Curso de Extensão

Programação de Aplicações Comerciais em JAVA

Aula 2

A Linguagem JAVA

("Episódio IV: Uma nova Esperança")

Prof. Monael

Estrutura de um Programa

- Todo programa JAVA é uma classe;
- Toda classe executável deve ter um médoto public static void main()
- O programa executável mais simples em JAVA seria:

```
public class Primeiro
{
    public static void main(String arguments[])
    {                }
}
```

Variáveis

- Variável é um lugar onde as informações podem ser armazenadas, e esse valor pode ser alterado em qualquer ponto do programa.
- A sintaxe para criar uma variável é a seguinte:

  <tipo> <nome> [= valor];

Variáveis

- Nomeando as variáveis:
  - Devem começar por letras, ou (_) ou ($)
  - Não podem começar por números
  - Java é "case sensitive"
    - Nomes válidos:
      - nome, _endereco, $valor, nome2, $valor31, endereco3
    - Nomes não válidos:
      - 1nome, class, super, new, 1234
    - Case sensitive:
      - NomeAluno, nomealuno, Nomealuno, nomeAluno

Variáveis

- Nomeando as variáveis:
  - Palavras reservadas:

abstract	do	implements	private	this
boolean	double	import	protected	throw
breaak	else	instanceof	public	throws
byte	extends	int	return	transient
case	false	interface	short	int
catch	final	long	static	try
char	finally	native	super	void
class	float	new	switch	volatile
continue	for	null	synchronized	while
default	if	package		

Variáveis

- Tipos básicos em JAVA:
  - Tipo Lógico:

    sintaxe: boolean <nome> [= valor];

    valores: true ou false

    exemplos:

    ```
    boolean luz;
    boolean resposta = true;
    ```

Variáveis

- Tipos básicos em JAVA:
  - Tipo Caractere:

    sintaxe: char <nome> [= valor];

    valores: qualquer caracter Unicode – (2 Bytes)

    exemplos:

```
char letra;
char vogal = 'a';
char pula_linha = '\n';
```

Variáveis

- Tipos básicos em JAVA:
  - Tipo Numéricos:

    sintaxe: byte <nome> [= valor];

    valores: de -128 a 127 – (1 Byte)

    exemplos:

```
byte numero;
byte contagem = 56;
```

Variáveis

- Tipos básicos em JAVA:
  - Tipo Númeral:

    sintaxe: short <nome> [= valor];

    valores: de -32.768 a 32.767 – (2 Bytes)

    exemplos:

```
short cont;
short soma = 346;
```

Variáveis

- Tipos básicos em JAVA:
  - Tipo Númeral:

    sintaxe: int <nome> [= valor];

    valores: de -2.147.483.648 a 2.147.483.647 – (4 Bytes)

    exemplos:

```
int numero;
int quantidade = 3456789;
```

Variáveis

- Tipos básicos em JAVA:
  - Tipo Númeral:

    sintaxe: long <nome> [= valor];

    valores: de -2^{63} a $2^{63}-1$ → (8 Bytes)

    exemplos:

    long number;

    long val = 987L;

Variáveis

- Tipos básicos em JAVA:
  - Tipo Númeral:

    sintaxe: float <nome> [= valor];

    valores: de 1.4^{-45} a 3.4^{38}

    exemplos:

    ```
    float n;
    float preco = 1.99f;
    ```

Variáveis

- Tipos básicos em JAVA:
  - Tipo Númeral:

    sintaxe: double <nome> [= valor];

    valores: de 4.9^{-324} a 1.7^{308}

    exemplos:

    ```
    double numero_grande;
    double conta_do_maluf = 984329094345;
    ```

Variáveis

- Em JAVA as variáveis podem ser:
  - De Instância
  - De Classe
  - Locais

Variáveis

- Exemplo de variável de instância:

```
public class exemplo
{
        int         valor;
        float       correção;
        public static void main(String arguments[])
        {           }
}
```

Variáveis

- Exemplo de variável de classe:

