

Tema: Introdução à programação

Atividade: Montagem de programas em Java

- 01.) Editar e salvar um esboço de programa em Java,
o nome do arquivo deverá ser Exemplo0001.java,
tal como o nome da classe abaixo,
concordando maiúsculas e minúsculas, sem espaços em branco:

```
/**
 * Exemplo0001
 *
 * @author
 * @version 01
 */

// ----- dependencias

import IO.*;

// ----- definicao da classe principal

public class Exemplo0001
{
// ----- definicao de metodos

// ----- definicao do metodo principal

    /**
     * main() – metodo principal
     */
    public static void main ( String [ ] args )
    {
        IO.println ( "EXEMPLO0001 - Programa em Java" );
        IO.println ( "Autor: _____" );
        IO.pause ( "Apertar ENTER para terminar." );
    } // fim main()
} // fim class Exemplo0001

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

02.) Compilar o programa.

Se houver erros, identificar individualmente a referência para a linha onde ocorrem.

Consultar atentamente o modelo acima na linha onde ocorreu o erro (e também linhas próximas), editar as modificações necessárias.

Compilar novamente e proceder assim até que todos os erros tenham sido resolvidos.

Se não houver erros, seguir para o próximo passo.

DICA: Se precisar de ajuda sobre como proceder a compilação, consultar os vídeos com as demonstrações sobre algumas formas para fazê-lo.

SUGESTÃO: Para se acostumar ao tratamento de erros, registrar a mensagem de erro (como comentário) e o que foi feito para resolvê-lo.

03.) Executar o programa.

Observar as saídas.

Registrar os resultados.

```
// ----- testes
//
// Versao    Teste
// 0.1       01. ( OK )      identificacao de programa
//
```

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) – Exemplo0002.java.

05.) Editar mudanças no nome do programa e versão,

conforme as indicações a seguir,

tomando o cuidado de modificar todas as referências,

inclusive as presentes em comentários.

Incluir na documentação complementar as alterações feitas,

acrescentar indicações de mudança de versão e

prever novos testes.

```

/**
 * Exemplo0002
 *
 * @author
 * @version 02
 */

// ----- dependencias

import IO.*;

// ----- definicao da classe principal

public class Exemplo0002
{
// ----- definicao de metodos

// ----- definicao do metodo principal

    /**
     * main() – metodo principal
     */
    public static void main ( String [ ] args )
    {
        IO.println ( "EXEMPLO0002 - Programa em Java" );
        IO.println ( "Autor: _____" );
        IO.pause ( "Apertar ENTER para terminar." );
    } // fim main()
} // fim class Exemplo0002

// ----- documentacao complementar
//
// ----- historico
//
// Versao      Data      Modificacao
// 0.1         01/08         esboco
// 0.2         01/08         mudança de versão
//
// ----- testes
//
// Versao      Teste
// 0.1         01. ( OK )      identificacao de programa
// 0.2         01. ( ____ )    identificacao de programa
//

```

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
// 0.2          01. ( OK )      identificacao de programa
//
```

- 08.) Copiar a versão atual do programa para outra (nova) – Exemplo0003.java.

- 09.) Acrescentar ao programa a definição de um dado (x)
para armazenar valor inteiro:

```
public static void main ( String [ ] args )
{
    // definicao de dados
    int x = 0; // valor inicial definido

    IO.println ( "EXEMPLO0003 - Programa em Java" );
    IO.println ( "Autor: _____" );
    IO.println ( "x = " + x ); // exibir valor do dado
    IO.pause ( "Apertar ENTER para terminar." );
} // fim main()
```

DICA: O melhor lugar para se colocar as definições de dados
é próximo aos cabeçalhos (assinaturas) dos métodos (ou da classe),
pois ficará mais fácil localizá-los no futuro.

SUGESTÃO: Recomenda-se uma rápida compilação
para verificar se a introdução de um novo dado
ocorre sem inserir erros no programa existente.
Recomenda-se também definir valores iniciais
para os dados que servirão como variáveis,
segundo o tipo valor que armazenarão.
Portanto, se quiser experimentar a forma de definição abaixo,
ela não deverá ter qualquer consequência sobre o resultado.

```
int x; // forma alternativa, sem definir o valor inicial...

...

x = 0; // e definir o valor depois
IO.println ( "x = " + x ); // exibir valor do dado apos a definicao
```

- 10.) Copiar a versão atual do programa para outra (nova) – Exemplo0004.java.

