# PILHA E FILA - Exercício

## Estrutura de Dados e Armazenamento

Crie um novo projeto chamado exercicio-conta-bancaria.

Atenção, pois em todo lugar que chamar o insert da fila / push da pilha, é recomendável que seja feito dentro de um try/catch porque estamos lançando exceção IllegalStateException.

1. Copie para esse projeto a classe FilaObj<T> e a classe PilhaObj<T>.

2. Crie uma classe chamada **ContaBancaria.**

    a. Atributos: numero (Integer) e saldo (Double), encapsulados. Forneça os getters e setters para esses atributos.
    b. Crie o construtor e o toString().
    c. Métodos:
        i. debitar(Double valor) – este método deve ser do tipo Boolean. Verifica se a conta tem saldo suficiente para fazer o débito. Se não tiver, exibe uma mensagem de "Saldo insuficiente" e retorna false. Se tiver saldo suficiente, atualiza o valor do saldo, exibe o saldo após a atualização e retorna true.
        ii. creditar(Double valor) – este método deve ser void. Atualiza o valor do saldo e exibe o saldo após a atualização.

3. Crie uma classe chamada **Operacao**.
    a. Atributos: objConta (que é um objeto da classe ContaBancaria), tipoOperacao (String para indicar se é "debito" ou "credito") e valor (Double).
    b. Crie o construtor e o toString().

4. Crie uma classe App, executável.

    Nessa classe, vamos simular o funcionamento de um banco que permite que operações sejam desfeitas e que permite que operações sejam agendadas para serem executadas num momento mais oportuno (de madrugada, por exemplo).

    Dessa forma, todas as operações realizadas (débito e crédito) deverão ser empilhadas para que possam ser desfeitas.

    E, a pedido do usuário, algumas operações podem ser agendadas. Essas serão enfileiradas para serem executadas posteriormente. Vamos assumir que essas operações não poderão ser desfeitas.

    a. No método main, crie um objeto PilhaObj<Operacao> e um objeto FilaObj<Operacao>. Pode criar com tamanho 10 cada um.
    b. Declare uma variável que é um contador de operações realizadas e inicialize o contador com zero (vai funcionar como um contador de operações empilhadas).
    c. Crie 2 contas bancárias, com valores de numero e saldo a sua escolha. Para facilitar, vamos imaginar que existem apenas essas 2 contas bancárias.

d. Fique num loop, exibindo o seguinte menu para o usuário:
   1- Debitar valor
   2- Creditar (Depositar) valor
   3- Desfazer operação
   4- Agendar operação
   5- Executar operações agendadas
   6- Sair

   Leia a opção escolhida pelo usuário.

   Utilizando switch case, implemente o que cada opção faz:

   1- Debitar valor: peça para o usuário digitar o número da conta da qual vai debitar, e o valor a ser debitado. Execute a operação, chamando o método do objeto correspondente. Se o método debitar retornar true, crie um objeto Operacao, contendo o objeto conta associado, "debito" para tipo de operação e o valor da operação. Empilhe esse objeto na pilha de operações e incremente o contador de operações. No momento de empilhar, precisa do try/catch.

   2- Depositar valor: peça para o usuário digitar o número da conta onde vai depositar, e o valor a ser depositado. Execute a operação, chamando o método do objeto correspondente. Crie um objeto Operacao, contendo o objeto conta associado, "credito" para tipo de operação e o valor da operação. Empilhe esse objeto na pilha de operações e incremente o contador de operações.

   3- Desfazer operação: solicite que o usuário digite a quantidade de operações a serem desfeitas. Se a quantidade for maior do que o contador de operações, exiba a mensagem de que essa quantidade é inválida. Desempilhe uma operação por vez da pilha de operações, desfazendo a operação. Desfaça a quantidade solicitada pelo usuário e subtraia a quantidade desfeita do contador de operações. Obs: desfazer a operação significa fazer o contrário do que ela faz, ou seja, se a operação empilhada era um debito, tem que fazer um credito, e vice-versa.

   4- Solicite que o usuário digite qual a operação desejada (debito ou credito), número da conta e o valor. Crie um objeto Operacao, contendo o objeto conta associado, tipo de operação e o valor da operação. Enfileire esse objeto (não se esqueça do try/catch).

   5- Se a fila de operações agendadas estiver vazia, exiba uma mensagem de que não há operações agendadas. Caso contrário, esvazie a fila, executando cada uma das operações.

   6- Sinalize que é o fim do programa.