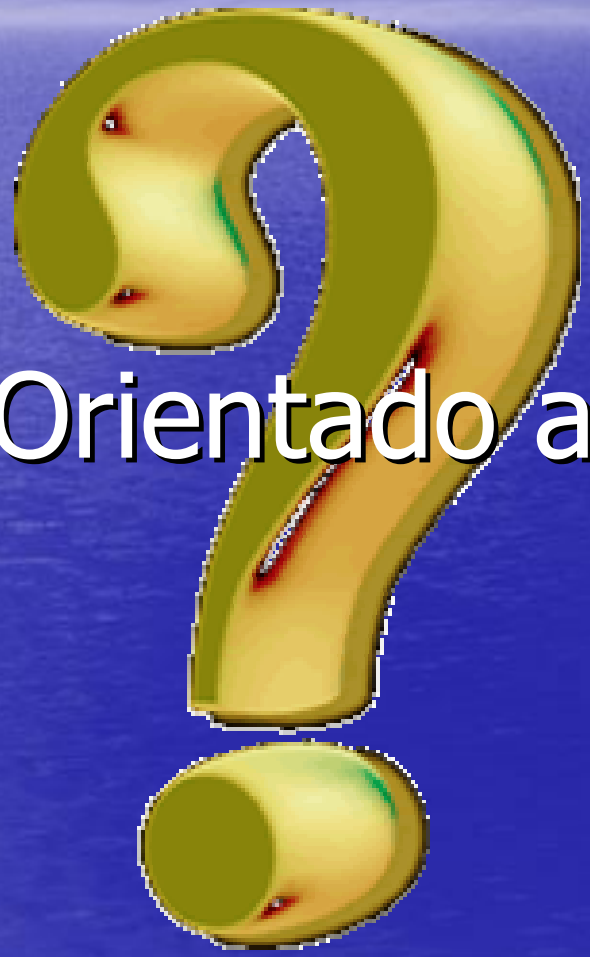
```
public class exemplo
{
        static    int       numero = 7;
        static    char      letra = 'g';
        public static void main(String arguments[])
        {         }
}
```

Variáveis

- Exemplo de variável de classe:

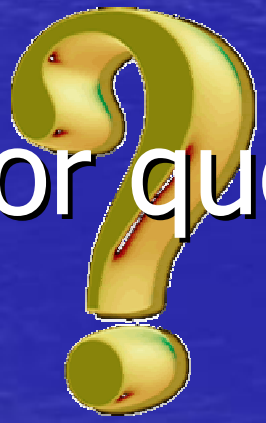
```
public class exemplo
{
        public static void main(String arguments[])
        {
                double salario;
        }
}
```

JAVA é Orientado a Objetos ?



JAVA não é totalmente Orientado a Objetos.

Por que ?



Variáveis

- Tipo Classe
  - Podem haver variáveis do tipo classe.
    - O que são essas variáveis?

      Sintaxe: <nome da classe> <nome>;

      Exemplo:

```
String nome = "Aristarcos";
Color    cabelos;
HashTable tabela;
```

Variáveis

- Constantes
  - É uma variável que o valor nunca muda...

  Sintaxe: final <tipo> <nome> = <valor>;

  Exemplo:

```
final float     PI = 3.141618f;
final int       ICMS = 0.18f;
final char      NEWLINE = '\n';
```

Comentários

- Informações incluídas no programa estritamente para beneficio das pessoas entenderem o que está acontecendo no programa.
  - Tipos de comentários em JAVA:

// esse é o comentário de uma única linha

/* esse comentário pode ocupar várias linhas no texo, desde que seja aberto e fechado adequadamente */

/** esse é um comentário especial, o qual é interpretado como sendo documentação oficial sobre a classe, métodos e atributos. Há uma ferramenta do SDK chamada JAVADOC que a captura e transforma em um documento HTML padrão da documentação do JAVA */

Expressões e Operadores

- Expressões são instruções que geram valores, as mais comuns são as expressões matemáticas.
  - Exemplo:

```
int x = 3;
int y = x;
int z = x * y;
```