11.) Acrescentar ao programa uma modificação do valor armazenado no dado (x):

```
public static void main ( String [ ] args )
{
    // definir dados
    int x = 0;

    IO.println ( "EXEMPLO0004 - Programa em Java" );
    IO.println ( "Autor: _____" );

    x = 5;                // atribuir novo valor ao dado
    IO.println ( "x = " + x ); // exibir novo valor do dado

    IO.pause ( "Apertar ENTER para terminar." );
} // fim main()
```

DICA: A atribuição (ou transferência de valor) para o dado será sempre feita com a referência para o dado (nome ou destino) à esquerda do sinal de atribuição ('='), e o valor a ser transferido à direita deste (fonte).

SUGESTÃO: Recomenda-se uma rápida compilação para verificar se a introdução de um novo comando ocorre sem inserir erros no programa existente.

12.) Acrescentar ao programa uma forma de se exibir o valor do dado (x):

```
public static void main ( String [ ] args )
{
    // definir dado
    int x = 0;

    IO.println ( "EXEMPLO0004 - Programa em Java" );
    IO.println ( "Autor: _____" );

    IO.println ( "x (inicial) = " + x ); // exibir valor do dado antes de modificar
    x = 5;                               // atribuir valor ao dado
    IO.println ( "x ( atual ) = " + x ); // exibir valor do dado depois de modificar

    IO.pause ( "Apertar ENTER para terminar." );
} // fim main()
```

DICA: A exibição (ou transferência para a saída padrão) de valor de um dado poderá ser feita, sempre que necessário, para se consultar o que estiver armazenado. Como a saída exige uma conversão para os símbolos correspondentes aos padrões da língua do usuário, faz-se necessário converter valores numéricos em equivalentes literais (caracteres), o que será indicado mediante a sequência (String) entre aspas que antecederá a referência para o valor a ser convertido (x). O sinal (+) indicará, neste caso, uma operação particular de composição (chamada de *concatenação*) da sequência com a conversão do valor. Para essa operação ser bem sucedida, a sequência deverá sempre vir primeiro, e entre aspas, se tiver um conteúdo constante.

SUGESTÃO: Recomenda-se preceder a exibição do valor pelo nome escolhido para o mesmo.

- 13.) Acrescentar na documentação complementar a indicação de teste do dado introduzido.

```
// ----- testes
//
// Versao    Teste
// 0.1       01. ( OK )      identificacao de programa
// 0.2       01. ( OK )      identificacao de programa
//           02. ( ____ )    introdução de dado inteiro
//
```

- 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.

```
// ----- testes
//
// Versao    Teste
// 0.1       01. ( OK )      identificacao de programa
// 0.2       01. ( OK )      identificacao de programa
//           02. ( OK )      introdução de dado inteiro
//
```

- 16.) Copiar a versão atual do programa para outra (nova) – Exemplo0005.java.

- 17.) Acrescentar a atribuição inicial de valor ao dado e suspender a execução da atribuição mediante uso de comentário (//), conforme indicado abaixo.

```
public static void main ( String [ ] args )
{
    // definir dado
    int x = 5; // forma alternativa para definir dado com valor inicial

    IO.println ( "EXEMPLO0005 - Programa em Java" );
    IO.println ( "Autor: _____" );

    //   x = 5;           // atribuir valor ao dado apos definido

    IO.println ( "x = " + x ); // exibir valor do dado

    IO.pause ( "Apertar ENTER para terminar." );
} // fim main()
```

- 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.

```
// ----- testes
//
// Versao    Teste
// 0.1       01. ( OK )      identificacao de programa
// 0.2       01. ( OK )      identificacao de programa
//           02. ( OK )      introdução de dado inteiro
//           03. ( OK )      introdução de dado inteiro com valor inicial
//
```

- 20.) Alterar a atribuição inicial, conforme indicado abaixo.

```
public static void main ( String [ ] args )
{
    // definir dado
    int x = 2+3; // forma alternativa para definir valor inicial por expressao aritmetica

    IO.println ( "EXEMPLO0005 - Programa em Java" );
    IO.println ( "Autor: _____" );

    IO.println ( "x = " + x ); // exibir valor inicial do dado

    X = IO.readint ( "Entrar com um valor inteiro: " );

    IO.println ( "x = " + x ); // exibir valor lido

    IO.pause ( "Apertar ENTER para terminar." );
} // fim main()
```

- 21.) 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.

