NOMES:

- Euller Henrique Bandeira Oliveira
- Felipe Dantas Vitorino
- Mateus Herrera Gobetti Borges

a) **Para a instituição financeira é muito importante manter um cadastro de todas as contas que foram criadas na instituição. No momento de criação da conta é importante associar uma senha a ela. Essa senha será solicitada antes da execução de qualquer transação.**
   - Para atender ao requisito:
     - As informações de cada conta foram armazenadas em objetos de uma classe 'Conta'. A senha da conta será armazenada como um atributo da classe.
     - Nos métodos de transações é necessário entrar com uma senha como parâmetro. Ao iniciar o método é verificado se a senha é igual à armazenada como atributo.
   - O que definimos:
     - Para armazenamento e gerenciamento das contas foi feita classe 'DadosContas', a classe contém um 'ArrayList' que guarda todas as contas de uma agência e métodos para: cadastrar/inserir, remover, buscar e listar as contas de sua agência. Resumidamente, para cada agência existirá (como atributo desta) um objeto da classe 'DadosContas' que gerenciará e guardará todas as suas contas.

b) **Para cada conta criada é importante saber se a conta está ativa ou já foi desativada (ou seja, o cliente encerrou a conta). Isso ajudará a filtrar as pesquisas. Ainda que um cliente encerre uma conta no banco, é importante manter o registro da conta na base de dados, mas com um indicativo de que ela não está ativa.**
   - Para atender ao requisito:
     - Para saber se a conta está ativa ou não existe um atributo na classe conta (status) que é um booleano, desta forma, se a conta está ativa o atributo será true, caso contrário será false.

c) **As contas da instituição podem pertencer às seguintes categorias: corrente, poupança e salário.**
   - Para atender ao requisito:
     - A classe 'ContaCorrente' terá as especificidades da categoria corrente, a classe 'ContaPoupanca' terá as especificidades da categoria poupança e a classe 'ContaSalario' terá as especificidades da categoria salário.
   - O que definimos:
     - Para esse requisito fizemos a classe 'Conta' (abstrata, porque não será instanciada) que será mãe das classes: 'ContaCorrente', 'ContaPoupanca' e 'ContaSalario'. A classe 'Conta' conterá todos atributos e métodos em comum das subclasses.

d) **Para as contas da categoria corrente, as seguintes informações devem ser armazenadas: nro da conta, saldo atual, data de abertura, data da última movimentação, limite do cheque especial e valor da taxa administrativa.**
   - Para atender ao requisito:
     - A classe 'ContaCorrente' (referente à categoria corrente), terá como atributos limite de cheque especial e valor da taxa administrativa (ambos padrão para todas as

contas, ou seja, static's), e terá como método saque, que sobreposição da classe mãe, por conta das especificidades em relação à classe mãe.

- O que definimos:
  - A classe mãe (Conta, definida no item '**c)**') possui os demais atributos e métodos da conta.

e) Para as contas da categoria poupança as seguintes informações devem ser armazenadas: nro da conta, saldo atual, data de abertura, data da última movimentação, rendimento do mês atual.

- Para atender ao requisito:
  - A classe 'ContaPoupanca' (referente à categoria poupança), terá como atributo o rendimento mensal em porcentagem (que será padrão para todas as contas, ou seja, static's, sendo assim, as poupanças renderam x% ao mês, em que x é determinado pelo atributo em questão), e terá um método que calcula o rendimento do mês atual, e atualiza o saldo de acordo, tal como a quantidade em reais é armazenada como atributo da classe.

- O que definimos:
  - A classe mãe (Conta, definida no item '**c)**') possui os demais atributos e métodos da conta.
  - Foi necessário armazenar a última data que foi calculado o rendimento mensal como atributo dos objetos da classe, para fazer o próximo calculo de rendimento (Melhor explicado na descrição dos requisitos adicionais).

f) Para as contas da categoria salário as seguintes informações devem ser armazenadas: nro da conta, saldo atual, data de abertura, data da última movimentação, limite para saque e limite para transferência.

- Para atender ao requisito:
  - A classe 'ContaSalario' (referente à categoria salário), terá como atributos limite para saque e limite para transferência (ambos padrão para todas as contas, ou seja, static's), e terá como métodos saque e transferência, que serão sobreposição da classe mãe, por conta das especificidades em relação à classe mãe.

