Princípios Básicos da Linguagem JAVA

Programas Java

Com Java podemos desenvolver três tipos de programas:

- Applets
- Aplicações Desktop
- Aplicações Web (Servlet e JSP)

Programas Java

Applet

- Executa no Browser
- Métodos init(), start(), paint()
- Limitações de I/O
- Aspectos de Segurança

Aplicação Desktop

- Executa sob S.O
- Método main()
- Programa comum

Aplicação Web

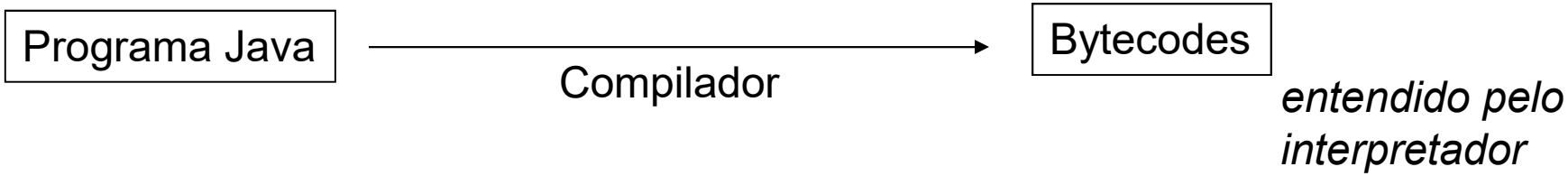
- Executado em um “Container”(Tomcat) no servidor Web
- Métodos Init(), service() e destroy()

Princípios básicos de um ambiente JAVA

Compilando o arquivo .java:

no prompt do shell

javac meuPrograma.java



O arquivo gerado pelo compilador recebe a extensão .class

Princípios básicos de um ambiente JAVA

Compilando o arquivo .java:

no prompt do shell

```
c:\ javac meuProgama.java
```

⇒ O arquivo gerado pelo compilador recebe a extensão .class

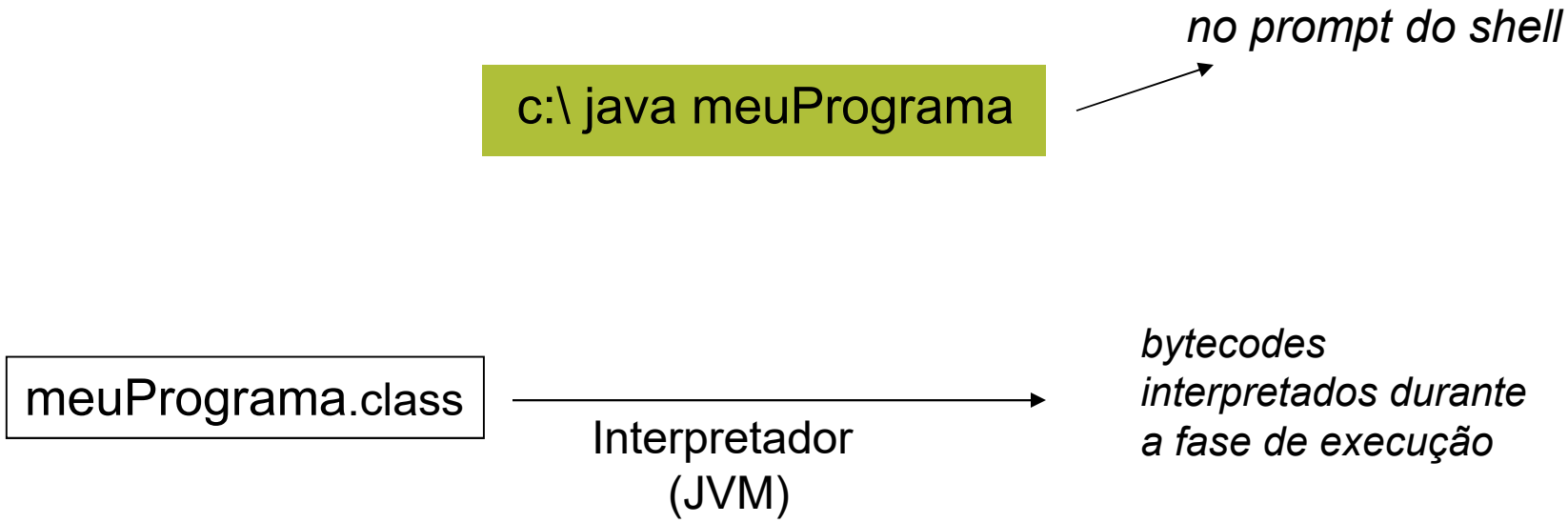
⇒ O arquivo .class pode ser carregado a partir de um disco ou de uma rede (local ou Internet).

