FeMASS
FACULDADE PROF. MIGUEL ÂNGELO DA SILVA SANTOS

Projeto de Sistemas I

Faculdade Prof. Miguel Ângelo da Silva Santos

Material 2 – Manipulações de string, numéricas e arrays

Professor: Isac Mendes Lacerda
e-mail: isac.curso@gmail.com

Tópicos

- Manipulações numéricas e de string
- Arrays (vetores e matrizes)

Operações matemáticas
(métodos)

- A classe **Math** possui vários métodos especializados em cálculos.
- Como os métodos de **Math** são estáticos, a forma de chamá-los é:

```
Math.<nome do método>(lista de argumentos)
```

Operações matemáticas
(métodos)

- Dois métodos usados para arredondamento são **ceil** (para o próximo inteiro) e **floor** (para o inteiro anterior). Duas maneiras de utilizar:

```
JOptionPane.showMessageDialog(null, Math.ceil(5.2)); // 6

System.out.println(Math.floor(5.2)); // 5
```

- Já o método **round** arredonda para baixo valores < 0.5 e para cima valores >= 0.5. Veja como usar:

```
System.out.println(Math.round(5.5)); // 6
```

Operações matemáticas
(métodos)

- Dois métodos usados para identificar valores máximos e mínimos em uma série são respectivamente: **max** e **min**. Como utilizar:

```
Math.max(<valor1>, <valor2>);

Math.min(<valor1>, <valor2>);
```

Operações matemáticas
(métodos)

- Dois métodos usados para potenciação e raiz quadrada são respectivamente: **sqrt** e **pow**. Como utilizar:

```
Math.sqrt(<double>);

Math.pow(<base>, <potência>);
```

Operações matemáticas
(métodos)

- O método **random** é utilizado para gerar valores (tipo double: 16 casas decimais) aleatórios entre 0 e 1 (onde o valor 1 nunca é gerado). Como utilizar:

```
Math.random(); //gera um double entre 0.0 a 0.9
```

0.0000000000000000	0.0000000000000001	...	0.9999999999999999

Qual é o intervalo de números para sorteio?

Operações matemáticas
(métodos)

- O método **random** é utilizado para gerar valores (tipo double: 16 casas decimais) aleatórios entre 0 e 1 (onde o valor 1 nunca é gerado). Como utilizar:

```
Math.random(); //gera um double entre 0.0 a 0.9
```

0.0000000000000000	0.0000000000000001	...	0.9999999999999999

Qual é o intervalo de números para sorteio?

100 quatrilhões

Operações matemáticas
(métodos)

- O método **random** é utilizado para gerar valores (tipo double: 16 casas decimais) aleatórios entre 0 e 1 (onde o valor 1 nunca é gerado). Como utilizar:

```
Math.random(); //gera um double entre 0.0 a 0.9

Math.random()*100; //gera um double * 100

(int)(Math.random()*100); //trunca as casas decimais
```

Operações matemáticas
(formatação)

```
import java.text.DecimalFormat;
public class Exercício_parte1 {
    public static void numericos(){
        DecimalFormat df = new DecimalFormat();
        //define qtd dígitos antes e depois da vírgula
        df.applyPattern("00.00");
        System.out.println(df.format(0.9700));
        //mostra nr, separando milhares de centenas
        df.applyPattern("###,###.00");
        System.out.println(df.format(1500));
        //mesma coisa do anterior, mas só casas decimais quando existem
        df.applyPattern("###,###.##");
        System.out.println(df.format(1500));

    }
}
```

Operações com Strings

- **String** é uma classe com vários métodos disponíveis e a forma de chamá-los será sempre a partir do objeto String existente ou da própria classe String, como ilustra a forma abaixo:

```
<Nome da string>.<nome do método>(<argumentos>);
String.<nome do método>(<argumentos>);
```

Operações com Strings (métodos)

- Entre os principais métodos estão:

Como usar método	O que faz?
<Nome string>.**length**( )	Número de caracteres
<Nome string>.**charAt**(<índice>)	Indica o caractere do índice
<Nome string>.**toUpperCase**( ) ou <Nome string>.**toLowerCase**( )	Transforma todos os caracteres em maiúsculo ou minúsculo
<Nome string>.**substring**(<início>, <fim>)	Retorna uma cópia da uma string a partir dos índices de início e fim
<Nome string>.**trim**( )	Elimina espaços no início e fim
<Nome string>.**replace**(<velho>, <novo>)	Troca conjuntos de caracteres
String.**valueOf**(<variável>)	Converte para string
<Nome string>.**indexOf**(<substring, <índice de início da busca>)	Indica o índice da primeira ocorrência

Exercícios 1

1. No método principal, crie a lógica que simule a jogada de um dado de seis lados (números de 1 a 6), pelo número de vezes informado pelo usuário. Ao final, some seus valores sorteados e apresente o resultado das jogadas.

