada essa comparação, foi criado uma aplicação orientada a objetos, capaz de simular investimentos em ativos reais ao longo do tempo.

Justificativas

Foi aplicado no projeto os conceitos de **herança**, **interfaces**, **polimorfismo**, **classes abstratas** e uso de estruturas de dados ou collections como **ArrayList**.

Quando analisamos o código, em especial o uso do ArrayList, percebe-se que há valores fixados de juros, tanto de inflação quanto de rendimento dos ativos em cada classe. Isso se deu por causa que fica muito mais simples e rápido de criar esse simulador na linguagem Java. A desvantagem é que o programa fica desatualizado rapidamente.

Já sobre a **classe Ativo**, criei ela para que fosse mais rápido e prático a criação das classes dos ativos específicos, RendaFixa, RendaVariavel, RendaVariavelUSA, Bitcoin, e com um tratamento de exceção caso o usuário insira o valor a ser simulado menor ou igual a zero. Ou seja, os ativos herdaram da classe Ativo. Além disso, nas classes, os nomes dos ativos foram inseridos como constantes, para que não houvessem enganos ou alterações nos construtores.

Sobre a **interface Operacoes**, ela foi utilizada em todas as classes de ativos para o armazenamento de dados, com o método dadosInvestimento, onde foi utilizado uma ArrayList sobre a classe DadosJuros, e o método de cálculo das simulações, mostrarInvestimento, que carregava os dados dos ArrayLists e calculava.

Sobre a escolha dos dados

Por questões de simplicidade e escopo, optei por utilizar dados de rendimento e inflação fixos dentro do código. Essa decisão facilita o desenvolvimento e validação do simulador na linguagem Java, ainda que traga a desvantagem de tornar os dados rapidamente obsoletos.

A começar pela renda fixa, escolhi o **Tesouro Selic** por ser o investimento de renda fixa mais seguro dentre essa classe de ativos tanto no aspecto de rendimento real quanto no de segurança financeira. Quando pensamos em quem são os pagadores dos juros, a poupança parece mais segura que o Selic, mas pelo contrário, é mais fácil um banco quebrar e não pagar do que o governo brasileiro.

Sobre a escolha dos ativos de renda variável brasileira e estadunidense, **PIBB11** e **IVVB11**. Escolhi baseado no quão próximo aqueles ativos replicam os índices de ambas as bolsas e também pensando na conversibilidade real x dólar, sem ter a necessidade de ter que realizar um novo cálculo para conversão cambial.

Já falando dos cripto ativos, decidi escolher o **Bitcoin** pensando no ativo que mais se provou resiliente ao longo do tempo. Por mais novo que ele seja, ainda sim é o cripto ativo mais forte ao longo desses anos e também o que mais se valorizou até o presente momento.

Fontes de dados utilizadas

Banco Central do Brasil: Calculadora do Cidadão
https://www3.bcb.gov.br/CALCIDADAO/publico/exibirFormCorrecaoValores.do?method=exibirFormCorrecaoValores&aba=1

Google Finance: PIBB11 e IVVB11
https://www.google.com/finance/quote/PIBB11:BVMF?sa=X&ved=2ahUKEwjnm6PP2eqMAxVeP7kGHR_EDvoQ3ecFegQIJxAX&window=MAX

https://www.google.com/finance/quote/IVVB11:BVMF?sa=X&ved=2ahUKEwi4mZ_Rke2MAxXfD7kGHTYtFDcQ3ecFegQILxAX&window=MAX

Status Invest: ETF IVVB11
https://statusinvest.com.br/etfs/ivvb11

CoinMarketCap: histórico do Bitcoin
https://coinmarketcap.com/currencies/bitcoin/

Funcionalidades Adicionais com Login e Banco de Dados

Cadastro de Usuários

- Tela para registrar novos usuários com nome e senha.
- Após cadastro, o ID do usuário é automaticamente utilizado para inserir um valor inicial 1.0 na tabela dados.

Login

- Autenticação por user e password.
- Caso as credenciais sejam válidas, os menus restritos (como Simulador) são habilitados.

Persistência de Valores

- Após o login, os usuários podem inserir novos valores (tipo double).
- Os valores são armazenados na tabela dados, relacionados ao campo cadastro_id.