⇒ Há três tipos de programas para o qual o carregador de arquivos atua:

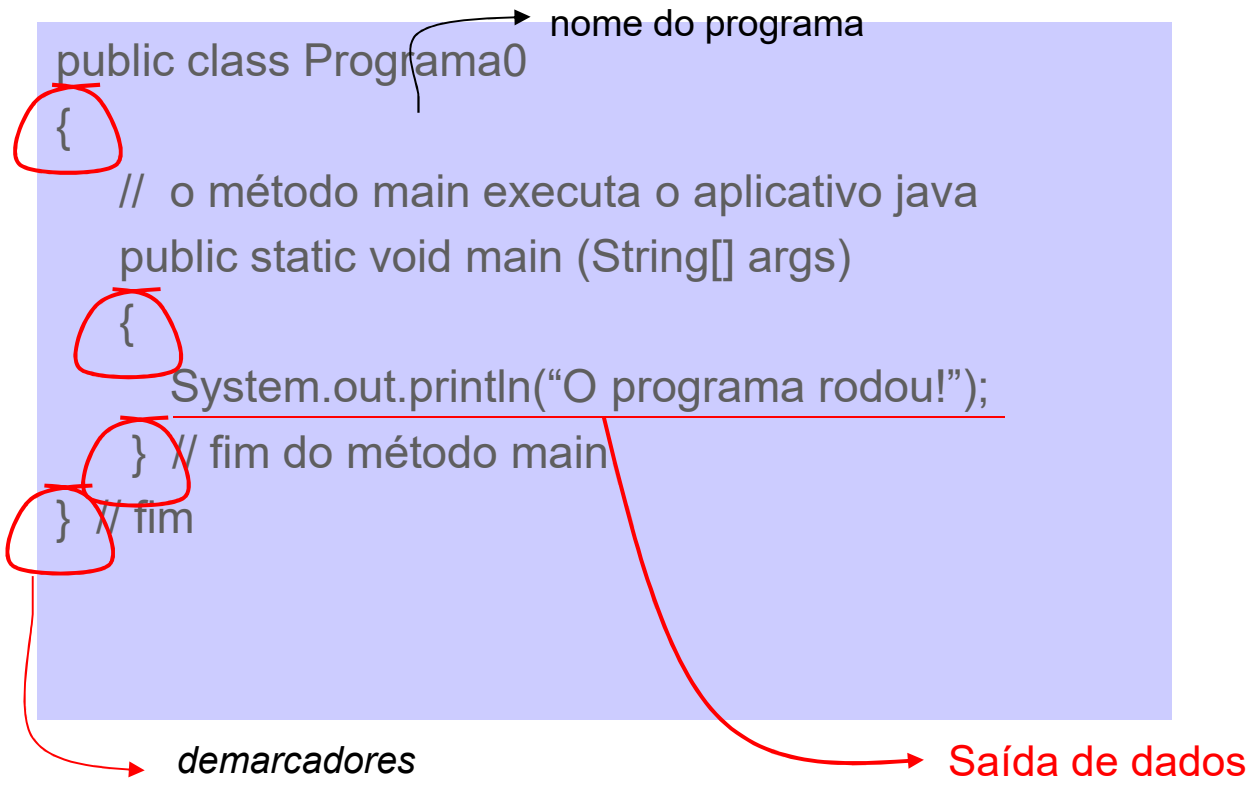
- Aplicativos (a partir do disco)
- *Applets (computador remoto)*
- *Aplicações Web (servlets e JSP)*

Princípios básicos de um ambiente JAVA

Executando programa.class no caso de um Aplicativo



Compreendendo a Construção de um programa



Compreendendo a Construção de um programa

```
// Primeiro Programa
        public class Alo
        {
            public static void main (String args[])
            {
                System.out.println("Alo Pessoal !");
            }
        }
```

```
c:/> java Alo
c:/> Alo Pessoal!
c:>
```

Compreendendo a Construção de um programa

- Compilar Programa: **c:\>javac Alo.java**
- Executar o programa: **c:\>java Alo**
- O resultado será:

```
c:\java>javac Alo.java

c:\java>java Alo
Alo Pessoal !
```

Compreendendo a Construção de um programa

O Método Main()
Forma geral de definição:

public static void main()

public: pode ser chamado por qualquer outro objeto
void: não retorna valor

⇒ Quando você executa uma aplicação Java com o Interpretador Java, você especifica o nome da classe que você quer rodar.

