FACULDADE QI BRASIL

CURSO: Análise e Desenvolvimento de Sistemas
UNIDADE CURRICULAR: Linguagem de Programação 1

**Tópico 03**
Revisão de algoritmos e programação.
Java básico.

BIBLIOGRAFIA:
Você pode encontrar mais conteúdo sobre o tema deste tópico no livro da Biblioteca Virtual QI:
DEITEL, Paul J.; DEITEL, Harvey M. **Java - como programar.** 8. ed. São Paulo: Pearson Prentice Hall, 2010. pp.189-221.
Sobre a classe Math, vista neste tópico:
Mesmo livro, pp.156-.
Sobre a classe String, vista neste tópico:
Mesmo livro, pp.517-.

# Tópico 03

Nesta aula será feita novamente uma REVISÃO sobre alguns conceitos básicos de programação, onde serão revistos **ARRAYS** (vetores e matrizes). Serão vistos também alguns métodos da classe **Math** e da classe **String** do Java.

## REVISÃO DE CONCEITOS BÁSICOS

### Arrays

Um **array** (em português: arranjo, vetor) é um tipo especial de objeto que contém zero ou mais valores primitivos ou referências. É como uma variável que pode armazenar mais de um valor (um conjunto de valores). Esses valores são mantidos nos elementos do *array*, que são variáveis não identificadas às quais fazemos referência pela sua posição ou índice. O tipo de um *array* é definido pelo seu elemento, e todos os elementos do *array* devem ser desse tipo.

Os elementos do *array* são numerados iniciando com zero, e índices válidos variam de zero ao número de elementos menos um. O elemento do *array* com índice 1, por exemplo, é o segundo elemento no *array*. O número de elementos em um *array* é seu comprimento. É especificado quando o *array* é criado e nunca muda seu tamanho.

### Criando e inicializando arrays

Para criar um *array* em Java, você utiliza a palavra-chave **new**, assim como faz para criar um novo objeto. Os tipos *array* não possuem construtores, mas é exigido que você especifique um tamanho sempre que criar um *array*. O tamanho desejado do *array* deve ser especificado dentro de colchetes.

```
int[] jogoMegaSena; //Declaração
jogoMegaSena = new int[6]; //Criação
jogoMegaSena[0] = 23; //Inicialização da posição 0
jogoMegaSena[1] = 12; //Inicialização da posição 1
jogoMegaSena[2] = 55; //Inicialização da posição 2
jogoMegaSena[3] = 02; //Inicialização da posição 3
jogoMegaSena[4] = 07; //Inicialização da posição 4
jogoMegaSena[5] = 19; //Inicialização da posição 5
```

**IMPORTANTE:** O OPERADOR **new** ALOCA (RESERVA/CRIA NA MEMÓRIA RAM DO COMPUTADOR) UM LOCAL ONDE SERÁ ARMAZENADO O VETOR. ISTO É FEITO DINAMICAMENTE, OU SEJA, EM TEMPO DE EXECUÇÃO DO PROGRAMA (QUANDO ELE ESTÁ RODANDO).

Outra forma de inicialização de um *array*, mais enxuta:

```
int[] outroJogoSena = new int[]{23, 12, 55, 02, 07, 19};
```

No exemplo acima, acontecem quatro coisas em apenas uma linha:

1. Declaração de uma referência a um *array* de inteiros chamado outroJogoSena;
2. Criação de um *array* com seis posições;
3. Inicialização das posições com os valores 23, 12, 55, 02, 07 e 19;
4. Atribuição do novo objeto (*array*) a outroJogoSena como uma referência.

O outro atalho que a linguagem Java nos permite é o seguinte:

```
int[] outrojogoMegaSena = {23, 12, 55, 02, 07, 19};
```

**Acessando elementos do array**

```
String[] respostas = new String[2]; //Cria um array de dois Strings
respostas[0] = "Sim"; //configura o primeiro elemento do array
respostas[1] = "Não"; //configura o segundo elemento do array

//Agora lê esses elementos do array

System.out.println(questão + " ("
                   + resposta[0] + "/" + resposta[1] + "): ");
```

**Limites de array**

Lembre-se de que o primeiro elemento de um *array* a é a[0]. O segundo é a[1] e o último é a[a.length-1].