Expressões e Operadores

- Operadores são símbolos especiais usados para funções matemáticas, lógicas, de comparação e de atribuição:
  - Operadores Matemáticos:

+	adição	3 + 4
-	subtração	5 – 7
*	multiplicação	5 * 5
/	divisão	14 / 7
%	Módulo	20 % 7

Expressões e Operadores

– Operadores de Atribuição:

x += y	x = x + y
x -= y	x = x – y
x *= y	x = x * y
x /= y	x = x / y
x %= y	x = x % y

- Incrementando e decrementando

x++	Atribui e incrementa 1
++x	Incrementa 1 e Atribui
x--	Atribui e decrementa 1
--x	Decrementa 1 e atribui

Expressões e Operadores

- Operadores de comparação:

==	Igual	2 == 2	true
!=	Diferente	3 != 3	true
<	Menor	4 < 1	false
>	Maior	6 > 5	true
<=	Menor ou igual	0 <= 7	true
>=	Maior ou igual	7 >= 7	true

Expressões e Operadores

- Operadores de comparação:

& ou &&	E lógico
| ou ||	OU lógico
^	OU exclusivo
!	Negação

Expressões e Operadores

- Aritmética de Strings
  - Concatenação é a ligação de duas coisas.

```
String sJan = "de Janeiro";
String s = "Sul";
String sRio = "Rio ";
String sEstado = sRio + sJan;

System.out.println(sEstado);

sEstado = sRio + " Grande do " + s;

System.out.println(sEstado);
```

Laboratório

Como sabemos existe várias unidades de medição de temperatura, onde as 3 principais escalas são: Celsius (°C), Fahrenheit (°F) e Kelvin (K).
Conforme a tabela abaixo:

De °C para	Fórmula
°F	°C × 1.8 + 32
K	°C + 273.15

Desenvolva um programa em Java que apresente quanto equivale 41°C nas escalas °F e K.

Laboratório

```
public class Conversor
{
	public static void main(String arguments[])
	{
		int temperatura = 41;

		float tempF = temperatura * 1.8f + 32;
		float tempK = temperatura + 273.15f;

		System.out.println("41 graus Celsius equivale a:");
		System.out.println(tempF + " graus Fahrenheit.");
		System.out.println(tempK + " graus Kelvin.");
	}
}
```

Trabalhando com Objetos

- O operador new
  - É usado para criar um novo objeto.

  Sintaxe:

  <nome da classe> <nome do objeto> = new <nome da classe> ([argumentos]);

  Exemplos:

```
String nome = new String();
Conta c = new Conta();

String nome = new String("Fabíola");
Conta c = new Conta(3284);
```

Trabalhando com Objetos

- Definindo os atributos e métodos de uma classe:

Sintaxe:

```
public class Classe
{
        <tipo> <nome do atributo> = <[valor]>;
        ...
        <tipo de retorno> <nome do método>(<[lista de argumentos]>)
        {
                ...
        }
        ...
}
```

Trabalhando com Objetos

- Definindo os atributos e métodos estáticos de uma classe
  - Definindo variáveis de classe:

Sintaxe:

```
public class Classe
{
        static <tipo> <nome do atributo> = <[valor]>;
        ...
        static <tipo de retorno> <nome do método>(<[lista de argumentos]>)
        {
                ...
        }

}
```

Trabalhando com Objetos

- Acessando atributos de um objeto
  - Para se acessar as variáveis de um objeto, usa-se o operador (.) ponto.

Sintaxe:

<nome do objeto>.<nome do atributo>;

Exemplos:

cta.saldo;

cta.nome;

livro.autor;
livro.paginas;

Trabalhando com Objetos

- Alterando valores da variáveis de um objeto

  Sintaxe:

  <nome do objeto>.<nome do atributo> = <novo valor>;

  Exemplos:

  cta.saldo = 1500f;

  cta.nome = "Anita";

  livro.autor = "Machado de Assis";

  livro.paginas = 265;

Trabalhando com Objetos

- Acessando atributos estáticos de um objeto
  - Para se acessar as variáveis estáticas de um objeto, usa-se o operador (.) ponto. Entretanto, usa-se o nome da classe diretamente.

