UAST
Unidade Acadêmica de Serra Talhada - PE
UFRPE
Desde 2006

BSi
Bacharelado em Sistemas de Informação

MPOO

site: https://sites.google.com/site/profricodemery/mpoo

Disciplina: Modelagem e Programação Orientada a Objetos (MPOO)

Profº: Richarlyson D'Emery **Data:** **16 / 04 / 2014**

Aluno: _____

2ª LISTA DE EXERCÍCIOS

- Crie uma pasta em Meus documentos chamada NomeSobrenome.
- No Eclipse crie um novo projeto contendo um projeto chamado NomeSobrenome, o sistema deverá ter um pacote para cada questão (questao1, questao2, ...). O pacote deverá ter sua(s) classe(s) de acordo com a descrição do problema.
- Ao finalizar a prova exporte o projeto contendo todos os arquivos necessários no formato .zip e o copie na pasta criada em Meus Documentos.
- Sugestão: Utilize um *pendrive* e salve as implementações a cada modificação, caso aconteça alguma falha no computador o trabalho será preservado.

1º) Responda:

a) Para que serve o this?

b) Qual a diferença entre um método e um método construtor? Como podemos identificá-los?

2º) Crie um cenário OO destacando uma classe, um objeto, dois atributos e um método.

3º) Escreva o diagrama de classes (no astah) e um programa em Java de uma classe chamada Conta, que possui os atributos privados nomeCliente e saldo, a classe possui um construtor que inicializa o nome do cliente na criação do objeto e saldo R$0,00. Faça também as classe Poupança e Corrente que herda de Conta. A Poupança possui os atributos, saldo e digitoVerificador. O saldo só poderá ser aumentado pelo método depositar() e diminuído pelo método sacar(). Crie uma aplicação chamada Banco onde um objeto de Poupança possui os seguintes atributos: José, R$200,00 de saldo e digitoVerificador 1. Imprima as informações referentes ao objeto Poupança. Sabendo ainda que existe conta corrente, implemente na aplicação Banco um método que realiza a transferência de valores entre duas contas quaisquer. Ilustre a realização de uma transferência e os valores antes e depois de realizá-la.

4º) O Problema 4 (anexo abaixo) foi solucionado seguindo uma solução estruturada, implemente-o fazendo uso dos conceitos envolvidos na OO (classe, objeto, atributo e método construtor).

Problema 3: Somando uma sequência

Os programas também podem ser constituídos por estruturas condicionais (**Se... Então... Senão...**), que pode realizar instruções a depender da condição analisada. Vejamos agora, algumas soluções que também podem ser constituídos por estruturas de repetição (**Repita... Até... / Enquanto... / Para...**).

Para exercitar, primeiramente vamos solucionar o seguinte problema: como fazer para somar de 1 até 5? Existem várias formas de resolver o problema, a solução mais simples é somar os valores e atribuir a uma variável chamada soma, vejamos o algoritmo:

```
Início
  Declare somatorio

  somatorio ← (1 + 2 + 3 + 4 + 5)

  Escreva somatorio

Fim
```


O algoritmo é bem simples, mas e se o problema fosse: como fazer para somar de 1 até 10? ou como fazer para somar de 1 até 100?, ou até mesmo como fazer para somar de 1 até 1000? Podemos observar que o simples algoritmo utilizado passaria a ter o problema de somar os vários números do intervalo, sendo assim começamos a observar que poderíamos passar a utilizar alguma outra solução, e para isso utilizaremos a estrutura de repetição **Repita... Até ...**

E é na fase de processamento que pensamos qual solução adotar. Para isso utilizaremos uma variável auxiliar “cont” que servirá de controle para a repetição. Vejamos como ficaria o algoritmo para realizar a soma de 1 até 5:

```
Início

  Declare somatorio, cont

  somatorio ← 0

  cont ← 1

  REPITA

    somatorio ← (somatorio + cont)
```


Observe que a repetição acontecerá até que cont >= 6, ou seja, enquanto cont < 6 o programa permanecerá no laço.

```
    cont ← (cont + 1)
  ATÉ (cont >=6)
  Escreva somatorio
Fim
```

