PUC-Minas - Ciência da Computação
AED1 – Estudo Dirigido 05

Tema: Introdução à programação II
Atividade: Funções e procedimentos recursivos em Java

01.) Editar e salvar um esboço de programa em Java:

```
/**
  * Exemplo0081
  *
  * @author
  * @version 01
  */

// ---------------------------------------------  dependencias

import IO.*;

// ---------------------------------------------  definicao da classe principal

public class Exemplo0081
{
// ---------------------------------------------  definicao de metodo auxiliar

   public static void metodo01 ( int x )
   {
     // repetir enquanto valor maior que zero
        while ( x > 0 )
        {
          // mostrar valor
             IO.println ( “Valor = ” + x );
          // passar ao próximo
             x = x - 1;
        } // fim se
   } // fim metodo01( )

// ---------------------------------------------  definicao do metodo principal

   /**
     * main() – metodo principal
     */
   public static void main ( String [ ] args )
   {
     // identificar
        IO.println ( "EXEMPLO0081 - Programa em Java" );
        IO.println ( "Autor: _____" );
        IO.println ( );
     // executar o metodo auxiliar
        metodo01 ( 5 );
     // encerrar
        IO.pause   ( “Apertar ENTER para terminar.” );
   } // fim main( )
} // fim class Exemplo0081
```

02.) Compilar o programa novamente.
Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.
Se não houver erros, seguir para o próximo passo.
Em caso de dúvidas, consultar a apostila, recorrer aos monitores ou apresentá-las ao professor.

03.) Executar o programa.
Observar as saídas.
Registrar os dados e os resultados.
Em caso de erro (ou dúvida), usar comentários para registrar a ocorrência e,
posteriormente, tentar resolvê-lo (ou esclarecer a dúvida).

04.) Copiar a versão atual do programa para outra nova – Exemplo0082.java.

05.) Editar mudanças no nome do programa e versão.
Acrescentar um método para mostrar valores inteiros decrescentes.
Na parte principal, editar a chamada do método para o novo.
Prever novos testes.

```
public static void metodo02 ( int x )
{
  // definir dado local
     int y = x;
  // repetir enquanto valor maior que zero
     while ( y > 0 )
    {
      // mostrar valor
         IO.println ( “Valor = ” + y );
      // passar ao próximo
         y = y - 1;
    } // fim se
} // fim metodo02( )
```

06.) Compilar o programa novamente.
Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.
Se não houver erros, seguir para o próximo passo.

07.) Executar o programa.
Observar as saídas.
Registrar os dados e os resultados.

08.) Copiar a versão atual do programa para outra nova – Exemplo0083.java.

09.) Editar mudanças no nome do programa e versão.
Acrescentar um método para mostrar valores inteiros decrescentes.
Na parte principal, editar a chamada do método para o novo.
Prever novos testes.

```
// ------------------------------------------------  definicao de metodo auxiliar

   public static void metodo03 ( int x )
   {
     // definir dado local
       int y;
     // repetir enquanto valor maior que zero
       for ( y = x; y > 0; y = y - 1 )
       {
         // mostrar valor
            IO.println ( "Valor = " + y );
       } // fim se
   } // fim metodo03( )
```

10.) Compilar o programa novamente.
Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.
Se não houver erros, seguir para o próximo passo.

11.) Executar o programa.
Observar as saídas.
Registrar os dados e os resultados.

12.) Copiar a versão atual do programa para outra nova – Exemplo0084.java.

13.) Editar mudanças no nome do programa e versão.
Acrescentar um método para mostrar valores inteiros crescentes.
Na parte principal, editar a chamada do método para o novo.
Prever novos testes.

```
// ------------------------------------------------  definicao de metodo auxiliar

   public static void metodo04 ( int x )
   {
     // definir dado local
       int y;
     // repetir enquanto valor maior que zero
       for ( y = 1; y <= x; y = y + 1 )
       {
         // mostrar valor
            IO.println ( "Valor = " + y );
       } // fim se
   } // fim metodo04( )
```

14.) Compilar o programa novamente.
Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.
Se não houver erros, seguir para o próximo passo.

15.) Executar o programa.
Observar as saídas.
Registrar os dados e os resultados.

16.) Copiar a versão atual do programa para outra nova – Exemplo0085.java.

17.) Editar mudanças no nome do programa e versão.
Acrescentar outro método para mostrar valores da sequência: 1 3 5 7 ...
Na parte principal, editar a chamada do método para o novo.
Prever novos testes.

```
// ------------------------------------------------  definicao de metodo auxiliar

   public static void metodo05 ( int x )
   {
     // definir dado local
        int y;
        int z = 1;
     // repetir enquanto valor maior que zero
        for ( y = 1; y <= x; y = y + 1 )
        {
          // mostrar valor
             IO.println ( "Valor = " + z );
           // passar ao proximo
             z = z + 2;
        } // fim se
   } // fim metodo05( )
```

18.) Compilar o programa novamente.
Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.
Se não houver erros, seguir para o próximo passo.