Sintaxe:

<nome da classe>.<nome do atributo>;

Exemplos:

Math.PI;

Math.E;

JOptionPane.YES_NO_OPTION;

Curso de Extensão

Programação de Aplicações Comerciais em JAVA

Aula 3

A Linguagem JAVA

("Episódio V: O Império Contra-ataca")

Prof. Monael

Estruturas de Controle

- Condicional
  - Define um bloco que só será executada se uma condição for verdadeira

  Sintaxe:

```
if (<condição>)
{
        <comandos>;
        ...
}
```

Estruturas de Controle

- Condicional

  Exemplo:

```
if (idade > 18)
{
        area.acesso = "permitido";
}
```

Estruturas de Controle

- Condicional
  - O bloco a seguir do if só executa se a <condição> for verdadeira, entretanto se for preciso executar algo caso a <condição> seja falsa usa-se o comando else.

Sintaxe:

```
if (<condição>)
{
        <comandos>;
        ...
}
else
{
        <comandos>
        ...
}
```

Estruturas de Controle

- Condicional

  Exemplo:

```
if (idade > 18)
{
        area.acesso = "permitido";
}
else
{
        area.acesso = "negado";
}
```

Laboratório

- Faça um programa que gere um número inteiro aleatório, mostre-o e diga se ele é ímpar ou par.

- Dica: Veja a classe Random no JavaDoc.
- Dica 2: importe-a para seu programa. Primeira linha: `import java.util.Random;`

Estruturas de Controle

- Condicional switch - case
  - Dependendo de quantos valores é necessário testar, torna-se interessante não usar a instrução if – else.

Sintaxe:

```
switch (<variável>)
{
        case(<condição>):
        <comandos>
        ...
        [default:]
        <comandos>
        ...
}
```

Estruturas de Controle

- Condicional switch - case

  Exemplo:

```
swith (opcao)
{
        case(1):
                Menu.compraCartao();
                break;
        case(2):
                Menu.compraCheque();
                break;
        case(3):
                Menu.compraDinheiro();
                break;
        default:
                System.out.println("Opção " + opcao + " não existe");
                break;
}
```

Laboratório

- Faça um programa gere um caracter e, usando comando SWITCH – CASE, ele diga se é vogal a, e, i, o, u ou não é vogal:

- Dica : Gere números inteiros até 122 e converta para caracteres. Através de CASTING explicito.

```
char c = (char) rd.nextInt(122);
```

Estruturas de Controle

- Estruturas de Repetição
  - Suponha que peçam para você escrever de 1 a 10 na tela:

  Como faríamos para exibir todos eles?

  Solução:

```
System.out.println("1");
System.out.println("2");
System.out.println("3");
System.out.println("4");
System.out.println("5");
...
```

Estruturas de Controle

- Estruturas de Repetição
  - Pode-se usar estruturas de repetição while:

Sintaxe:

```
while(<condição de parada>)
{
        <comandos>
        ...
}
```

Estruturas de Controle

- Estruturas de Repetição
  - Escrever de 1 a 10 na tela:

    Solução:

```
int contador = 0;
while(contador < 10)
{
        System.out.println(contador);
        contador = contador + 1;
}
```

Laboratório

- Para fixar o comando repetidor while, escreva o seguinte programa e execute-o.

```
class ContaWhile{
    public static void main(String[] args)
    {
        int i=0;
        while(i<=10){
            System.out.println(i);
            i++;
        }
    }
}
```

Estruturas de Controle

- Estruturas de Repetição
  - Pode-se usar estruturas de repetição for:

  Sintaxe:

```
for(<variável contadora> = <valor inicial>; <condição de parada>; <incremento>)
{
        <comandos>
        ...
}
```

