PUC-Minas - Ciência da Computação
AED1 – Estudo Dirigido 02

Tema: Introdução à programação
Atividade: Testes, repetições e alternativas em Java

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

```
/**
  * Exemplo0021
  *
  * @author
  * @version 01
  */

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

import IO.*;

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

public class Exemplo0021
{
// ---------------------------------------------  definicao do metodo principal

   /**
    * main() – metodo principal
    */
   public static void main ( String [ ] args )
   {
     // definir dados
        int x;
     // identificar
        IO.println ( "EXEMPLO0021 - Programa em Java" );
        IO.println ( "Autor: _____" );
     // ler valor inteiro do teclado
        x = IO.readint ( “Entrar com um valor inteiro: ” );
        IO.println ( “Valor lido = ” + x );
     // encerrar
        IO.pause   ( “Apertar ENTER para terminar.” );
   } // fim main( )
} // fim class Exemplo0021

// ---------------------------------------------- documentacao complementar
//
// ---------------------------------------------- historico
//
// Versao      Data        Modificacao
// 0.1                     __ / __      esboco
//
// ---------------------------------------------- testes
//
// Versao      Teste
// 0.1                     01. ( ___ )     identificacao de programa e leitura
//
```

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 resultados.

```
// ---------------------------------------------- testes
//
// Versao      Teste
// 0.1                  01. ( OK )     identificacao de programa e leitura
//                                     Valores previstos: 5, 0, -5
//
```

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 – Exemplo0022.java.

05.) Editar mudanças no nome do programa e versão.
Acrescentar uma repetição para mostrar todos os valores inteiros menores ou iguais a ele, e maiores que zero.
Prever novos testes.

```
    // ler valor inteiro do teclado
      x = IO.readint ( "Entrar com um valor inteiro: " );
     while ( x > 0 )
     {
        IO.println ( "Valor lido = " + x );
        x = x – 1;
     } // fim repetir

// ---------------------------------------------- documentacao complementar
//
// ---------------------------------------------- historico
//
// Versao      Data       Modificacao
// 0.1                  __ / __     esboco
// 0.2                  __ / __     mudança de versão
//
// ---------------------------------------------- testes
//
// Versao      Teste
// 0.1                  01. ( OK )     identificacao de programa e leitura
//                                     Valores previstos: 5, 0, -5
//
// 0.2                  01. ( ___ )    identificacao de programa
//                                     Valores previstos: 5, 0, -5
//
```

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 resultados.

```
// ---------------------------------------------- testes
//
// Versao      Teste
// 0.1                  01. ( OK )     identificacao de programa
//                                     Valores previstos: 5, 0, -5
//
// 0.2                  01. ( OK )     identificacao de programa
//                                     Valores previstos:
//                                     com 5:
//                                     com 0:
//                                     com-5:
//
//
```

08.) Copiar a versão atual do programa para outra nova – Exemplo0023.java.

09.) Editar mudanças no nome do programa e versão.
Substituir a repetição para mostrar todos os valores inteiros
maiores que zero e menores ou iguais a ele, em ordem crescente.
Prever novos testes.
DICA: Será necessário definir outra variável, para não perder o dado (x).

```
// definir dados
   int x;
   int y;


// ler valor inteiro do teclado
   x = IO.readint ( "Entrar com um valor inteiro: " );
  y = 1;
  while ( y <= x )
  {
     IO.println ( "" + y );
     y = y + 1;
  } // fim repetir
```

```
// ---------------------------------------------- documentacao complementar
```

```
//
// ------------------------------------------------ historico
//
// Versao      Data        Modificacao
// 0.1                     __ / __      esboco
// 0.2                     __ / __      mudança de versão
//
// ------------------------------------------------ testes
//
// Versao      Teste
// 0.1                     01. ( OK )     identificacao de programa e leitura
//                                        Valores previstos: 5, 0, -5
//
// 0.2                     01. ( ___ )    identificacao de programa
//                                        Valores previstos: 5, 0, -5
//
// 0.3                     01. ( ___ )    identificacao de programa
//                                        Valores previstos: 5, 0, -5
//
```

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 resultados.

12.) Copiar a versão atual do programa para outra nova – Exemplo0024.java.