⇒ O interpretador invoca o método main() definido dentro daquela classe.

⇒ Controla o fluxo do programa, aloca todos os recursos que são necessários, e roda qualquer outro método que provê a funcionalidade da aplicação

⇒ O cérebro de qualquer aplicação Java é seu método main()

Entrada de Dados

O método **main** aceita um argumento simples: um array de Strings :

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

- Cada String do array é inserida na linha de comando após o nome do programa, como um argumento, permitindo que usuários afetem a operação da aplicação sem precisar recompilar :

  Ex.: **java programaNomes maria jose rui**

- Este array de Strings é o mecanismo através do qual o *runtime system* passa a informação para sua aplicação

  args[0]= maria

  args[1] = jose

  args[2] = rui

Saída de Dados

- Pelo Console:

System.out.print(“frase qualquer”)

Imprime a “frase qualquer” e o cursor continua na mesma linha esperando que seja impresso mais alguma coisa.

System.out.println(“nova frase”)

Imprime a frase “nova frase” e desta vez o cursor pula para linha de baixo, caso nova frase seja impressa esta o será na linha imediatamente abaixo.

Construção de um programa

Entrada de dados via array de String

```
// Programa Programa1.java
//  o método main executa o aplicativo java
public class Programa1 {

    public static void main (String[] args) {
        System.out.println("Bom dia sr.(a) " + args[0]);
    }
}  // fim da classe Programa1
```

```
c:/> java Programa1 Alberto
c:/> Bom dia sr.(a) Alberto
c:>
```

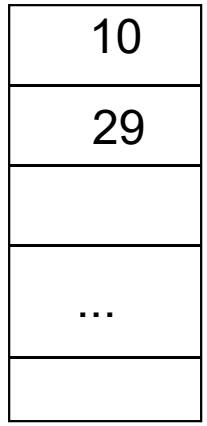
Construção de um programa

Entrada de dados via array de String

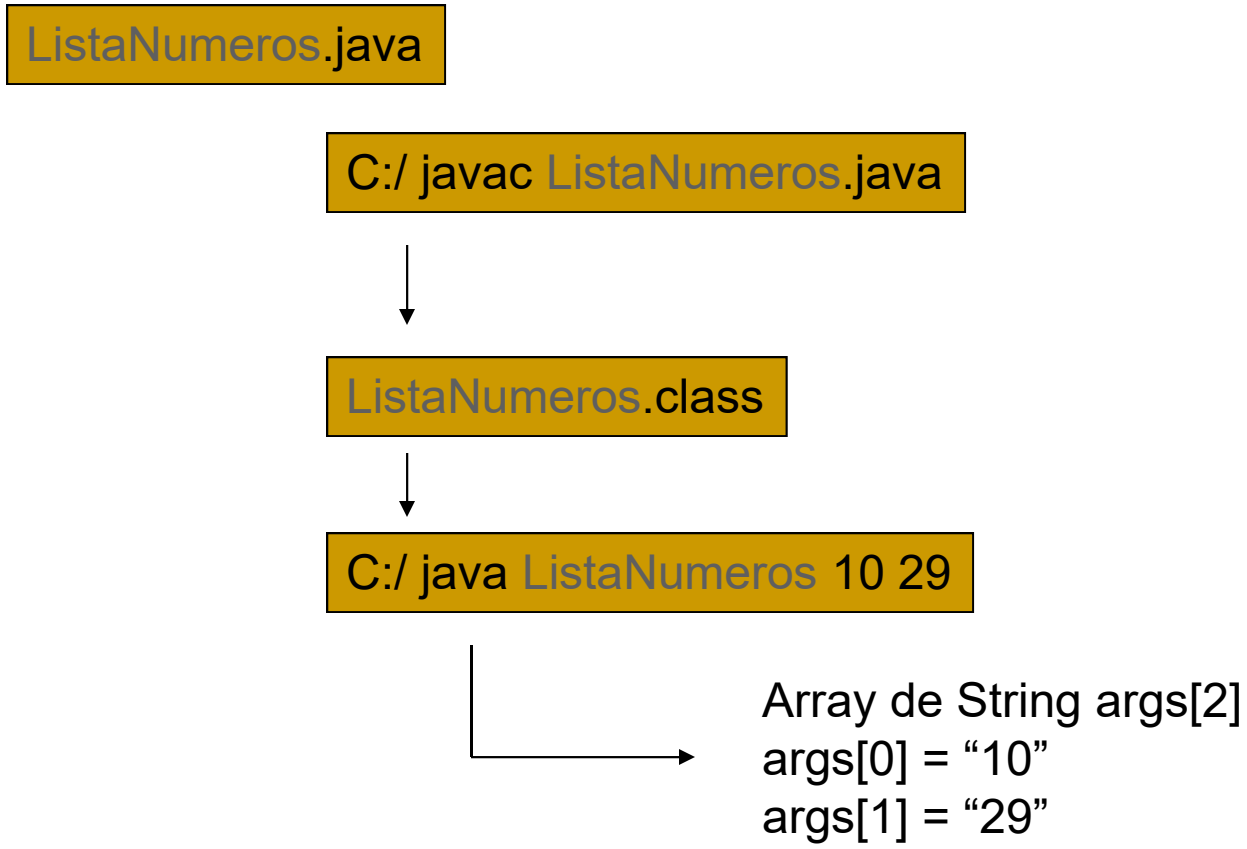
```
public class ListaNumeros {
    public static void main (String args[]) {

        System.out.println("O primeiro número eh: " + args[0]);
        System.out.println("O segundo número eh: " + args[1]);

    }  // fim do método main
}  // fim da classe ListaNumeros
```

```
c:/> java ListaNumeros 10 29
c:/> O primeiro número eh 10
c:/> O segundo número eh 29
c:/>
```