- O que definimos:
  - A classe mãe (Conta, definida no item '**c)**') possui os demais atributos e métodos da conta.

g) Os clientes da instituição devem ser registrados com as seguintes informações: CPF, nome, endereço completo, estado civil, escolaridade, data de nascimento.

- Para atender ao requisito:
  - A classe 'Cliente' terá como atributos as informações: data de nascimento e escolaridade. As demais informações são armazenadas através da classe mãe (Pessoa).

- O que definimos:
  - Foi necessário cria uma classe 'Pessoa' que é responsável por generalizar as seguintes informações para clientes e funcionários: CPF, nome, endereço completo, estado civil.
  - Criamos também a classe 'Endereco' para armazenar o endereço completo, não somente do cliente, como também de outras classes como a de funcionários e de agências.

h) É importante também saber em qual agência o cliente foi cadastrado.

- Para atender ao requisito:
  - Os clientes estão instanciados na agência em que foram cadastrados, desta forma é possível, para a agência, saber quem são seus clientes.

- O que definimos:
  - O gerenciamento de cliente e dado por meio de uma classe 'DadosClientes', semelhante ao funcionamento de 'DadosContas', explicado no item '**a)**'.

i) As agências bancárias devem ser previamente cadastras, sendo que cada agência possui um número, um nome fictício e um endereço, sendo que os campos cidade, estado e bairro são campos frequentemente usados para se realizar buscas por agências.

- Para atender ao requisito:
  - Algumas agências serão instanciadas previamente ao iniciar o programa.
  - Número da Agência, nome e endereço serão atributos da classe 'Agencia'.

- O que definimos:
  - As agências são gerenciadas através de uma classe 'DadosAgencias', semelhante à 'DadosContas' explicada no item '**a)**', porém esta será instanciada apenas uma vez como atributo da classe 'Admin', que cuidará do gerenciamento das agências.
  - Na classe 'DadosAgencias' existira um atributo estático, que começa em 1 e é referente ao número da próxima agência a ser cadastrada, sendo que, quando uma nova agência é cadastrada esse valor é atualizado, desta maneira nenhuma agência terá número repetido.
  - Utilizamos, para armazenar o endereço a classe 'Endereco' explicada no item '**g)**', contudo utilizamos, para busca de agências, o número desta, entretanto os campos cidade, estado e bairro estão disponíveis para utilização caso necessário.

j) Cada agência possui um gerente, que é um funcionário nomeado para tal tarefa. Cada gerente pode gerenciar apenas uma agência.

- Para atender ao requisito:
  - No programa não é possível cadastrar dois gerentes com o mesmo número de CPF, desta forma, um gerente só pode ser cadastrado como gerente de uma agência.
  - O gerente será instanciado na classe 'Agencia', desta forma a agência, também saberá quem é seu gerente.
  - O gerente será uma subclasse de funcionário. Na classe 'Funcionario' há o atributo que indica se este é ou não gerente, em caso positivo trataremos de uma instanciação de gerente, assim sendo, eles (os objetos da classe 'Gerente') possuirão como atributo tudo que a classe 'Fucionario' tem e por consequência tudo que a classe 'Pessoa' tem, além dos específicos, entre estes o número da agência que gerencia.

k) Os gerentes são funcionários da empresa que precisam ter armazenados alguns atributos específicos: data de ingresso na carreira de gerente, agência que ele gerencia e se possui curso de formação básico em gerência.

- Para atender ao requisito:
  - Como citado no item '**j)**' gerente herda 'Funcionario', então os atributos específicos de 'Gerente' serão: data de ingresso na carreira, número da agência que ele gerencia, se possui formação e a senha que lhe da acesso às funcionalidades de gerentes do sistema.

l) **Para os funcionários da agência é preciso armazenar as seguintes informações: CPF, nome completo, nro da carteira de trabalho, RG, data de nascimento, endereço, sexo, estado civil, cargo na empresa (existem diferentes cargos além do gerente) e salário.**
   - Para atender ao requisito:
     - Como explicado no item '**g)**' os dados CPF, nome, data de nascimento, endereço e estado civil estão na classe 'Pessoa' que é mãe da classe 'Funcionario', os demais são armazenados como atributos específicos da classe 'Funcionario'.