19.) Executar o programa.
Observar as saídas.
Registrar os dados e os resultados.

20.) Copiar a versão atual do programa para outra nova – Exemplo0086.java.

21.) Editar mudanças no nome do programa e versão.
Acrescentar outro método para mostrar valores da sequência: 1 2 4 6 8 ...
Na parte principal, editar a chamada do método para o novo.
Prever novos testes.

```
// ------------------------------------------------  definicao de metodo auxiliar

   public static void metodo06 ( int x )
   {
     // definir dado local
        int y;
        int z = 1;
     // repetir enquanto valor maior que zero
        IO.println ( "Valor = " + z );
        z = 2;
        for ( y = 1; y < x; y = y + 1 )
        {
          // mostrar valor
             IO.println ( "Valor = " + z );
          // passar ao proximo
             z = z + 2;
        } // fim se
   } // fim metodo06( )
```

OBS.: Reparar que a repetição se encerrará para o valor igual a x.
A repetição será feita, portanto, (x-1) vezes.

22.) Compilar o programa novamente.
Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.
Se não houver erros, seguir para o próximo passo.

23.) Executar o programa.
Observar as saídas.
Registrar os dados e os resultados.

24.) Copiar a versão atual do programa para outra nova – Exemplo0087.java.

25.) Editar mudanças no nome do programa e versão.
Acrescentar outro método para mostrar valores da sequência:
1/1 2/3 4/5 6/7 8/9 ...
Na parte principal, editar a chamada do método para o novo.
Prever novos testes.

```
// ------------------------------------------------  definicao de metodo auxiliar

   public static void metodo07 ( int x )
   {
     // definir dado local
        int y;
        int z;
     // repetir enquanto valor maior que zero
        IO.println ( "Valor = 1/1" );
        z = 3;
        for ( y = 2; y <= x; y = y + 1 )
        {
          // mostrar valor
             IO.println ( "Valor = " + (z-1) + "/" + z );
          // passar ao proximo
             z = z + 2;
        } // fim se
   } // fim metodo07( )
```

OBS.: Reparar que a repetição se iniciará para o valor igual a 2, e terminará em (x).
A repetição será feita, portanto, (x-1) vezes.

26.) Compilar o programa novamente.
Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.
Se não houver erros, seguir para o próximo passo.

27.) Executar o programa.
Observar as saídas.
Registrar os dados e os resultados.

28.) Copiar a versão atual do programa para outra nova – Exemplo0088.java.

29.) Editar mudanças no nome do programa e versão.
Acrescentar outro método para somar os valores na sequência:
1/1 + 2/3 + 4/5 + 6/7 + 8/9 ...
Na parte principal, editar a chamada do método para o novo.
Prever novos testes.

```
// ----------------------------------------------  definicao de metodo auxiliar

   public static void metodo08 ( int x )
   {
     // definir dado local
        int y;
        int z;
        double soma = 1.0;
     // repetir enquanto valor maior que zero
       IO.println ( "Valor = 1/1" );
       z = 3;
       for ( y = 2; y <= x; y = y + 1 )
        {
          // mostrar valor
             IO.println ( "Valor = " + (z-1) + "/" + z );
          // somar uma parcela
             soma = soma + 1.0 * (z-1) / z;
          // passar ao proximo
             z = z + 2;
        } // fim se
        // mostrar o resultado
          IO.println ( "Soma = " + soma );
   } // fim metodo08( )
```

OBS.: Reparar que o valor inicial da soma será 1.0,
e um valor a menos da quantidade de vezes deverá ser descontado.

30.) Compilar o programa novamente.
Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.
Se não houver erros, seguir para o próximo passo.

31.) Executar o programa.
Observar as saídas.
Registrar os dados e os resultados.

32.) Copiar a versão atual do programa para outra nova – Exemplo0089.java.

33.) Editar mudanças no nome do programa e versão.
Acrescentar uma função para calcular a soma dos valores na sequência:
1/1 + 2/3 + 4/5 + 6/7 + 8/9 ...
Na parte principal, editar a chamada do método para o novo.
Prever novos testes.

```
// ------------------------------------------------  definicao de metodo auxiliar

   public static double metodo09 ( int x )
   {
    // definir dado local
      int y;
      int z;
     double soma = 1.0;
    // repetir enquanto valor maior que zero
     IO.println ( "Valor = 1/1" );
     z = 3;
     for ( y = 1; y < x; y = y + 1 )
      {
        // mostrar valor
          IO.println ( "Valor = " + (z-1) + "/" + z );
       // somar uma parcela
          soma = soma + 1.0 * (z-1) / z;
       // passar ao proximo
          z = z + 2;
      } // fim se
    // retornar a resposta
      return ( soma );
   } // fim metodo09( )
```

Na parte principal, incluir a chamada à função:

```
// mostrar o resultado
  IO.println ( "Soma = " + metodo09 ( 5 ) );
```

34.) Compilar o programa novamente.
Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.
Se não houver erros, seguir para o próximo passo.