**Iterando por arrays**

É comum escrever laços que iteram (percorrem) por cada um dos elementos de um *array*, afim de realizar uma operação sobre ele:

```
int[] primos = {2,3,5,7,11,13,17,19};
int somaPrimos = 0;
for(int i = 0; i < primos.length; i++)
    somaPrimos += primos[i];
```

Outra forma de fazer iteração de *Arrays* é utilizando o **for-each** ("para-cada"). Esta forma de percorrer o vetor é mais rápida e torna o programa mais claro:

```
int[] vetor = {12,23,45,76,98};
for (int x: vetor) { //Para cada int x de vetor faça...
    System.out.println(x);
}
```

**Copiando arrays**

Todos os tipos *array* implementam a interface **Cloneable**, e qualquer *array* pode ser copiado invocando seu método clone(). É necessário fazer a coerção ("casting"):

```
int[] dados = {1, 2, 4};
int[] copia = (int[]) dados.clone();
```

**Utilitários de arrays**

A classe **java.util.Arrays** contém alguns métodos utilitários estáticos para trabalhar com *arrays*, por exemplo:

- sort() - classificar *arrays*.
- binarySearch() – busca em *arrays*.
- equals() – comparar *arrays*.
- Arrays.toString() – converte o conteúdo do *Array* em um string.

**Arrays multidimensionais**

*Arrays* unidimensionais (uma só sequência de dados) são estruturas de dados bastante simples. Uma estrutura um pouco mais complexa são os *arrays* multi-dimensionais ou n-dimensionais. A função destes *arrays* é a mesma dos seus irmãos unidimensionais, porém *arrays* multi-dimensionais permitem a construção de estruturas de dados mais ricas (por exemplo, matrizes, tabelas). No caso de *arrays* multi-dimensionais a declaração, construção e inicialização é realizada conforme exemplos a seguir. Observe que os trechos de códigos são substituíveis entre si, isto é, equivalentes:

```
String[][] jogoDaVelha;

jogoDaVelha = new String[3][3];
```

```
String[][] jogoDaVelha;

jogoDaVelha = new String[3][];
jogoDaVelha[0] = new String[3];
jogoDaVelha[1] = new String[3];
jogoDaVelha[2] = new String[3];
```

```
String[][] jogoDaVelha;

jogoDaVelha = new String[][]{{null, null, null}, {null, null, null},
                             {null, null, null}};
```

**FUNÇÕES MATEMÁTICAS**

**Classe Math**

```
Math.<nome da função>(<argumentos ou lista de argumentos>)
```

Duas constantes matemáticas:

- Math.PI
- Math.E

Algumas funções

FUNÇÃO **ceil()**

Arredonda um número do tipo double para seu próximo inteiro ("para cima" - ceil vem do inglês *ceiling* = teto).

```
public class ExemploCeil {
    public static void main(String[] args) {
        double A = 5.2, B = 5.6, C = -5.8;
        System.out.println("Arredondando 5.2: " + Math.ceil(A));
        System.out.println("Arredondando 5.6: " + Math.ceil(B));
        System.out.println("Arredondando -5.8: " + Math.ceil(C));
    }
}
```

FUNÇÃO **floor()**

Arredonta um número do tipo double para seu inteiro anterior ("para baixo").

```
public class ExemploFloor {
    public static void main(String[] args) {
        double A = 5.2, B = 5.6, C = -5.8;
        System.out.println("Arredondando 5.2: " + Math.floor(A));
        System.out.println("Arredondando 5.6: " + Math.floor(B));
        System.out.println("Arredondando -5.8: " + Math.floor(C));
    }
}
```

FUNÇÃO **max()**

Verifica o maior valor entre dois números.

```
public class ExemploMax {
   public static void main(String[] args) {
      int A = 10, B = 15;
      double C = -5.9, D = -4.5;
      System.out.println("O maior valor entre 10 e 15 é: " +Math.max(A,B));
      System.out.println("O maior valor entre -5.9 e -4.5 é: "
                         + Math.max(C,D));
      System.out.println("O maior valor entre 10 e -5.9 é: "
                         + Math.max(A,C));
   }
}
```

FUNÇÃO **min()**

Verifica o menor valor entre dois números.

```
public class ExemploMin {
   public static void main(String[] args) {
      int A = 10, B = 15;
      double C = -5.9, D = -4.5;
      System.out.println("O menor valor entre 10 e 15 é: " +Math.min(A,B));
      System.out.println("O menor valor entre -5.9 e -4.5 é: "
                         + Math.min(C,D));
      System.out.println("O menor valor entre 10 e -5.9 é: "
                         + Math.min(A,C));
   }
}
```

FUNÇÃO **sqrt()**

Calcula a raiz quadrada. O parâmetro é do tipo double.

```
public class ExemploSqrt {
   public static void main(String[] args) {
      double A = 900, B = 30.25;
      System.out.println("A raiz quadrada de 900 é: " + Math.sqrt(A));
      System.out.println("A raiz quadrada de 30.25 é: " + Math.sqrt(B));
   }
}
```

