Seja o sistema bancário que implementamos na aula 1, seguindo a abordagem de programação estruturada. As funcionalidades desejadas são:

1. Cadastro de clientes armazenando nome, CPF e número da conta, com cadastro de conta com número único, saldo inicial zero e cpf do cliente
2. Registro de saques
3. Registro de depósitos
4. Transferência de valores entre os clientes.

Os dados não precisam ser salvos em arquivo (podem ficar apenas na memória do computador).

Agora vamos transformá-lo em um programa que segue a abordagem orientada a objetos, considerando as classes descritas a seguir. Lembre-se de que **além dos atributos e métodos indicados**, você poderá criar outros que possibilitem/facilitem a implementação das funcionalidades requeridas.

As classes são:

1 - `GerenciadorContaBancaria`: esta classe deverá também ser o nome do projeto. Ela será usada para iniciar a execução do sistema

Atributos de classe:

- `banco` (classe `Banco`): objeto usado para invocar as funcionalidades de gerenciamento de conta bancária.

Métodos de classe:

- `main`: inicia a execução do programa e mostra o menu de opções, invocando o método de execução. Todos as funcionalidades requeridas pelo usuário deverão ser delegadas ao atributo `banco`.
    - Argumentos: `args` (array de `String`) - recebe os argumentos de linha de comando, quando existem.
    - Retorno: não há

2 - `Banco`: deverá conter os métodos para o gerenciamento da conta, bem como armazenar as informações de contas e clientes

Atributos (de instância):

- `código`: (inteiro) - armazena o código do banco (valor 1)
- `clientes`: (array de `Cliente`) - armazena as informações de clientes
- `contas`: (array de `Conta`) - armazena as informações sobre as contas bancárias

Métodos:

- Construtor: inicia os atributos de classe
    - Argumentos: não há
- `cadastro`: cadastra um novo cliente, armazenando as informações nos arrays clientes e contas. O cadastro das informações dos clientes deverão ser delegados para a classe Cliente. O número da conta deverá ser criado fornecido de modo incremental (iniciando em 1).
    - Argumentos: não há
    - Retorno: não há
- `saque`: realiza o saque da conta bancária a partir do número da conta e do valor a ser sacado (a serem pedidos para o usuário). A atualização do saldo deve ser delegada à classe `Conta`. Deve ser informado ao cliente se o saque pôde ser realizado com sucesso
    - Argumentos: não há
    - Retorno: não há
- `deposito`: realiza um depósito na conta bancária a partir do número da conta e do valor a ser depositado (a serem pedidos para o usuário). A atualização do saldo deve ser delegada à classe `Conta`. Deve ser informado ao cliente se o depósito pôde ser realizado com sucesso.
    - Argumentos: não há
    - Retorno: não há
- `transferencia`: realiza uma transferência entre a conta do depositante e a conta do recebedor de um valor a ser depositado (a serem pedidos para o usuário). As atualizações dos saldos do depositante e do recebedor deve ser delegada à classe Conta. Deve ser informado ao cliente se o saque pôde ser realizado com sucesso.
    - Argumentos: não há
    - Retorno: não há

3 - `Cliente`: deverá conter os métodos para o cadastro dos dados, bem como informar o nome e o cpf

Atributos (de instância):

- `nome`: (String) - armazena o nome
- `cpf`: (`String`) - armazena o CPF
- `numero`: (inteiro) - armazena o número da conta

Métodos (de instância):

- Construtor: construtor padrão, não é necessário implementar
    - Argumentos: não há
- `cadastro`: pede os dados ao usuário e armazena nos atributos
    - Argumentos: não há
    - Retorno: não há

- `getNome`: informa o nome do cliente
    - Argumentos: não há
    - Retorno: `String`
- `getCpf`: informa o CPF do cliente.
    - Argumentos: não há
    - Retorno: `String`

4 - `Conta`: deverá conter os métodos para o gerenciamento do saldo da conta

Atributos (de instância):

- `numero`: (inteiro) - número da conta corrente
- `saldo`: (número real) - armazena o saldo
- `cpf`: (`String`) - CPF do titular

Métodos:

- Construtor: inicia o saldo e atrela o CPF
    - Argumentos: numero (`inteiro`) e cpf (`String`)
- `saque`: realiza o saque de um valor da conta bancária, verificando se há saldo suficiente, indicando o sucesso da operação como retorno.
    - Argumentos: `valor` (número real) - valor a ser sacado
    - Retorno: `true` se o saque foi realizado com sucesso
- `deposito`: realiza um depósito na conta bancária a partir do valor a ser depositado Deve ser informado ao cliente se o depósito pôde ser realizado com sucesso.
    - Argumentos: `valor` (número real) - valor a ser depositado
    - Retorno: `true` se o depósito foi realizado com sucesso