

# Linguagem de Programação C++

## Polimorfismo

Pode-se especificar uma classe abstrata em C++. Para tal, defini-se uma função-membro virtual como uma função virtual pura. Desta forma, a classe base que contém uma função virtual pura não pode ser instanciada.

Uma função virtual é definida como pura igualando seu protótipo a zero. Vejamos um exemplo:

```

#include <iostream>
using namespace std;
class Base {
    public:
        virtual void func()=0;
};
class Derivada: public Base {
    public:
        void func()
        {
            cout << "Esta eh func() de derivada" << endl;
        }
};
int main()
{
    Base *ptr;
    Derivada objeto1;
    ptr = &objeto1;
    ptr->func();
    Base objeto2; //Erro: pois a classe base é abstrata
    return 0;
}
422 }

```

# Linguagem de Programação C++

## Polimorfismo

Um aluno atento deve ter se perguntado:

Existe uma maneira de determinar em tempo de execução para qual das classes derivadas um ponteiro para a classe base está apontando?

A resposta é sim. A linguagem C++ disponibiliza para esta finalidade o *dynamic\_cast*.

O exemplo a seguir o utiliza.

```

#include <iostream>
using namespace std;
class Base {
    public:
        virtual void func()=0; };
class Derivada1: public Base {
    public:
        void func() { cout << "Esta eh func() de derivada1" << endl; } };
class Derivada2: public Base {
    public:
        void func() { cout << "Esta eh func() de derivada2" << endl; } };
int main()
{
    Base *ptr;
    Derivada1 objeto1, *ptr2;
    Derivada2 objeto2;
    ptr = &objeto1;
    ptr2 = dynamic_cast <Derivada1 *> (ptr);
    if (ptr2)
        ptr->func();
    ptr = &objeto2;
    ptr->func();
    return 0;
}

```

# Linguagem de Programação C++

## Exercício:

Utilizando este conceito implemente uma nova opção no menu do exercício do slide 416 a qual possibilita determinar o custo total apenas dos pacotes que serão entregues à noite.

```
//conteúdo do arquivo principalpacote.cpp
```

```
...
```

```
"5 - Imprimir o custo total apenas dos pacotes que serao  
entregues a noite;" << endl <<
```

```
...
```

```
case 5:
```

```
{
```

```
double auxiliar=0;
```

```
PacoteNoite *ptrAux;
```

```
for (int i=0; i<numeroDeElementos; i++)
```

```
{
```

```
ptrAux = dynamic_cast <PacoteNoite *> (vetor[i]);
```

```
if (ptrAux)
```

```
auxiliar += vetor[i]->calculaCusto();
```

```
}
```

```
cout << endl << "Custo total das postagens para " <<
```

```
"entrega a noite: " << auxiliar << endl;
```

```
}
```

```
break;
```

```
...
```

```
}while(opcao!=6);
```

```
return 0;
```

```
}
```

# Linguagem de Programação Java

# Linguagem de Programação Java

## Breve histórico:

- Green (1991) – projeto de pesquisa corporativa da Sun Microsystems;
- Desenvolveu uma linguagem baseada em C++ denominada Oak (árvore de carvalho);
- Desenvolvida por James Gosling;
- Renomeada para Java;
- Apresentada ao público em 1995.

# Linguagem de Programação Java

## Características básicas:

- Desenvolvida inicialmente para dispositivos eletrônicos inteligentes;
- Utilizada para adicionar conteúdo dinâmico na World Wide Web;
- Atualmente, também utilizada para desenvolver aplicativos corporativos de grande porte.

# Linguagem de Programação Java

Java é uma linguagem de programação orientada a objetos projetada para ser portátil a todas as plataformas. Esta portabilidade baseia-se no fato da linguagem ser interpretada.

O processo de compilação gera um código independente de máquina denominado *bytecode*.

Durante a execução o bytecode é interpretado por uma máquina virtual (JVM) instalada na máquina real, em outras palavras, a JVM lê o bytecode e o traduz para uma linguagem que o computador possa entender.

Sendo assim, para portar Java para uma arquitetura específica, basta instalar a máquina virtual específica para a arquitetura em questão.

# Linguagem de Programação Java

Antes de partirmos para a construção de nosso primeiro programa Java vamos analisar algumas características da linguagem, como os tipos primitivos, identificadores, operadores, e etc.

Iniciaremos pelos **Tipos Primitivos**.

Java não trata-se de uma linguagem de programação orientada a objetos pura, ou seja, ela dispõe de um conjunto de tipos primitivos que não constituem objetos.

Estes tipos primitivos possibilitam a representação de valores: booleanos, caracteres, numéricos inteiros e numéricos em ponto flutuante.