Array de Strings
args



Entrada de Dados



Estruturas Fundamentais de Programação

- Comentários

```
// Comentário de uma linha
/* Comentário de várias linhas */
/** Comentário Documentado */
```

- Blocos, ponto-e-vírgula e espaço
  - toda declaração termina com “**;**”
  - “{“ e “}” delimitam um bloco de declarações para classes, métodos ou estruturas de controle

Palavras reservadas

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

Identificadores

- Palavras utilizadas para nomear variáveis, métodos, classes e labels.

- Regras:
  - Não podem ser utilizadas palavras reservadas
  - Devem começar com uma letra, dolar ($), underscope(_)
  - Não podem começar por números

- São case sensitive

Estruturas Fundamentais de Programação

Tipos Básicos

- **boolean** **true** ou **false**
- **char** caracter UNICODE (16 bits)
- **byte** número inteiro com sinal (8 bits)
- **short** número inteiro com sinal (16 bits)
- **int** número inteiro com sinal (32 bits)
- **long** número inteiro com sinal (64 bits)
- **float** número em ponto-flutuante (32 bits)
- **double** número em ponto-flutuante (64 bits)

Estruturas Fundamentais de Programação

- Declarações de variáveis

```
int x, y;              // variáveis inteiras
float z;               // variável float
double  w;             // variável double
boolean verdade;       // variável booleana
char c;                // variável caracter
```

- As variáveis locais devem ser obrigatoriamente inicializas
- Inicialização default do Java para variáveis de instância
  - variáveis numéricas com zero
  - variáveis boolean com false
  - variáveis String com null

Estruturas Fundamentais de Programação

Atribuições e inicialização

```
int x, y;                          // declaração
float z = 3.144f;                  // declaração com atribuição
double  w = 3.1415;
boolean verdade = true;
char c, d;                         // declaração
c = 'A';                           // atribuição
d = '\u0013';                      /* representação hexadecimal
                                        de um caracter */

x = 6;
y = 1000;
```

Operadores Matemáticos

Operador Algébrico	Exemplo	Significado
+	x + y	Soma
-	x – y	Subtração
*****	x * y	Multiplicação
/	x / y	Divisão
%	x % y	Mod
==	x == y	Igual a
!=	x != y	Não é igual a
>	x > y	É maior que
<	x < y	É menor que
>=	x >= y	É maior ou igual a
<=	x <= y	É menor ou igual a

Operadores de incremento e decremento

Operador	Chamado de	Expressão de exemplo	Explicação
++	Pré-incremento	++a	Incrementa de 1 a variavel a, e depois utiliza seu valor na proxima expressao em que ele reside
++	Pos-incremento	a++	Utiliza o valor atual de a na expressao em que a reside e depois incremente a de 1
--	Pré-decremento	--b	Decrementa de 1 a variavel b, e depois utiliza seu valor na proxima expressao em que ele reside
--	Pos-decremento	b--	Utiliza o valor atual de b na expressao em que b reside e depois decremente b de 1

Operadores de incremento e decremento

```
// Pré-incremento e Pos-incremento

public class Incremento{
    public static void main( String args[] )
    {
        int c;
          c = 5;
          System.out.println( --c );
          System.out.println( c++ );
          System.out.println( ++c );

          System.out.println(c--);

          System.out.println( ++c );
          System.out.println( ++c );
          System.out.println( c++ );
     }
}
```

Operadores de incremento e decremento

```
public class Lista {
   public static void main (String args[] )
   {
      int a=1;
       int b=1;
       int c=1;
       String[] lista = {"banana","laranja","pera"};
        lista [--a] += "maca";
        lista [b++] += "lima";
        lista [c--] += "exportacao";
       for (int i=0 ; i < lista .length; i++)
          System.out.println(lista [i]);
        }
}
```

Qual será o conteúdo do array lista depois da execução do programa ?