- 22.) Executar o programa.
Observar as saídas.
Registrar os resultados.

```
// ----- testes
//
// Versao    Teste
// 0.1       01. ( OK )      identificacao de programa
// 0.2       01. ( OK )      identificacao de programa
//           02. ( OK )      introdução de dado inteiro
//           03. ( OK )      introdução de dado inteiro com valor inicial
//           04. ( OK )      introdução de dado inteiro com valor inicial com operação
//
```

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 (Exemplo0006) capaz de guardar e mostrar um valor literal (**char**).

DICAS: Copiar o modelo do Exemplo0005, e realizar as modificações sugeridas abaixo.

Para definir (em substituição à definição anterior):

```
char x;
```

Para atribuir um valor constante:

```
x = 'A'; // um único símbolo, sempre entre apóstrofes
```

Para ler um caractere por vez e armazená-lo, antes de mostrar seu valor:

```
x = IO.readchar ( "Entrar com um caractere: " );
```

Para mostrar, manter o comando para exibição:

```
IO.println ( "x=" + x );
```

Repetir o comando anterior para exibir o valor lido.

02.) Fazer um programa (Exemplo0007) capaz de guardar e mostrar um valor real (**double**).

DICAS: Repetir o procedimento indicado no exercício anterior.

Para definir:

```
double x;
```

Para atribuir um valor constante:

```
x = 43.21; // parte inteira separada da fracionária por ponto
```

Mostrar o valor atribuído. (Ver comando para exibição de valor acima.)

Para ler um dado real e armazená-lo, antes de mostrar seu valor:

```
x = IO.readdouble ( "Entrar com um valor real: " );
```

Repetir o comando para exibir o valor lido.

03.) Fazer um programa (Exemplo0008) capaz de guardar e mostrar um valor lógico (**boolean**).

DICAS: Para definir:

```
boolean x;
```

Para atribuir:

```
x = true; // ou false
```

Mostrar o valor atribuído.

Para ler um dado lógico e armazená-lo, antes de mostrar seu valor:

```
x = IO.readboolean ( "Entrar com um valor lógico: " );
```

Repetir o comando para exibir o valor lido.

OBS.: Para testar. usar apenas as palavras **true** ou **false**.

04.) Fazer um programa (Exemplo0009) capaz de guardar e mostrar uma sequência de literais (**String**).

DICAS: Para definir:

```
String x;
```

Para atribuir:

```
x = "43.21"; // uma sequência de literais (caracteres) deve sempre vir entre aspas
```

Mostrar o valor atribuído.

Para ler um dado literal e armazená-lo, antes de mostrar seu valor:

```
x = IO.readString ( "Entrar com uma cadeia de caracteres: " );
```

Repetir o comando para exibir o valor lido.

- 05.) Fazer um programa (Exemplo0010) capaz de guardar e mostrar uma sequência de literais (**String**) cujo valor será atribuído a partir da conversão de um valor inteiro.

DICAS: Para definir:

```
String x;  
int y = 5;
```

Para atribuir:

```
x = ""+y; // guardar em (x) uma sequência de literais vazia ( "" )  
// concatenada à conversão de valor inteiro (y) para caracteres
```

Para mostrar, usar comando para exibir apenas (x) e acrescentar outro (separado) para (y).

- 06.) Fazer um programa (Exemplo0011) capaz de guardar e mostrar uma sequência de literais (**String**) cujo valor será atribuído a partir da conversão de um valor inteiro somado a outro real.

DICAS: Para definir:

```
String x;  
int y = 5;  
double z = 0.4;
```

Para atribuir:

```
x = ""+(y+z); // guardar em (x) uma sequência de literais vazia ( "" )  
// concatenada à soma (por isso os parênteses) convertida para valor real
```

Para mostrar, usar comando para exibir apenas (x) e acrescentar outros para (y e z).

- 07.) Fazer um programa (Exemplo0012) capaz de guardar e mostrar uma sequência de literais (**String**) cujo valor será atribuído a partir da conversão de um valor lógico (ou booleano).