# Linguagem de Programação Java

## TIPOS PRIMITIVOS

➤ Palavra-reservada: boolean - define variáveis lógicas

➤ Podem assumir valores true ou false

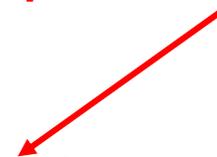
➤ O valor padrão é false

➤ Ocupa 1 bit

➤ Exemplo de declaração:

```
boolean sinalizador, sinalizador2 = true;  
sinalizador = true;
```

Operador de atribuição



# Linguagem de Programação Java

## TIPOS PRIMITIVOS

- Palavra-reservada: char - define variáveis caractere
  - Capazes de armazenar valores de caracteres Unicode
  - Ocupa 16 bits
  - O valor padrão é o caracter NULL
  - Exemplo de declaração:  
char caractere1, caractere2 = 'a';  
caractere1 = 'b';

# Linguagem de Programação Java

## TIPOS PRIMITIVOS

- Palavra-reservadas: byte, short, int e long - definem variáveis do tipo inteiro

<u>Tipo</u>	<u>Tamanho</u>	<u>Faixa de valores</u>
byte	8 bits	-128 a 127
short	16 bits	-32.768 a 32,767
int	32 bits	-2.147.483.648 a 2.147.483.647
long	64 bits	-9.223.372.036.854.775.808 a 9.223.372.036.854.775.807

O valor padrão destes tipos é 0

Exemplo de declaração:

```
int v1=3, v2;
```

```
long v3;
```

# Linguagem de Programação Java

## TIPOS PRIMITIVOS

- ➔ Palavra-reservadas: float e double - definem variáveis do tipo real

<u>Tipo</u>	<u>Tamanho</u>	<u>Faixa de valores</u>
float	32 bits	$\pm 1.40239846 \times 10^{-45}$ a $\pm 3.40282347 \times 10^{+38}$ com nove dígitos de precisão
double	64 bits	$\pm 4.94065645841246544 \times 10^{-324}$ a $\pm 1.79769313486231570 \times 10^{+308}$ com dezoito dígitos de precisão

O valor padrão destes tipos é 0.0

Exemplo de declaração:

```
float v4=3.1, v5;
```

```
double v6=3.1415921234;
```

# Linguagem de Programação Java

## IDENTIFICADORES

Compostos por sequências de caracteres que devem obedecer as seguintes regras:

- podem ser compostos por letras, dígitos e pelos símbolos `_` e `$`;
- não podem ser inicializados por um dígito;
- não podem ser iguais a palavras reservadas.

**Observação:** Java é sensível ao caso.

Para melhorar a legibilidade dos programas em Java, vamos adotar a grafia camelo.

# Linguagem de Programação Java

## ➤ Operadores Aritméticos

➤ Unários: -, --, ++

Exemplos:     -1  
                  -5.9  
                  ...  
                  short a;  
                  ...  
                  a++;

# Linguagem de Programação Java

## ➤ Operadores Aritméticos

➤ Binários: +, -, \*, /, %

### Associação

<i>Símbolo</i>	<i>Operação</i>
+	Soma
-	Subtração
*	Multiplicação
/	Divisão real e Quociente da divisão inteira
%	Resto da divisão inteira

# Linguagem de Programação Java

## ➔ Operadores Aritméticos

### Precedência (Hierarquia nas operações)

Hierarquia	Operação
1	- (unário)
2	*, /, %
3	+, -

# Linguagem de Programação Java

## ➤ Operadores Relacionais

Operador	Ação
>	maior que
>=	maior ou igual a
<	menor que
<=	menor ou igual a
==	igual a
!=	diferente de

# Linguagem de Programação Java

## ➤ Operadores Lógicos

Operador	Ação
&&	e
	ou
!	negação

# Linguagem de Programação Java

## ➤ Operadores Lógicos bit-a-bit

Operador	Ação
&	e
	ou
^	xou
<<	deslocamento a esquerda
>>	deslocamento a direita
>>>	deslocamento a direita com preenchimento de zeros
~	complemento

# Linguagem de Programação Java

## Operadores de Atribuição

=, +=, -=, \*=, /=, %=, <<=, >>=, >>>=, &=, |=, ^=

Exemplos:     a=5;  
                  a+=5;  $\Leftrightarrow$  a=a+5;  
                  a-=5;  $\Leftrightarrow$  a=a-5;

# Linguagem de Programação Java

## ➤ Operador Condicional

?:

