PARADIGMAS DE LINGUAGENS DE PROGRAMAÇÃO – 2019-1

Profa. Heloisa - Terceiro Trabalho – Programação OO e Concorrente

==========

Escreva um programa em JAVA para simular operações de transferência entre contas bancárias, usando threads.

- Definir uma classe **`Banco`**, cujo objeto será compartilhado pelas threads, que representam os correntistas.
- A classe **`Banco`** deve possuir pelo menos 5 contas, que são representadas em uma estrutura do tipo **`ArrayList`**, em que os objetos representam cada conta, com dois campos: nome do titular da conta e saldo. Todas as contas tem um valor inicial de R$10.000,00.
- Além dos campos de dados, a classe **`Banco`** deve conter o método que faz transações de **`transferencia(de,para,valor)`**.
- A operação de transferência é feita de uma conta de origem (correspondente a thread que pede a transferência) para uma conta destino, que é uma das outras 4 contas do banco e que deve ser gerada aleatoriamente.
- A operação de transferência consiste simplesmente em diminuir o valor especificado da conta de origem e somá-lo na conta destino. O valor transferido deve ser gerado aleatoriamente.
- Criar uma linha de execução para cada conta (cada correntista) como uma instância da classe **`TransferThread`**, usando a classe **`Thread`** ou a interface **`Runnable`**. O método **`run()`** deve fazer várias transferências, repetindo a execução do método em um processo iterativo.
- Sincronizar o método para transferências, para garantir que essa operação não seja feita simultaneamente por mais de uma linha de execução.
- Usar os métodos **`wait()`** e **`notify()`** para fazer o sincronismo de cooperação, de tal forma que a transação não seja feita a partir de uma conta que não tenha saldo suficiente.
- A cada operação, imprimir uma mensagem indicando quais as contas envolvidas e o valor envolvido.
- O método **`main()`** deve instanciar e inicializar o objeto da classe **`Banco`**, as threads e chamar os métodos **`start()`**.
- As threads podem parar de duas maneiras (escolher apenas uma): interrompendo o loop com Crtl+C, como no exemplo abaixo, ou definindo um número fixo de operações para cada conta (semelhante ao que foi feito no exemplo da corrida dado em aula), sendo no mínimo 10 operações para cada conta.

Exemplo da classe TransferThread:

```
class TransferThread extends Thread {
   public TransferThread (Banco b, int de) {

      banco = b;
      contaorigem = de;
  }
public void run ( ) {
    try {
       while (!interrupted ( )) {
           // define a conta destino aleatoriamente
           // define o valor a ser transferido aleatoriamente
           // faz a operação de transferência
          sleep(1000);
       }
  } catch (InterruptedException e) {}
}
private Banco banco;
private int contaorigem;
}
```

Observações:

• Os trabalhos podem ser feitos em duplas, sendo as mesmas do trabalho anterior;

• Entregar, no ambiente da disciplina no ava, listagem do programa (arquivo .java), programa compilado (arquivo .class), relatório com documentação do programa (explicação do funcionamento de cada classe) e exemplos de execução.

• Data de entrega: 10/07/2019.