Estruturas de Controle

- Estruturas de Repetição
  - Escrever de 1 a 10 na tela:

    Solução:

```
for(int cont = 0; cont < 10; cont=cont+1)
{
        System.out.println(cont);
}
```

Laboratório

- Para fixar o comando repetidor for, escreva o seguinte programa e execute-o.

```
class ContaWhile{
    public static void main(String[] args)
    {
        for(int i=0; i<=10; i++){
            System.out.println(i);
        }
    }
}
```

Estruturas de Controle

- Estruturas de Repetição
  - Pode-se usar estruturas de repetição do - while:

Sintaxe:

```
do
{
        <comandos>
        ...
}while(<condição de parada>);
```

Estruturas de Controle

- Estruturas de Repetição
  - Escrever de 1 a 10 na tela:

    Solução:

```
int contador = 0;
do
{
        System.out.println(contador);
        contador = contador + 1;
}while(contador < 10);
```

Laboratório 1

- Para fixar o comando repetidor while, escreva o seguinte programa e execute-o.

```
class ContaWhile{
    public static void main(String[] args)
    {
        int i=0;
        do
        {
            System.out.println(i);
            i++;
        }while(i<=10);
    }
}
```

Laboratório 2

- Faça um programa que diga quantos números divisíveis por 9 existem de 0 à 1000.

Listas

- Vetores (Arrays)
  - Declarar a variável para manter o vetor,
  - Criar um objeto vetor e atribuir para esta variável
  - Armazenar informações nesse vetor

  Sintaxe da declaração e criação:

  <tipo>[] <nome da variável> = new <tipo>[<tamanho>];

  Exemplo:

```
String[] nomes = new String[20];
int[]    idades = new int[20];
```

Listas

- Acessando e Alterando elementos de um vetores (Array)
  - Isso pode ser feito através dos índices do vetor
    Sintaxe da declaração e criação:

    <nome da variável>[<indice>];

    Exemplo:

```
nomes[7] = "Ed";
int[] idades = {24, 58, 56};
System.out.println("Nome: " + nomes[7]);
valor = 2 * preco[5];
```

Listas

- Argumento por linha de comando.
  - Desde o primeiro programa em JAVA usamos a seguinte definição do método principal:

```
public static void main(String[] arguments)
```

  - Observe que String[] arguments é um vetor.
    - Para que serve arguments?

Listas

- Quando executamos um programa JAVA na linha de comando é possível passar argumentos para o método principal da seguinte forma:

  Sintaxe:

  C:>java <nome do arquivo.class> <[arg1]>,<[arg2]>,<[arg3]>

  Exemplo:

  C:>java Aluno "Michel", 18, 'B'

  – Dentro do programa estes argumentos estarão disponíveis no vetor de String arguments.

    arguments[0] contem a string "Michel"
    arguments[1] contem a string "18"
    arguments[2] contem a string "B"

Listas

- Método Length
  - Retorna o tamanho de uma lista.

  Sintaxe:

  <nome da lista>.length;

  Exemplo:

  ```
  int [] lista = {1, 2, 3, 4, 5, 6, 7, 8, 9, 0};
  int quant = lista.length;
  ```

Laboratório 1

- Faça um programa que receba uma lista de números via linha de comando e mostre o somatório e a média entre eles.

Exemplo:

```
C:\>java somador 1 2 3 4 5

Sua soma é 15
A média é 3.0
```

Listas, Lógica e Loops

- Vetores Multidimensionais

  Sintaxe da declaração e criação:

  <tipo>[][] <nome da variável> = new <tipo>[<tamanho>] [<tamanho>];

  Exemplo:

  String[][] nomes = new String[20][20];
  int[][][] idades = new int[20][40][60];

Trabalhando com Objetos

- Chamando os métodos de um objeto
  - Para se chamar métodos de um objeto, usa-se também o operador (.) ponto.