Exercícios 2

1. Uma farmácia precisa ajustar os preços de seus produtos em 12%. No método principal crie a lógica que receba o valor do produto e aplique o percentual de acréscimo. O novo valor a ser calculado deve ser arredondado usando o método round. O método deve também conter um laço de repetição que encerre o programa quando o usuário fornecer o valor zero (0) para o valor do produto. Dessa forma, o usuário digita o valor do produto, o método calcula e mostra o valor com acréscimo, a seguir solicita um novo valor. Esse processo continua enquanto o valor do produto for diferente de zero; caso contrário o programa será encerrado.

Exercícios 3

- No método principal gere um número aleatoriamente (entre 5 e 10) por Math.random. Em seguida, faça com que apareça em tela uma senha numérica contendo a mesma quantidade de dígitos correspondentes ao valor aleatório gerado. Apresente em tela o número sorteado e a senha .

Exercícios 4

- No método principal, receba uma frase qualquer e mostre essa frase de trás para a frente e sem espaços em branco.

Exercícios 5

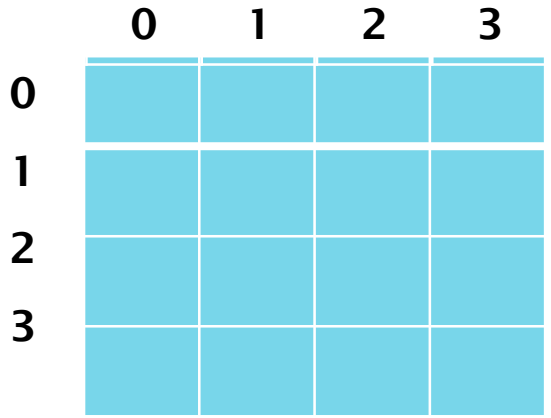
- No método principal receba uma frase e verifique se essa frase possui palavras impróprias. As palavras impróprias são: sexo e sexual. Caso encontre uma dessas palavras, emita em tela a mensagem “conteúdo impróprio”, caso contrário “conteúdo liberado”.

Arrays (unidimensional e multidimensional)

- Arrays são estruturas de dados unidimensionais ou multidimensionais, manipuláveis por índices.

Array unidimensional (vetor)

Array bidimensional (matriz)



Arrays (vetores e matrizes)

- Como declarar um array unidimensional:

```
Tipo-de-dado[] nome = new Tipo-de-dado[qtd];

int[] nrs = new int[5];

for (i=0; i<5; i++){
    nrs[i] = (int)(Math.random()*100);
}

String[] nomes = {“José”, “Maria”, “Pedro”};
```

Arrays (vetores e matrizes)

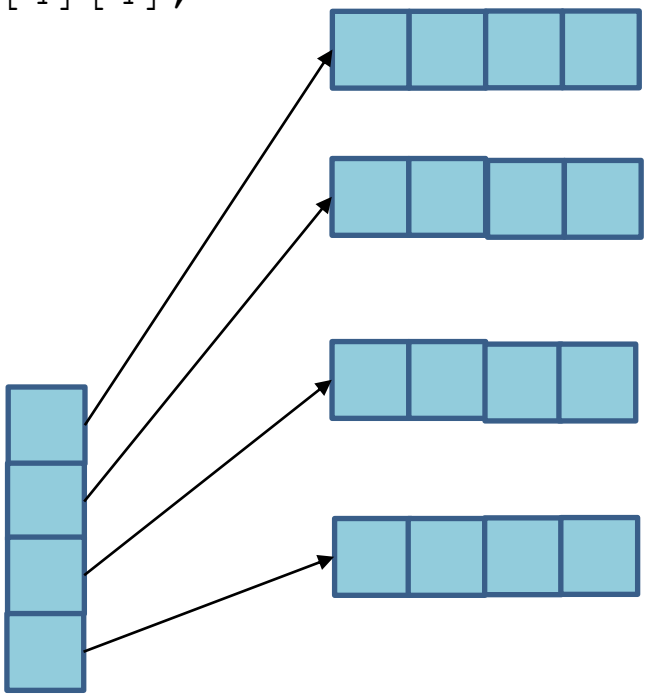
- Como declarar um array bidimensional:

```
Tipo-de-dado[][] nome = new Tipo-de-dado[qtd][qtd];

int[][] matriz = new int[2][4];

for (linha=0; linha<2; linha++){
    for (coluna=0; coluna<4; coluna++){
          matriz[linha][coluna] = (int)(Math.random()*100);
    }
}
```

Arrays (vetores e matrizes)

Arrays bidimensionais são arrays dentro de array.
A figura equivale à seguinte linha de código:

```
int[][] matriz = new int[4][4];
```



Arrays (limitações)

- Em java, esse tipo de array, tem **estrutura estática**. Isso significa que precisamos definir as dimensões antes de usar.

- Mas e se precisarmos redimensionar?

  - Podemos **criar uma nova estrutura** (com novo tamanho de interesse) e **copiar os elementos** da estrutura antiga para a nova.

  - Podemos **mudar** a tipo de estrutura **para *Collections***, que tem tipo dinâmico.