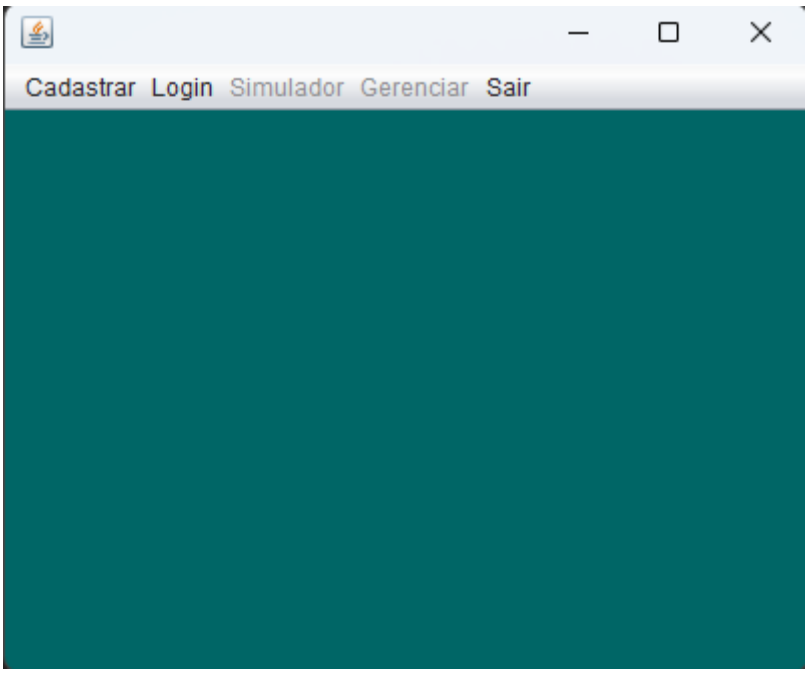
Banco de dados

```
    CREATE TABLE cadastro (
    id      INTEGER NOT NULL,
    user    TEXT NOT NULL,
    password      TEXT NOT NULL,
    PRIMARY KEY(id AUTOINCREMENT)
)

CREATE TABLE dados (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  valor REAL NOT NULL,
  cadastro_id INTEGER NOT NULL,
  FOREIGN KEY (cadastro_id) REFERENCES cadastro(id)
)
```

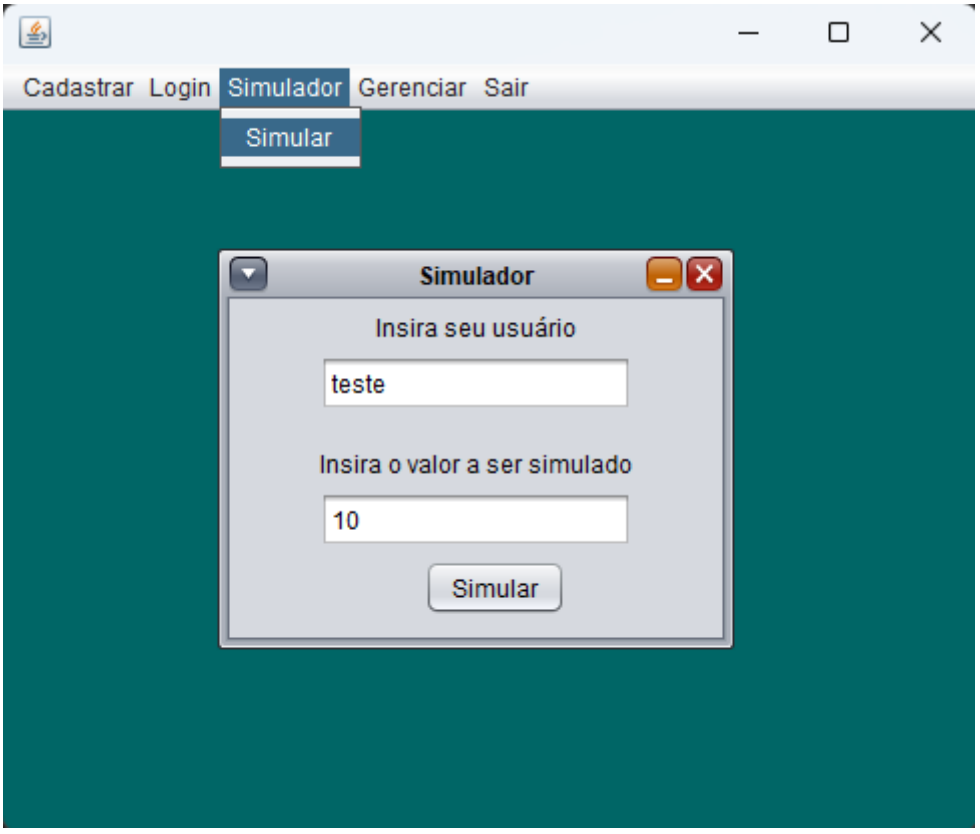
Exemplo de Funcionamento

1. **O usuário se cadastra com um nome e senha.**



2. **Automaticamente, um valor de 1.0 é inserido na tabela dados, vinculado ao usuário.**