Sintaxe:

<nome do objeto>.<nome do método>(<[argumentos]>);

Exemplos:

```
cta.getSaldo();
cta.setNome("João Carlos");

livro.buscaAutor("Guimarães Rosa");
livro.excluiLivro("Dom Casmurro");
```

Trabalhando com Objetos

- Chamando os métodos de classe
  - Para se chamar métodos de um objeto, usa-se também o operador (.) ponto. Entretando, pode-se usar diretamente o nome da classe.

Sintaxe:

<nome da classe>.<nome do método>(<[argumentos]>);

Exemplos:

```
Math.max(10, 15);
String.valueOf(1458);
```

Laboratório

Faça um programa que receba através da linha de comando os valores a, b e c da Equação de 2° grau:

$$ax^2 + by + c = 0$$

E resolva segundo a fórmula de Baskara:

Ache o valor de Δ, como segue: $\Delta = b^2 - 4ac$

Encontre as raízes da equação, segundo o critério:

Se Δ > 0, há duas raízes x′ e x″

Se Δ = 0, há apenas x′

Se Δ < 0, não há raiz

$$x' = \frac{-b + \sqrt{\Delta}}{2a} \qquad x'' = \frac{-b - \sqrt{\Delta}}{2a}$$

Definindo Classes

```
class <<nome da classe>>
{
        // corpo da classe
}
```

Exemplo:

```
class Pessoa
{

}
```

Definindo Classes

Por default todas as classes herdam de Objetc. Para definir uma herança se usa a palavra-chave extends.

```
class <<nome da classe filho>> extends << nome da classe pai >>
{
        // corpo da classe
}
```

Exemplo:

```
class Funcionario extends Pessoa
{

}
```

Definindo Classes

Definindo variáveis de instância:

```
class <<nome da classe filho>> extends << nome da classe pai >>
{
        <<tipo>> <<nome da variavel>>;
}
```

Exemplo:

```
class Funcionario extends Pessoa
{
    String cargo;
}
```

Definindo Classes

Definindo variáveis de Classe:

```
class <<nome da classe filho>> extends << nome da classe pai >>
{
        static <<tipo>> <<nome da variavel>>;
}
```

Exemplo:

```
class Funcionario extends Pessoa
{
    String cargo;
    static String contratante = "UESA";
}
```

Definindo Classes

Criando métodos:

```
class <<nome da classe filho>> extends << nome da classe pai >>
{
        <<tipo de retorno>> <<nome do método>>(<<lista de argumentos>>)
        {
                // corpo do método
        }
}
```

Exemplo:

```
class Pessoa
{
    String getNome()
    {
        //corpo do método getNome
    }
}
```

Definindo Classes

- Métodos de Classe

```
class <<nome da classe filha>> extends <<nome da classe pai>>
{
        static <<valor de retorno>> <<nome do método>>(<<lista de argumentos>>)
        {
                //corpo do método
        }
}
```

Exemplo:

```
class Pessoa
{
        static String getEspecie()
        {
                //corpo do método
        }
}
```

Criando uma aplicação em JAVA

- Para ter uma aplicação é necessário ter uma classe como ponto de partida.
  - Para isso a classe deve ter o método main().
  - O método main() é o primeiro a ser chamado.

```
public static void main(String[] arguments)
```

  - public: método disponível para outras classes e objetos
  - static: é um método de classe
  - void: não retorna nenhum valor
  - (String[] arg): vetor de argumentos da linha de comando

  - Atenção: Quando main() é executada não é instanciado o objeto da classe a qual main() pertence.

Definindo a Primeira Classe

Definindo a Primeira Classe

- Definir a classe Conta.
  - Abstrair o objeto conta.
  - Definir seus atributos.
  - Definir seus métodos.

Definindo a Primeira Classe

```
class Conta
{
    String          numero;
    String          titular;
    String          cpf;
    float           saldo;
    // métodos
}
```

Definindo a Primeira Classe

```
class Conta
{
// atributos
    void saque(float valor){            }
    void deposito(float valor){         }
    void juros(){         }
    void extrato(){               }
}
```