Exercício 6

- No método *main* gere aleatoriamente mil números inteiros, entre 50 e 1050, e os armazene num *array* bidimensional (com o formato de 50 linhas e 20 colunas). A seguir, peça para o usuário digitar um número entre 50 e 1050 para ser pesquisado no *array*. Informe ao usuário se o número existe ou não no *array*. Se existir, pode ser que o número tenha sido gerado em mais de uma posição na matriz. Neste caso, informe em quais posições (índice da linha e índice da coluna) o número informado pelo usuário foi encontrado.

ArrayList (unidimensional e multidimensional)

- **ArrayList** gera objetos com estruturas unidimensionais ou multidimensionais, que fazem parte do que é chamado de ***Collections Framework***.

- Além disso, **ArrayList** trabalha com **estrutura dinâmica** que gera uma série de vantagens em relação a estrutura estática de **Array** comum. Por exemplo:

    - Saber o número de posições preenchidas;
    - Redimensionamento automático;
    - Buscar por um elemento, cujo o índice não se sabe.

ArrayList (unidimensional)

- Como declarar um arrayList unidimensional:

```
ArrayList<Classe> nome = new ArrayList<>();

ArrayList<Double> peso = new ArrayList<>();

for (i=0; i<50; i++){
    peso.add(Math.random()*100);
}
```

Exercício 7

Faça um programa que leia do teclado 20 números inteiros e armazene-os num ArrayList. Armazene os números pares em um ArrayList PAR e os ímpares em um ArrayList ÍMPAR. Ordene os vetores. Imprima os três vetores.

Exercício 8

Escreva um programa que leia um número inteiro não negativo *n*, em seguida leia tais *n* números e calcule a frequência de ocorrência de cada um deles. Por exemplo:

n = 5

6	7	6	88	3

6: 2 vezes
7: 1 vez
88: 1 vez
3: 1 vez

ArrayList (bidimensional)

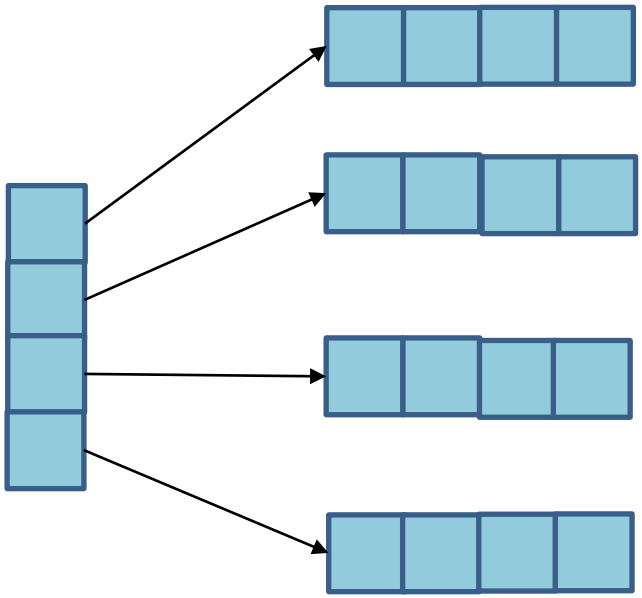
- Como declarar um arrayList bidimensional:

```
ArrayList<ArrayList<Classe>> nome = new ArrayList<>();

ArrayList<ArrayList<Integer>> mtx = new ArrayList<>();
ArrayList<Integer> linha_1 = new ArrayList<>();
mtx.add(linha_1);

for (int i=0; i < 50; i++){
    linha_1.add(Math.random()*100);
}

for (ArrayList<Integer> linha : mtx){
    for (int item : linha){
          System.out.print(item + "\t");
    }
}
```

ArrayList (bidimensional)

ArrayList bidimensional é um objeto com objetos internos. A linha de código abaixo cria uma estrutura vazia que pode incluir objetos ArraysList com valores inteiros:

```
ArrayList<ArrayList<Integer>> matriz = new ArrayList<>();
```



Exercício 9 (exercício 6 mas com ArrayList)

- No método *main* gere aleatoriamente mil números inteiros, entre 50 e 1050, e os armazene num *ArrayList* bidimensional (com o formato de 50 linhas e 20 colunas). A seguir, peça para o usuário digitar um número entre 50 e 1050 para ser pesquisado no *ArrayList*. Informe ao usuário se o número existe ou não no *ArrayList*. Se existir, pode ser que o número tenha sido gerado em mais de uma posição na matriz. Neste caso, informe em quais posições (índice da linha e índice da coluna) o número informado pelo usuário foi encontrado.

Exercício 10

- Faça um programa que solicite ao usuário números e os armazene em uma matriz quadrada. Em seguida, crie um *ArrayList* que armazene os elementos da diagonal principal da matriz e imprima esses elementos. Para ler a matriz considere que o usuário irá digitar cada linha de uma vez e os elementos da linha são separados por " ; " Observe o exemplo abaixo:

```
digite a linha da matriz ou 0 para sair:10;2;3;4
digite a linha da matriz ou 0 para sair:3;20;1;2
digite a linha da matriz ou 0 para sair:5;2;7;8
digite a linha da matriz ou 0 para sair:2;8;9;10
digite a linha da matriz ou 0 para sair:0
```