FUNÇÃO **pow()**

Calcula a potência de um número.

```
public class ExemploPow {
   public static void main(String[] args) {
      double base = 5.5, poten = 2;
      System.out.println("5.5 elevado a 2 é: " + Math.pow(base,poten));
      System.out.println("25 elevado a 0.5 é: " + Math.pow(25,.5));
      System.out.println("5678 elevado a 0 é: " + Math.pow(5678,0));
   }
}
```

FUNÇÃO **random()**

Gera valores de forma aleatória. Gera um valor entre 0.0 e 1.0. O valor 1.0 nunca é gerado.

```
(int) (Math.random() * 100) //Gera valores entre 0 e 99
```

Exemplo:

```
public class ExemploRandom {
   public static void main(String[] args){
      for(int qtd = 1; qtd <= 5; qtd++){
         for(int x = 1; x <= 6; x++){
            int num = (int) (Math.random() * 80);
            System.out.println(num + " ");
         }
         System.out.println();
      }
   }
}
```

**MÉTODOS DA CLASSE STRING**

**String (ou cadeia de caracteres)**

Todos os tipos utilizados na linguagem Java, com exceção dos tipos primitivos (int, long, float, double, char e boolean), são **"objetos"**. O tipo **String**, com S maiúsculo, é um dos objetos mais utilizados.

Ao contrário que ocorre em C e C++, "strings" em Java não são tratadas como sequências de caracteres terminadas por NULL. São objetos ou instâncias da classe java.lang.String, e portanto devem ser declarados e instanciados. Por exemplo:

```
// declaração
String ola;

// instanciação
ola = new String("Alô Mundo Java !");

// declaração e instanciação (mais simples)
String ola = "Alô Mundo Java !";
String nome = "Prof. Luiz Duarte";

// concatenação (\n = próxima linha)
String aula = ola + "\nby " + nome;
System.out.println(aula);
```

Resultado:

```
Alô Mundo Java !
by Prof. Luiz Duarte
```

Para obter uma String digitada pelo usuário deve-se utilizar o método "nextLine" da classe **Scanner**. Já para efetuar a leitura de um caractere, delimitado entre aspas simples, deve-se utilizar o método "read" do pacote de classes **System.in** como pode ser observado na aplicação Java transcrita abaixo:

```
import java.io.IOException;
import java.util.Scanner;

public class String3 {
// Através da cláusula throws indicamos que não iremos
// tratar possíveis erros na entrada de dados realizada
// através do método "read" do pacote de classes System.in
    public static void main(String[] args) throws IOException {
        Scanner ler = new Scanner(System.in);

        String nome;
        char sexo;
```

```
        System.out.printf("Informe um nome:\n");
        nome = ler.nextLine();

        System.out.printf("\nInforme o sexo (M/F):\n");
        sexo = (char)System.in.read();

        if((sexo == 'M') || (sexo == 'm'))
            System.out.printf("\nSeja bem-vindo Sr. %s.\n", nome);
        else System.out.printf("\nSeja bem-vinda Sra. %s.\n", nome);
    }
}
```

Metódos da classe String

Método **lenght()** (número de caracteres)
Método **charAt()** (caractere da posição informada)
Métodos **toUpperCase()** e **toLowerCase()** (passar para maiúsculas ou minúsculas)
Método **substring()** (copia um pedaço do string)
Método **trim()** (remove espaços no início e no fim do string)
Método **replace()** (substitui caracteres por outros)
Método **String.valueOf()** (transforma números em string)

**Exercícios de fixação do conteúdo**

1) Implemente a lógica para realização de saques em um caixa eletrônico, considerando que o mesmo armazena cédulas de R$100,00, R$50,00, R$20,00, R$10,00, R$5,00 e R$2,00 e devem ser entregues ao cliente o menor número possível de cédulas.
2) Escreva um programa em Java para ler um vetor X de 10 elementos inteiros. Logo após copie os elementos do vetor X para um vetor Y fazendo com que o 1º elemento de X seja copiado para o 10º de Y, o 2º de X para o 9º o de Y e assim sucessivamente. Após o término da cópia, imprimir o vetor Y.
3) Escreva um programa em Java para ler um vetor A de 10 elementos inteiros e um valor X. A seguir imprimir os índices do vetor A em que aparece um valor igual a X.
4) Escreva um programa em Java para ler um vetor A de 10 elementos inteiros e um valor X. A seguir imprimir "ACHEI" se o valor X existir em A e "NÃO ACHEI" caso contrário.
5) Escreva um programa em Java para ler um vetor A de 10 elementos e um valor X. Copie para um vetor S (sem deixar elementos vazios entre os valores copiados) os elementos de A que são maiores que X. Logo após imprimir o vetor S.
6) Escreva um programa em Java para ler a quantidade de elementos que serão armazenados em um vetor A. Crie um vetor com o tamanho apropriado. Depois o programa deve solicitar os números conforme a quantidade informada pelo usuário, e armazená-los no vetor A.