35.) Executar o programa.
Observar as saídas.
Registrar os dados e os resultados.

36.) Copiar a versão atual do programa para outra nova – Exemplo0090.java.

37.) Editar mudanças no nome do programa e versão.

Acrescentar uma função para calcular a soma dos valores na sequência:
1/1 + 2/3 + 4/5 + 6/7 + 8/9 ...
Na parte principal, editar a chamada do método para o novo.
Prever novos testes.

```
// ------------------------------------------------  definicao de metodo auxiliar

   public static double metodo10 ( int x )
   {
    // definir dado local
      int y;
      int numerador     = 1;
      int denominador = 1;
      double soma = (double) numerador / denominador;
    // repetir enquanto valor maior que zero
      IO.println ( “”+soma );
     for ( y = 1; y < x; y = y + 1 )
      {
        // passar ao proximo numerador
          numerador = y * 2;
        // passar ao proximo denominador
          denominador = denominador + 2;
        // mostrar valor
          IO.println ( “+ ” + numerador + “/” + denominador );
        // somar uma parcela
          soma = soma + 1.0 * numerador / denominador;
      } // fim se
     // retornar a resposta
       return ( soma );
   } // fim metodo10( )
```

Na parte principal, incluir a chamada à função
cujo valor deve ser guardado em variável, definida previamente:

```
// definir dado para receber o resultado
   double resposta;
```

e mais adiante:

```
// receber o resultado
   resposta = metodo10 ( 5 );
// mostrar o resultado
   IO.println ( “Soma = “ + resposta );
```

38.) Compilar o programa novamente.

Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.
Se não houver erros, seguir para o próximo passo.

39.) Executar o programa.

Observar as saídas.
Registrar os dados e os resultados.

Exercícios:
DICAS GERAIS: Consultar o Anexo Java 02 na apostila para outros exemplos.
Prever, realizar e registrar todos os testes efetuados.

01.) Fazer um programa (Exemplo0091)
para ler uma quantidade inteira do teclado e, mediante o uso de um método,
mostrar essa quantidade em valores múltiplos de 5 em ordem crescente.

02.) Fazer um programa (Exemplo0092)
para ler uma quantidade inteira do teclado e, mediante o uso de um método
mostrar essa quantidade em valores pares múltiplos de 5 em ordem crescente.

03.) Fazer um programa (Exemplo0093)
para ler uma quantidade inteira do teclado e, mediante o uso de um método,
mostrar essa quantidade em valores ímpares múltiplos de 5 em ordem decrescente.

04.) Fazer um programa (Exemplo0094)
para ler um valor inteiro do teclado e, mediante o uso de um método,
mostrar essa quantidade em valores crescentes nos denominadores
(sequência dos inversos) múltiplos de 5: 1 1/5 1/10 1/15 1/20 1/25 ...

05.) Fazer um programa (Exemplo0095)
para ler um valor inteiro do teclado e, mediante o uso de um método,
mostrar essa quantidade em valores crescentes nos denominadores
da sequência: 1 1/5 1/25 1/125 ...
DICA: Usar Math.pow (x, y) para calcular a potência.

06.) Fazer um programa (Exemplo0096) com função para calcular
a soma dos primeiros valores pares positivos começando em 2
e não múltiplos de 5.
Testar essa função para quantidades diferentes.

07.) Fazer um programa (Exemplo0097) com função para calcular
a soma dos inversos (1/x) dos primeiros valores pares positivos começando em 2
e não múltiplos de 5.
Testar essa função para quantidades diferentes.

08.) Fazer um programa (Exemplo0098) com função para calcular
a soma dos primeiros números naturais começando em 1.
Testar essa função para quantidades diferentes.

09.) Fazer um programa (Exemplo0099) com função para calcular
a soma dos quadrados dos primeiros números naturais começando em 1.
Testar essa função para quantidades diferentes.

10.) Fazer um programa (Exemplo0100) com função para calcular
a soma dos inversos dos primeiros números naturais começando em 1.
Testar essa função para quantidades diferentes.

Tarefas extras

E1.) Fazer um programa para ler um número inteiro do teclado e,
mediante o uso de uma função, calcular e mostrar o fatorial desse valor:

$n! = n * (n-1) * (n-2) * ... * 3 * 2 * 1$ se $n>0$

E2.) Fazer um programa para ler uma quantidade inteira do teclado e,
mediante o uso de uma função, calcular e mostrar o resultado de

$f (n) = (1+2/3) * (1+3/4) * (1+4/5) * ...$