   - O que definimos:
     - Para armazenamento fizemos 'DadosFuncionarios' de comportamento semelhante à 'DadosContas' explicado no item '**a)**'.

m) **Toda conta bancária deve ter no mínimo um cliente associado. No entanto, uma conta pode ser conjunta. Nesse caso, é possível ter dois clientes associados à mesma conta. Um mesmo cliente pode ter diferentes contas bancárias.**
   - Para atender ao requisito:
     - Para cada objeto da classe 'Conta' há um atributo 'ArrayList' de 'Cliente' que guarda os clientes da conta, tal como um atributo que diz quantos clientes são proprietários da conta e um atributo estático que vale 2 e é referente à quantidade máxima de cliente que uma conta pode ter.
     - Nada no programa impede que um cliente possa ter mais de uma conta, uma vez que a instanciação dele não depende da instanciação de uma conta.
     - Cada conta tem que ter, pelo menos, um cliente associado e não mais que 2

n) **Toda conta bancária está ligada a uma agência bancária, ou seja, a agência na qual ela foi criada.**
   - Para atender ao requisito:
     - Como explicado no item '**a)**', as contas criadas ficam instanciadas em 'DadosContas', que por sua vez está instanciada em uma das agências, sendo assim, toda conta criada está contida na sua agência de criação.

o) **Uma vez que um cliente tenha uma conta bancária ele pode efetuar diferentes movimentações financeiras, conhecidas como transações bancárias. Cada transação bancária está relacionada a uma conta bancária. Além disso, é importante armazenar a data em que a transação foi realizada, o valor da transação e o canal onde foi feito a transação (internet banking, caixa eletrônico ou caixa físico). Toda vez que uma transação é realizada o saldo do cliente deve ser atualizado.**
   - Para atender ao requisito:
     - Todo tipo de transação é feito por métodos na classe 'Conta' (quando há alguma especificidade, nas subclasses, por meio de sobreposição), esses métodos, além de atualizarem o saldo da conta, também criam um objeto da classe 'Transacao', criada para armazenar dados das transações, incluindo o canal utilizado para realização da transação.

   - O que definimos:
     - Foi criada a classe 'Transacao' que armazena todas as informações das transações.

p) **Cada transação possui um tipo. Os tipos de transação atualmente disponíveis são: saque, transferência, depósito e pagamento. Em cada uma dessas transações é importante armazenar apenas o valor da transação. Com o objetivo de simplificar o trabalho não vamos armazenar os atributos específicos de cada tipo de transação, embora esse seja um requisito interessante em problemas reais.**

- Para atender ao requisito:
  - Como explicado no item anterior, foi feito uma classe ‘Transacao’ que armazena todos os dados das transações.

- O que definimos:
  - Foi decidido colocar algumas informações extras na classe ‘Transacao’, como: saldo antes da transação, saldo depois da transação, valor da transação, data da transação, tipo (saque, pagamento, etc), canal (internet banking, caixa eletrônico, etc), e status (se a transação deu certo ou não).

Requisitos adicionais:

1. **Calculo do rendimento mensal:**
   - Criamos um método que calcula e atualiza o saldo da conta. Esse método será chamado assim que o cliente acessa sua conta, ou seja, há também um atributo que guarda mês e ano do último calculo de rendimento. Ao chamar o método primeiro calcula-se quantos meses se passaram (de acordo com a data da máquina e da última data calculada), e faz-se o rendimento referente a essa quantidade de meses.

2. **Validação de CPF:**
   - Criamos a classe ‘ValidaCpf’ que só possui três métodos estáticos, sendo eles:
     1. ehNumerico – Que verifica se a string candidata a CPF só possui números;
     2. validaCpf – Que verifica se a string candidata a CPF corresponde a um CPF válido;
     3. toString – Que coloca o CPF no padrão: 000.000.000-00.
   - Essa classe é usada para verificar se um CPF inserido é válido e em caso positivo colocá-lo em um padrão para facilitar as comparações feitas entre CPF’s.