Exemplo:  $E ? V1 : V2$

# Linguagem de Programação Java

## COMENTÁRIOS

`/* Determina comentários em  
múltiplas linhas */`

`// comentário até o final da linha`

`/** Comentário específico de Java. Os textos destes  
comentários podem ser utilizados pela ferramenta  
javadoc para gerar a documentação do código em  
formato hipertexto */`

# Linguagem de Programação Java

## CARACTERES ESPECIAIS OU SEQÜÊNCIAS DE ESCAPE

Caractere	Significado
\n	nova linha
\t	tab
\b	backspace
\r	retorno de carro
\f	avança página da impressora (form feed)
\\	barra invertida
\'	apóstrofe
\"	aspas

# Linguagem de Programação Java

## ESTRUTURAS DE CONTROLE DE FLUXO

```
if (condicao)  
    comando
```

```
if (condicao)  
{  
    sequenciaDeComandos  
}
```

# Linguagem de Programação Java

## ESTRUTURAS DE CONTROLE DE FLUXO

```
if (condicao)
    comando1
else
    comando2
```

# Linguagem de Programação Java

## ESTRUTURAS DE CONTROLE DE FLUXO

```
if (condicao)
{
    sequenciaDeComandos1
}
else
{
    sequenciaDeComandos2
}
```

# Linguagem de Programação Java

## ESTRUTURAS DE CONTROLE DE FLUXO

switch (variavel) // variavel deve ser inteira ou caractere

```
{  
    case valor1:  
        blocoDeComandos  
        break;  
    case valor2:  
        blocoDeComandos  
        break;  
    ...  
    default:  
        blocoDeComandos  
}
```

# Linguagem de Programação Java

## ESTRUTURAS DE CONTROLE DE FLUXO

while (condicao)

{

    blocoDeComandos

}

do

{

    blocoDeComandos

} while (condicao);

# Linguagem de Programação Java

## ESTRUTURAS DE CONTROLE DE FLUXO

```
for (inicializacao; condicao; incremento)
{
    blocoDeComandos
}
```

break

continue

# Linguagem de Programação Java

## ESTRUTURAS DE CONTROLE DE FLUXO

Em situações onde há diversos laços de repetição (comandos de iteração) aninhados, os comandos *break* e *continue* transferem a execução para o ponto correspondente no bloco onde ocorrem.

Se for necessário especificar a transferência para outro laço de repetição, os comandos *break* e *continue* rotulados podem ser utilizados.

Veremos um exemplo:

# Linguagem de Programação Java

```
...  
label: for( ... ; ... ; ... )  
{  
    ...  
    while ( ... )  
    {  
        ...  
        if ( ... )  
            break label;  
        ...  
    }  
    ...  
}  
...
```

# Linguagem de Programação Java

## ESTRUTURA GERAL DE UM PROGRAMA EM JAVA

```
public class NomeDaClassePrincipal
{
    public static void main (String args[])
    {
        ...
    }
}
```

# Linguagem de Programação Java

## ESTRUTURA GERAL DE UM PROGRAMA EM JAVA

```
public class ClassePrincipal
{
    public static void main (String args[])
    {
        System.out.print("Primeiro programa Java.");
    }
}
```

# Linguagem de Programação Java

## ESTRUTURA GERAL DE UM PROGRAMA EM JAVA

```
public class ClassePrincipal
{
    public static void main (String args[])
    {
        System.out.println("Primeiro programa Java.");
    }
}
```

# Linguagem de Programação Java

## ESTRUTURA GERAL DE UM PROGRAMA EM JAVA

```
public class ClassePrincipal
{
    public static void main (String args[])
    {
        System.out.printf("%c%s%c", '\n',
            "Primeiro programa Java.", '\n');
    }
}
```

# Linguagem de Programação Java

```
//exemplo de leitura de dados inteiros do teclado
import java.util.Scanner;
public class Soma
{
    public static void main(String args[])
    {
        Scanner input = new Scanner(System.in);
        int numero1;
        int numero2;
        int soma;
        System.out.print("Forneca um valor inteiro: ");
        numero1 = input.nextInt();
        System.out.print("Forneca outro valor inteiro: ");
        numero2 = input.nextInt();
        soma = numero1 + numero2;
        System.out.printf("A soma de eh %d\n", soma);
    }
}
```

# Linguagem de Programação Java

## COMPILAÇÃO

Para compilar um programa em Java utilizamos o comando:

```
javac nomeDoArquivo.java
```

O comando anterior compilará o código fonte contido no arquivo e gerará, caso o mesmo não apresente erros de compilação, o bytecode armazenando-o no arquivo:

```
nomeDoArquivo.class
```

O qual pode ser executado pela JVM com o seguinte comando:

```
java nomeDoArquivo
```