∑igma Bank

- Ideia:
  - Um aplicativo de banco de Investimento e Empréstimo totalmente digital

- Funcionalidades:
  - Transações bancárias
  - Consulta de extrato e saldo
  - Empréstimos
  - Cadastro e login de clientes
  - Conta Bancária
  - Fazer investimentos
  - Proteger os dados dos clientes

- Tecnologias:
  - Desktop: JavaFX
  - Build Sys: Gradle
  - UnitTest: JUnitJupiter

- Classes:
  - **Register - Pessoa:**
    - Atributos: nome: final String, nascimento: final date , E-mail: String, telefone: String, endereço: String, senha: String, UUID(unique): final String
    - Métodos: solicitarFechamentoConta

  - **Client - Cliente Físico (**herda pessoa**):**solicitarFechamentoConta
    - Atributos: CPF: final String, contaCorrente: ContaCorrente, , listaEmprestimo : ArrayList<Emprestimo>, listaInvestimento : ArrayList<Investimento>
    - Métodos: criarContaCorrente, criarContaPoupanca, criarContaSalario,

  - **Admin (**herda pessoa**):**
    - Atributo: CPF: final String
    - Método: habilitaCliente, cadastraAdmin, autorizarEmprestimo, autorizarFechamentoConta

  - **Transação (**struct**):**
    - Lógica: o cliente solicita e o adm analisa e aprova

        - Atributos: UUID, ID remetente, ID destinatario, Valor, Data e hora, tipo, descrição: String

    - **Empréstimo:**
        - Atributos: valor, juros, ID solicitante, data de inicio,data do último update, montante
        - Métodos:
            - pagarParteEmprestimo(): cliente diz um valor que deseja pagar do montante total, então função confere se o cliente tem saldo suficiente pra pagar se não nega
            - updateAmount(): de acordo com a data de início, a cada mês o montante é recalculado de acordo com o juros aplicado
            - simulaPagamento(): o cliente fala em quanto tempo gostaria de pagar, com isso o método calcula o valor de quanto o cliente teria que pagar por mês, de acordo com o montante e o juros
            - -CheckUpdateDate() : retorna um booleano que diz se o update é válido ou não de acordo com as datas

    - **Investimentos (**Factory?**):**
        - Atributos: valor investido, valor, valor retirado, ID cliente, data de ínicio
        - Métodos: Calcular Lucro : valor - valor investido + valor retirado

    - **Asset Investment - Investimento de Recurso (**Herda Investimento e implementa investment Operations**):**
        - Atributos: valor do asset, quantidade do asset,
        - Métodos:

    - **Rate Investment - Investimento de Taxa(**Herda Investimento e implementa investment Operations**):**
        - Atributos: taxa: BigDecimal, valor a ser adicionado:BigDecimal, frequência: ROIFrequencyType, último update:LocalDate
        - Métodos: -checkUpdateDate, Update Value, Retornar Investimento, Investir mais

    - **ContaCorrente(**Herda Conta**):** pode fazer o que quiser
        - Atributos: List<Investimento>, List<Empréstimos>,
        - Metodo: pagarBoleto, gerarCartao, adicionarInvestimento, fazerEmprestimo

- Enum:
    - **Tipo Da Transação:** Transferência,Empréstimo, Saque Investimento
    - **Status Da Transação:** processamento, aguardando
    - **Tipo da rendimento do investimento:** Diário, Mensal, Anual

- Interfaces:
    - **Investment Operations - Operações de Investimento**
        - investMore - método para investir mais dinheiro no investimento
        - retrieveInvestment - método para retornar dinehiro do investimento para a conta bancária

    - **ReaderXML**
        - readFromXML(String pathToXML) - lê todos os objetos do tipo no xml e os retorna
        - readFromXML(String pathToXML, String Identifier) - lê todos os objetos do tipo no xml e retorna os que possuem o identificador igual ao definido

    - **WritterXML**
        - writeToXML(String pathToXML) - escreve um objeto em um xml de acordo com o caminho

- Design Patterns:
    - Factory nos readers and writers
    - Multiton no Investimento
    - Singleton no Banco de Dados

- Exceptions:
    - Problema 1: CPF inválido no register - InvalidCPFException
    - Problema 2: Usuário não encontrado no login
    - Problema 3: Senha incorreta no login
    - Problema 4: A data de nascimento do usuário é menor que 18 anos - InvalidBirthDateException

Requisitos e Equipe

Equipe: Ana Beatriz Hidalgo , Gustavo Esteche Araújo , marcos paulo evers , Rafael Yukio Omiya
data de início: 29/05/2024
+data de entrega: 24/06/2024

Requisitos:

- 3 Interfaces
    - ☑ ~~ReadXML - interface para ler os objetos nos arquivos de xml banco de dados~~
    - ☑ ~~WriteXML - interface para escrever nos arquivos de xml banco de dados~~
    - ☑ ~~InvestmentOperations - interface para obrigar as subclasses de investimento aplicarem as operações básicas de um investimento~~

- 2 Classes abstratas
  - ☑ ~~Investiment~~
  - ☑ ~~Register~~

- Contemplar relacionamentos utilizando estruturas polimórficas
  - ☑ ~~Cliente possui diferentes tipos de investimentos que ele interage~~

- 3 Design patterns
  - ☑ ~~Factory no Writter e Reader~~
  - ☑ ~~Multition no Investiment~~
  - ☑ ~~Singleton no Banco de Dados~~

- 2 Exceptions definidas pelo grupo
  - ☑ ~~userNotFound - no contexto de login~~
  - ☑ ~~wrongPassword - no contexto de login~~
  - ☑ ~~invalidCPFExceptions - no contexto de register~~
  - ☑ ~~invalidBirthDateException - no contexto de register~~

- Leitura e Gravação de arquivos
  - ☑ ~~Banco de Dados, readerXML e writerXML~~

- Interface Gráfica
  - ☑ ~~Tela de registro~~
  - ☑ ~~Tela de login~~
  - ☑ ~~Tela de transferência~~
  - ☑ ~~Tela de Empréstimo~~
  - ☑ ~~Tela de investimento~~