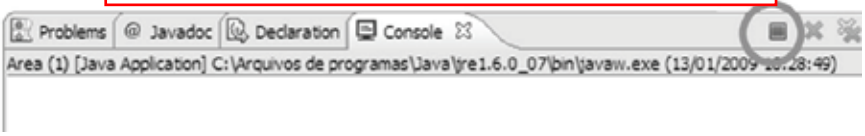
Podemos observar que ao final do programa teremos o valor da **somatorio** igual a **15** (1+2+3+4+5), isso porque a estrutura de repetição **REPITA ... ATÉ...** é executada enquanto a condição é **verdadeira** e os valores de **somatorio** e **cont** são atualizados a cada interação do **laço** até que a condição (**cont >= 6**) seja satisfeita.

Logo se o problema fosse: *"Como fazer para somar de 1 até 10?"*, bastava substituir a condição do laço, ou seja, **cont >= 11**. Se fosse até 100 a condição seria **cont >= 101** e assim sucessivamente.

Fique por dentro!

Se por um descuido a condição tivesse sido colocada de forma **a nunca** interromper o laço, ou seja, a condição da estrutura de repetição nunca venha a ser **falsa**, o programa entraria no que chamamos de ***laço infinito*** e, desta forma, o programa continuaria executando por muito tempo, até esgotar os recursos do computador, a exemplo a memória RAM, que apesar de possui muitos *gigabytes*, é finita.

Dessa forma, devemos evitar cometer tais erros para que o programa rode. Mas se acontecer você poderá interromper a execução do programa clicando no botão parar no Console do IDE Eclipse:

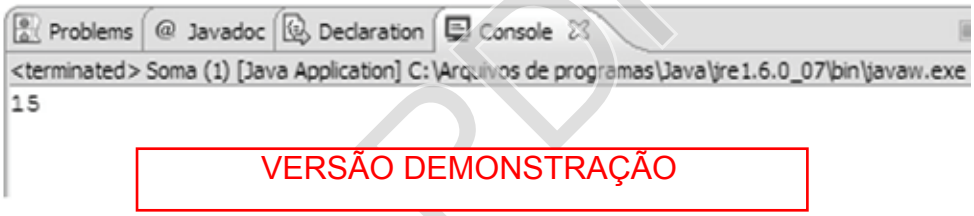


Vejamos o código fonte do programa em Java:

```
public class Soma {
  public static void main (String [] args) {
    int somatorio, cont;
    somatorio =0;
    cont = 1;
```

```
    do {
      somatorio = (somatorio + cont);
      cont = (cont + 1);
    } while (cont < 6);
    System.out.println(somatorio);
  }
}
```

Após compilado vejamos o resultado:



Problema 4: Otimizando o problema anterior

Podemos aperfeiçoar o problema anterior de forma que somemos um intervalo cujo limite superior é dependerá da entrada fornecida, ou seja,

somatorio = 1 + 2 + 3 + 4 + 5 + ... + **N**

Onde:

- **N** é o limite superior, e
- **somatorio** é a soma dos números do intervalo [1, N].

Vejamos como ficará o algoritmo:

```
Início
  Declare somatorio, cont, N
  N ← 5
  somatorio ← 0
  cont ← 1
  REPITA
```

Limite Superior

Condição de parada depende de N. Observe que a repetição acontecerá até que cont >= N, ou seja, enquanto cont <= N o programa permanecerá no laço.

```
    somatorio ← (somatorio + cont)
    cont ← (cont + 1)
  ATÉ (cont >=(N+1)
  Escreva somatorio
Fim
```


No algoritmo acima a condição **cont >= (N+1)**, permite que a repetição seja realizada de forma a somar todos os valores do intervalo. E aproveitamos para destacar que **cont** é o **incremento** da estrutura de repetição.

Vejamos o código fonte do programa em Java:

```
public class Soma2 {
  public static void main (String [] args) {
    int somatorio, cont, N;
    N=5;
    somatorio =0;
    cont = 1;
    do {
      somatorio = (somatorio + cont);
      cont = (cont + 1);
    } while (cont <= N);
    System.out.println(somatorio);
  }
}
```