13.) Editar mudanças no nome do programa e versão.
Substituir a forma de repetição para outra mais compacta.
Prever novos testes.
DICA: Será necessário definir outra variável, para controlar a repetição.

```
// ler valor inteiro do teclado
  x = IO.readint ( "Entrar com um valor inteiro: " );
for ( y = 1; y <= x; y = y + 1 )
{
   IO.println ( "" + y );
} // fim repetir
```

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 resultados.

16.) Copiar a versão atual do programa para outra nova – Exemplo0025.java.

17.) Editar mudanças no nome do programa e versão.
Modificar a forma de repetição para outra mais compacta,
em ordem decrescente.
Prever novos testes.
DICA: Será necessário definir outra variável, para controlar a repetição.

```
// ler valor inteiro do teclado
  x = IO.readint ( "Entrar com um valor inteiro: " );
 for ( y = x; y >= 1; y = y - 1  )
 {
   IO.println ( "" + y );
 } // fim repetir
```

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 resultados.

20.) Copiar a versão atual do programa para outra nova – Exemplo0026.java.

21.) Editar mudanças no nome do programa e versão.
Acrescentar outra repetição para ler uma dada quantidade de valores
cujos valores deverão ser testados e mostrados,
apenas se forem maiores que zero.
Prever novos testes.
DICA: Será necessário definir outra variável, para o dado (z) a ser lido.

```
// definir dados
  int x, y;
 int z;


// ler valor inteiro do teclado
  x = IO.readint ( "Entrar com a quantidade: " );
  for ( y = x; y >= 1; y = y - 1 )
  {
    z = IO.readint ( "Entrar com valor inteiro: " );
    if ( z > 0 )
    {
      IO.println ( "" + z );
    } // fim se
 } // fim repetir
```

22.) Copiar a versão atual do programa para outra nova – Exemplo0027.java.

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

Modificar o teste para mostrar dentre os valores lidos
apenas os que forem maiores que zero e menores que 10.
Prever novos testes.
DICA: Será necessário usar o conectivo lógico && (e).

```
// definir dados
 int x, y, z;


// ler valor inteiro do teclado
   x = IO.readint ( "Entrar com a quantidade: " );
   for ( y = x; y >= 1; y = y - 1 )
   {
     z = IO.readint ( "Entrar com valor inteiro: " );
    if ( z > 0 && z < 10 )
     {
       IO.println ( "" + z );
     } // fim se
  } // fim repetir
```

24.) 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.

25.) Executar o programa.
Observar as saídas.
Registrar os valores usados para testes e os resultados.

26.) Copiar a versão atual do programa para outra nova – Exemplo0028.java.

27.) Editar mudanças no nome do programa e versão.
Modificar o teste para mostrar dentre os valores lidos
apenas os que forem menores ou iguais a zero ou maiores ou iguais a 10.
Prever novos testes.
DICA: Será necessário usar o conectivo lógico || (ou).

```
// ler valor inteiro do teclado
   x = IO.readint ( "Entrar com a quantidade: " );
   for ( y = x; y >= 1; y = y - 1 )
   {
     z = IO.readint ( "Entrar com valor inteiro: " );
    if ( z <= 0 || z >= 10 )
     {
       IO.println ( "" + z );
     } // fim se
  } // fim repetir
```

28.) 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.

29.) Executar o programa.
Observar as saídas.
Registrar os valores usados para testes e os resultados.

30.) Copiar a versão atual do programa para outra nova – Exemplo0029.java.

31.) Editar mudanças no nome do programa e versão.
Modificar o teste para mostrar dentre os valores lidos
apenas os que forem pares (múltiplos de 2).
Prever novos testes.
DICA: Serão necessários usar o operador % (resto inteiro) e
o operador lógico == (comparação por igualdade).

```
// ler valor inteiro do teclado
  x = IO.readint ( "Entrar com a quantidade: " );
  for ( y = x; y >= 1; y = y - 1 )
  {
    z = IO.readint ( "Entrar com valor inteiro: " );
   if ( z % 2 == 0 )
    {
      IO.println ( "par: " + z );
    } // fim se
 } // fim repetir
```

32.) 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.

33.) Executar o programa.
Observar as saídas.
Registrar os valores usados para testes e os resultados.

34.) Copiar a versão atual do programa para outra nova – Exemplo0030.java.