3. **Após login, o usuário digita um valor a ser simulado.**



4. **O sistema calcula, para os períodos de 5, 10 e 15 anos:**

Rendimento nominal = valorInvestido * jurosRendimento
Inflação acumulada = valorInvestido * jurosInflacao
Rendimento real = rendimentoNominal – valorCorrigidoInflacao

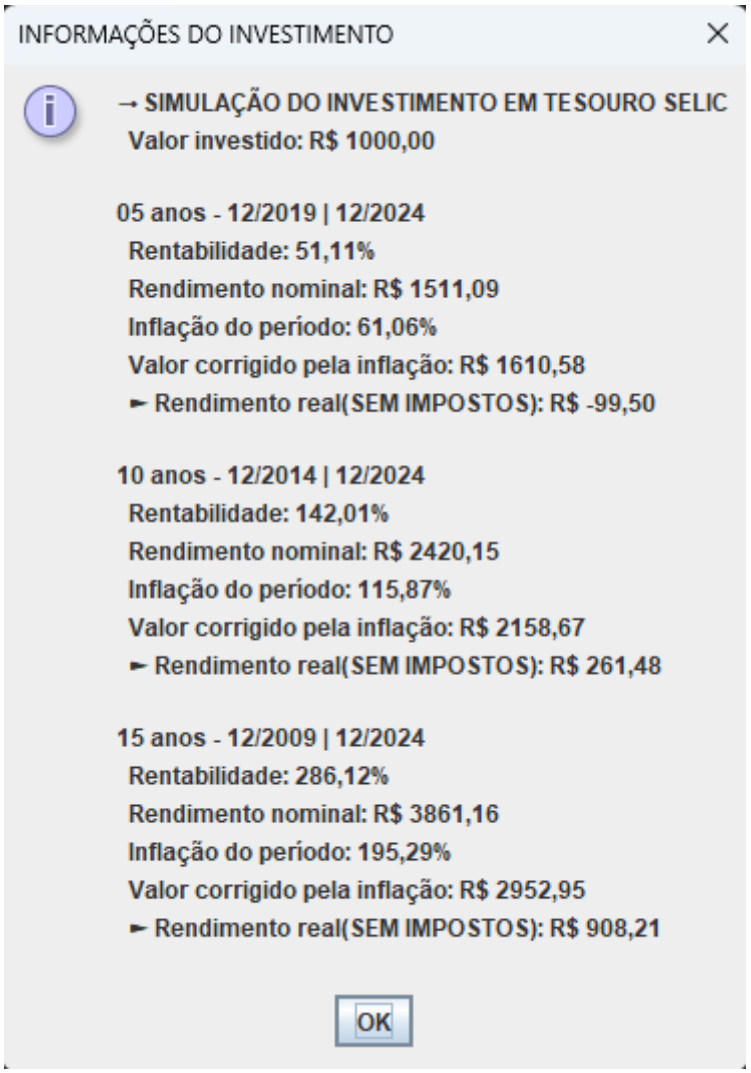
```
for (int i = 0; i < rendimento.size(); i++) {
    DadosJuros dadoRendimento = rendimento.get(i);
    DadosJuros dadoInflacao = inflacao.get(i);

    double valorInvestido = getValorInvestido();

    double rendimentoNominal = valorInvestido * (dadoRendimento.getJuros());
    double valorCorrigidoInflacao = valorInvestido * (dadoInflacao.getJuros());
    double rendimentoReal = rendimentoNominal - valorCorrigidoInflacao;
```

5. **O valor digitado é armazenado no banco com referência ao usuário autenticado.**

6. **Todos os valores são exibidos ao usuário com duas casas decimais, juntamente com os juros do rendimento daquele ativo e a inflação, ambos de cada período.**



Conclusão

O simulador ajuda a entender o impacto da inflação sobre o poder de compra. Ele cumpre seu objetivo reforçando os conceitos fundamentais de programação orientada a objetos juntamente com essa simulação e exemplificação do funcionamento dos juros compostos.

A nova versão do simulador expande o projeto inicial, agora incorporando multiusuário com login, interface gráfica com restrição de acesso, e persistência de dados via banco. Essa evolução permite não apenas a simulação isolada, mas também o controle e análise contínua por parte de cada usuário.