Construção de Programas

Entrada de dados através da classe **Scanner**

A classe Scanner surgiu a partir da versão 1.5 do Java. Ela pertence ao pacote java.util e deve ser explicitamente importada no programa.

```
Scanner <variavel> = new Scanner(System.in);
```

A classe Scanner possui os seguintes métodos para leitura:

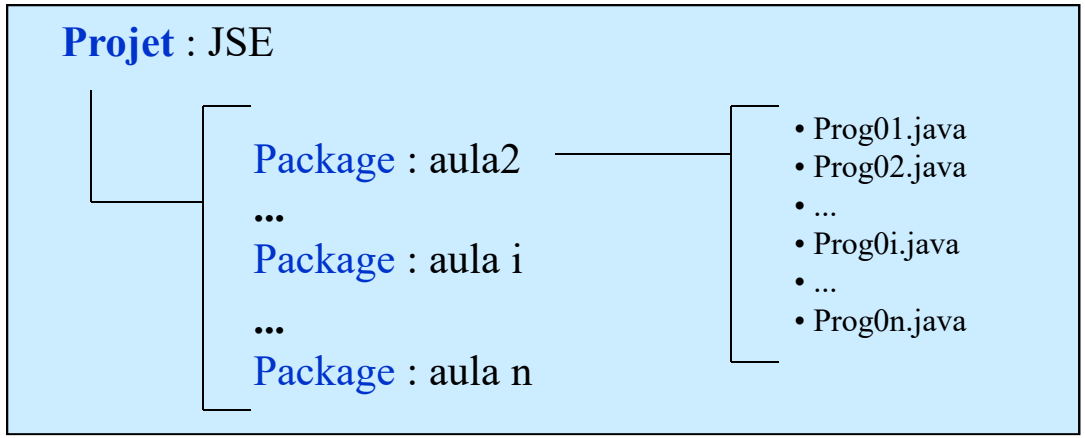
```
        String nome = entrada.nextLine();        → lê uma String
ou      String nome = entrada.next();            → lê uma String

        byte num = entrada.nextByte();           → lê um byte
        short num = entrada.nextShort();         → lê um short
        int num = entrada.nextInt();             → lê um int
        long num = entrada.nextLong();           → lê um long
        float num = entrada.nextFloat();         → lê um float
        double num = entrada.nextDouble();       → lê um double
```

Construção de Programas

Vamos entrar no ambiente de desenvolvimento do Eclipse.

Analogamente como realizamos no ambiente DOS, iremos organizar os nossos Programas, nossas classes, em uma hierarquia de diretórios.

Crie um projeto, em seguida crie um package, e suas classes serão criadas dentro do seu package. Vamos estabelecer um critério para organização.



Construção de Programas

```
import java.util.Scanner;

public class LeVariaveis {
    public static void main (String args[]) {
          Scanner entrada = new Scanner(System.in);
          System.out.println("Digite o nome: ");
          String nome = entrada.next();
          System.out.println("Digite o endereço: ");
          String endereço = entrada.next();
          System.out.println("Digite o cep: ");
           int cep = entrada.nextInt();

     }  // fim do método main
}  // fim da classe LeVariaveis
```

Construção de Programas

```
// Programa Programa.java
import java.util.Scanner;

public class Programa {

    public static void main (String args[]) {
            Scanner entrada = new Scanner(System.in);
            System.out.println("Entre com o primeiro numero: ");
            int num1 = entrada.nextInt();
            System.out.println("Entre com o segundo numero: ");
            int num2 = entrada.nextInt();
            System.out.println("A soma eh " + (num1+num2));
    }  // fim do método main
}  // fim da classe Programa
```

Acrescente a subtração, divisão e multiplicação ao programa.

Note que a divisão por zero com tipos *inteiros* gera um erro de execução, entretanto o interpretador consegue calcular quando os tipos são *double*, emitindo o resultado **infinity**.