DICAS: Para definir:

```
String x;  
int y = 5;  
double z = 0.4;  
boolean w;
```

Para atribuir:

```
w = (y>z); // operação relacional para se avaliar uma condição  
x = ""+ w; // guardar em (x) uma sequência de literais vazia ( "" )  
// concatenada à conversão de valor lógico
```

Para mostrar, acrescentar um comando para cada valor separadamente.

08.) Fazer um programa (Exemplo0013) capaz de guardar e mostrar uma sequência de literais (**String**) cujo valor será exibido a partir da conversão de um valor lógico (ou booleano).

DICAS: Para definir:

```
String x;  
int y = 5;  
double z = 0.4;  
boolean w;
```

Sem usar atribuições, incluir a expressão em um comando para exibir o resultado:

```
x = ""+(y>z) // guardar em (x) uma sequência de literais vazia ( "" ) concatenada  
// ao resultado do teste (por isso o parênteses) entre valores inteiro e real
```

Para mostrar, acrescentar um comando para cada valor separadamente.

09.) Fazer um programa (Exemplo0014) capaz de guardar e mostrar valores inteiros (**int**) cujos valores serão lidos da entrada padrão.

DICAS: Para definir:

```
int x;  
int y;
```

Para ler um valor inteiro por vez e armazená-lo, antes de mostrá-lo:

```
x = IO.readint ( "Entrar com um valor inteiro: " );
```

Para mostrar, usar o mesmo comando do primeiro exercício:

```
IO.println ( "x=" + x );
```

E repetir os comandos para ler e exibir o valor do outro dado (y).

- 10.) Fazer um programa (Exemplo0015) capaz de guardar dois dados reais (**double**) cujos valores serão lidos da entrada padrão, somados e exibidos, bem como o resultado de sua soma.

DICAS: Para definir:

```
double x, y; // é possível definir mais de um dado do mesmo tipo
```

Para ler um valor real e armazená-lo em um dado, antes de mostrar seu valor:

```
x = IO.readdouble ( "Entrar com um valor real: " );
```

Acrescentar um comando semelhante para ler o valor de (y).

Para mostrar, usar um comando para os dois valores, juntos.

```
IO.println ( "x=" + x + " y=" + y );
```

- 11.) Fazer um programa (Exemplo0016) capaz de guardar dois dados literais (**char**) cujos valores serão lidos da entrada padrão e mostrados separadamente.

DICAS: Para definir:

```
char x, y; // é possível definir mais de um dado do mesmo tipo
```

Para ler um caractere por vez e armazená-lo em um dado, antes de mostrar seu valor:

```
x = IO.readchar ( "Entrar com um caractere: " );
```

Acrescentar um comando semelhante para ler o valor de (y).

Para mostrar, usar um comando para os dois valores, juntos.

- 12.) Fazer um programa (Exemplo0017) capaz de guardar dois dados literais (**char**) cujos valores serão lidos da entrada padrão e mostrar o resultado de sua concatenação (+) em uma única cadeia de caracteres (z).

DICAS: Para definir:

```
char x, y; // é possível definir mais de um dado do mesmo tipo
String z;  // para guardar o resultado
           // rever as dicas dos exercícios 05 (Exemplo0010) e 06 (Exemplo0011)
```

Para ler um caractere e armazená-lo em um dado,
antes de mostrar seu valor:

```
x = IO.readChar ( "Entrar com um caractere: " );
```

Para ler o valor de (y), em duas etapas, usar a forma alternativa:

```
z = IO.readString ( "Entrar com outro caractere: " ); // para ler uma palavra
```

```
y = IO.getChar ( z ); // para extrair o primeiro caractere daquele lido (z)
```

Para concatenar os valores lidos em (z), usar o modelo do exercício 06:

Para mostrar, usar um comando para o resultado com os dois valores concatenados.

- 13.) Fazer um programa (Exemplo0018) capaz de guardar dois dados literais (*String*) cujos valores serão lidos da entrada padrão e mostrar o resultado de sua concatenação (+) em uma única cadeia de caracteres (z).

DICAS: Para definir:

```
String x, y; // é possível definir mais de um dado do mesmo tipo
String z;    // para guardar o resultado da concatenação
           // rever as dicas dos exercícios 05 (Exemplo0010) e 06 (Exemplo0011)
```

Para ler uma cadeia de caracteres e armazená-los em um dado,
antes de mostrar seu valor:

```
x = IO.readString ( "Entrar com uma sequencia de caracteres: " );
```

Fazer o mesmo para (y).