35.) Editar mudanças no nome do programa e versão.
Modificar o teste para separar dentre os valores lidos
ímpares e pares.
Prever novos testes.
DICA: Serão necessários usar o operador % (resto inteiro) e
o operador lógico != (comparação por diferença).

```
// ler valor inteiro do teclado
  x = IO.readint ( "Entrar com a quantidade: " );
  for ( y = x; y >= 1; y = y - 1 )
  {
    z = IO.readint ( "Entrar com valor inteiro: " );
   if ( z % 2 != 0 )
    {
      IO.println ( "impar: " + z );
    }
    else
    {
      IO.println ( "par:     " + z );
    } // fim se
  } // fim repetir
```

36.) Compilar e testar o programa novamente.
Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.
Se não houver erros, testar o programa, anotar os dados e os resultados e seguir em frente.

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 (Exemplo0031) capaz de lidar com valores inteiros (**int**).
Ler a quantidade, primeiro; e depois, outros valores, um por vez;
mostrar cada valor e dizer se são positivos ou negativos. Desconsiderar o zero.

02.) Fazer um programa (Exemplo0032) capaz de lidar com valores inteiros (**int**).
Ler a quantidade, primeiro; e depois, outros valores, um por vez;
e mostrar apenas os valores entre -15 e 35, excluindo esses.

03.) Fazer um programa (Exemplo0033) capaz de lidar com valores inteiros (**int**).
Ler a quantidade, primeiro; e depois, outros valores, um por vez;
e mostrar apenas os valores que não estão entre -15 e 35, excluindo esses.

04.) Fazer um programa (Exemplo0034) capaz de lidar com valores inteiros (**int**).
Ler a quantidade, primeiro; e depois, outros valores, um por vez;
e mostrar apenas os valores que não estão entre -15 e 35, incluindo esses.
DICA: Usar o operador de negação ! (não) antes da condição:

```
if ( ! (-15 <= z && z <= 35) )
```

05.) Fazer um programa (Exemplo0035) capaz de lidar com valores inteiros (**int**).
Ler a quantidade de vezes, primeiro; e dois outros valores, depois, em cada repetição.
Mostrar os valores apenas se o primeiro for par e o segundo for ímpar.

06.) Fazer um programa (Exemplo0036) capaz de lidar com valores inteiros (**int**).
Ler a quantidade de vezes, primeiro; e dois outros valores, depois, em cada repetição.
Mostrar os valores apenas se o primeiro for maior que zero e menor que o segundo.

07.) Fazer um programa (Exemplo0037) capaz de lidar com dados com valores inteiros (**int**).
Ler a quantidade de vezes, primeiro; e dois outros valores, depois, em cada repetição.
Mostrar apenas os valores em que ambos estiverem entre no intervalo [-15 : 35].

08.) Fazer um programa (Exemplo0038) capaz de lidar com valores inteiros (**int**).
Ler a quantidade de vezes, primeiro; e dois outros valores, depois, em cada repetição.
Mostrar apenas os valores em que ambos estiverem entre [-15 :35] e forem ímpares.

09.) Fazer um programa (Exemplo0039) capaz de lidar com valores inteiros (**int**).
Ler a quantidade de vezes, primeiro; e dois outros valores, depois, em cada repetição.
Mostrar apenas os valores em que ambos não estiverem entre [-15 : 35] e forem pares.

10.) Fazer um programa (Exemplo0040) capaz de lidar com valores inteiros (**int**).
Ler a quantidade de vezes, primeiro; e dois outros valores, depois, em cada repetição.
Separar e mostrar todos os valores em que ambos não estiverem entre [-15 : 35] ,
não forem pares e também não forem negativos.

Tarefas extras

E1.) Fazer um programa capaz de lidar com valores reais (**double**).
Ler a quantidade de vezes, primeiro; e dois outros valores reais, depois, em cada repetição.
Separar e mostrar todos os valores em que ambos estiverem entre [-15 : 35] ,
e o segundo maior que o primeiro.

E2.) .Fazer um programa capaz de lidar com valores reais (**double**).
Ler a quantidade de vezes, primeiro; e dois outros valores reais, depois, em cada repetição.
Separar e mostrar todos os valores em que ambos não estiverem entre [-15 : 35] ,
sejam diferentes, as partes inteiras não sejam pares e o segundo maior que o primeiro.
DICA: Para obter a parte inteira de um valor real,
usar a conformação de tipo (***type casting***) para inteiro:

```
double x;
int y;

. . .

y = (int) x;
```