Juntar (concatenar) os valores lidos e incluir um separador entre eles ("|").

Para mostrar, usar um comando para o resultado dos dois valores concatenados,
com o separador entre eles.

- 14.) Fazer um programa (Exemplo0019) capaz de guardar dois dados literais (*String*) e dois inteiros cujos valores serão lidos da entrada padrão, e mostrar como o resultado a sua concatenação (+) em uma única cadeia de caracteres (z).

DICAS: Para definir:

```
String x, y; // é possível definir mais de um dado do mesmo tipo
int    a, b;
String z;    // para guardar o resultado da concatenação de (x) com (a) e (y) com (b)
            // rever as dicas dos exercícios 05 (Exemplo0010) e 06 (Exemplo0011)
```

Para ler uma cadeia de caracteres e armazená-la:

```
x = IO.readString ( "Entrar com uma sequencia de caracteres: " );
```

Repetir para a outra variável literal (y);

Para ler um dos valores inteiros e guardá-lo em seu respectivo armazenador, usar a sugestão do exercício 09 (Exemplo0014).

Para ler e guardar um valor inteiro, experimentar a forma alternativa:

```
z = IO.readString ( "Entrar com uma sequencia de dígitos: " );
```

```
a = IO.getInt ( z ); // para tentar converter de literal para valor inteiro
```

Para armazenar uma informação nova no mesmo lugar da memória, juntando cadeia de caracteres com outro tipo de valor:

```
x = x + a;
```

Repetir para (y).

Para mostrar, usar um comando para cada valor resultante (x e y).

- 15.) Fazer um programa (Exemplo0020) capaz de guardar e mostrar dados reais (**double**) e inteiros cujos valores serão lidos da entrada padrão bem como o resultado de sua concatenação (+) em uma única cadeia de caracteres (z), antes de mostrar tudo.

DICAS: Para definir:

```
double x, y;  
int    a, b;
```

```
String z ; // para guardar o resultado da concatenação  
          // rever as dicas dos exercícios 05 (Exemplo0010) e 06 (Exemplo0011)  
          // e mostrar apenas seu valor ao final
```

Para ler um dos valores reais e guardá-lo em seu respectivo armazenador, usar a sugestão do exercício 10.

Para ler um dos valores inteiros e guardá-lo em seu respectivo armazenador, usar a sugestão do exercício 09.

Para ler e guardar o segundo valor real, experimentar a forma alternativa:

```
z = IO.readString ( "Entrar com uma sequencia de dígitos: " );
```

```
x = IO.getDouble ( z ); // para tentar converter de literal para valor real
```

Para ler e guardar o segundo valor inteiro, experimentar a forma alternativa, como descrito no exercício 14.

Para armazenar informações em um mesmo lugar da memória, juntando cadeia de caracteres com outro tipo de valor:

```
z = "(" + x + ", " + a + " )";
```

Mostrar continuar juntando outras cadeias de caracteres:

```
z = z+ "(" +y + ", " + b+ " )";
```

Mostrar o valor final (z), com a concatenação do par (x,a) e do par (y,b) em um único valor.

Tarefas extras

E1.) Fazer um programa capaz de guardar e mostrar um valor literal (**char**) e seu código inteiro para representação interna.

DICAS: Copiar o modelo do Exemplo0006, e incluir a modificação sugerida abaixo.

Acrescentar à definição anterior:

```
char x;  
int y;
```

Para obter o código inteiro para representação interna,
usar a conformação de tipo (**type casting**) para inteiro:

```
y = (int) x;
```

Sugestão: Testar letras maiúsculas, minúsculas e algarismos lidos de teclado.

E2.) Fazer um programa capaz de guardar e mostrar um valor inteiro (**int**) no intervalo [32:126] e seu símbolo correspondente.

DICAS: Copiar o modelo do Exemplo0006, e incluir a modificação sugerida abaixo.

Acrescentar à definição anterior:

```
char x;  
int y;
```

Para obter o símbolo correspondente ao valor inteiro,
usar a conformação de tipo (**type casting**) para caractere:

```
x = (char) y;
```

Sugestão: Testar valores inteiros acima de 127.