Curso de Introdução ao Java

Curso de Introdução ao Java

Objetivo do curso

- Conhecer a Plataforma Java
- Conhecer a sintaxe e as características do Java
- Conhecer as APIs do Java
- Escrever aplicações simples em Java
- Entender e aplicar conceitos da Programação Orientada a Objetos

Curso de Introdução ao Java

A Tecnologia Java

A tecnologia Java é composta por uma gama de produtos, baseados no poder da rede e na idéia de que um software deveria ser capaz de rodar em diferentes máquinas, sistemas e dispositivos. Por diferentes dispositivos entendemos: computadores, servidores, notebooks, handhelds, PDAs (Palm), celulares, TV, geladeiras e tudo mais o que for possível.

Os programas feitos em Java rodam em diferentes ambientes graças a um componente da plataforma chamado JVM (Java Virtual Machine) – que é um tipo de tradutor de código Java para instruções específicas de cada sistema e dispositivo.

A tecnologia Java foi lançada em 1995, e desde então tem crescido em popularidade e se tornado uma plataforma muito estável e madura. Atualmente a tecnologia Java está em sua segunda versão, chamada de Java 2 Platform.

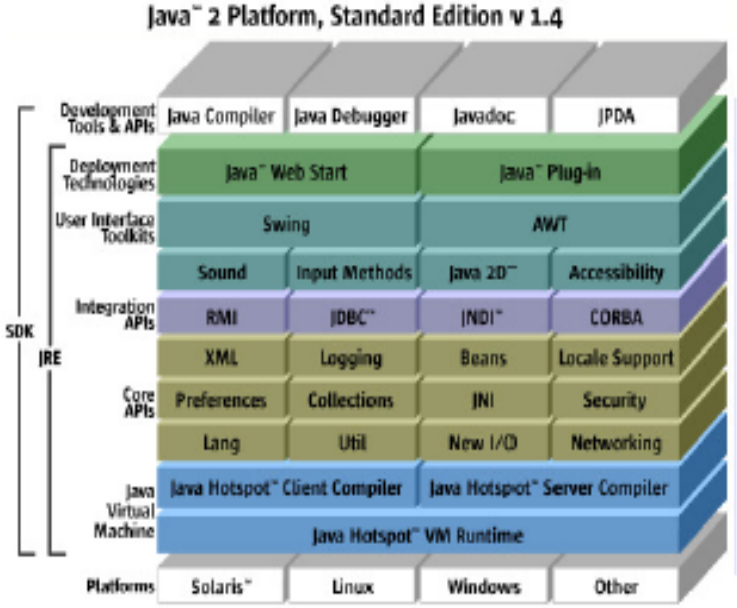
A tecnologia Java é, basicamente, sub-dividida em:

- **J2SE (Java 2 Standard Edition)**
- **J2EE (Java 2 Enterprise Edition)**
- **J2ME (Java 2 Micro Edition)**
- **Java Card**
- **Java Web Services**

A Tecnologia Java – Java Standard Edition (JSE)

A JSE é uma rica plataforma que oferece um completo ambiente para o desenvolvimento de aplicações para clientes e servidores. A J2SE é também a base das tecnologias J2EE e Java Web Services, e é dividida em dois grupos conceituais: Core Java e Desktop Java.

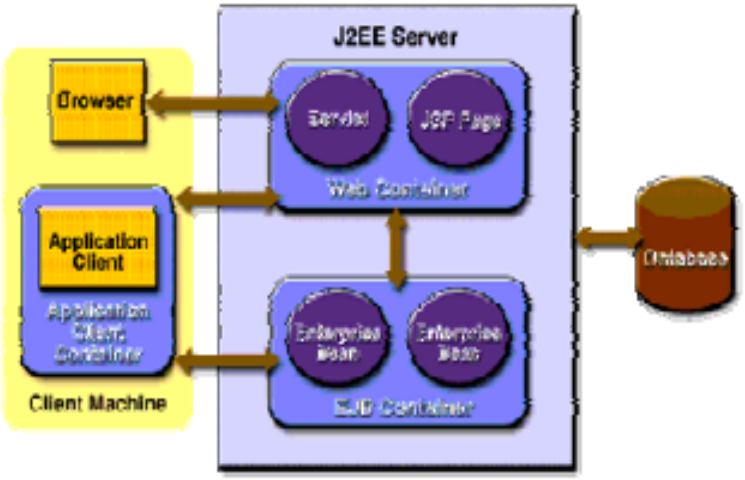
A Sun distribui a JSE na forma de um SDK (Software Development Kit), em conjunto com uma JRE (Java Runtime Environment). O pacote do SDK da J2SE vem com ferramentas para: compilação, debugging, geração de documentação (javadoc), empacotador de componentes (jar) e a JRE, que contém a JVM e outros componentes necessários para rodar aplicações Java.



Curso de Introdução ao Java

A Tecnologia Java – Java Enterprise Edition (JEE)

- A tecnologia JEE não é um produto, mas sim de uma especificação definida pela Sun.
- Simplifica as aplicações empresariais e multi-camadas
- É baseando nos componentes padronizados, modulares e reusáveis, os (EJB)
- Oferecendo um conjunto completo de serviços para estes componentes
- Manipula muitos detalhes do comportamente da aplicação automaticamente
- Não precisa reaprender a programa, pois se utiliza dos mesmo recursos do Java (JSE)
- Roda em servidores de aplicações JEE diferentes e padronizados pela Sun



A tecnologia JEE não está no escopo deste curso

A Tecnologia Java – Java Micro Edition (JME)

A tecnologia JME é voltada para aplicações que rodam em pequenos dispositivos como celulares, PDAs, smart cards e etc. Ela possui uma API bastante completa para o desenvolvimento de aplicações para pequenos dispositivos. A tecnologia JME não está no escopo deste curso.

A Tecnologia Java – Web Services

- Baseada na tecnologia XML
- Usado para troca de informações pela rede
- Muito utilizado por sites de e-commerce
- Utiliza padrões definidos (SOAP, ...)
- A API JAXP (Java API for XML Processing) oferece facilidades para Web Services

Curso de Introdução ao Java

O que é Java, afinal???

- Java é uma linguagem de programação
- Java é um ambiente de desenvolvimento
- Java é uma completa plataforma de soluções para tecnologia

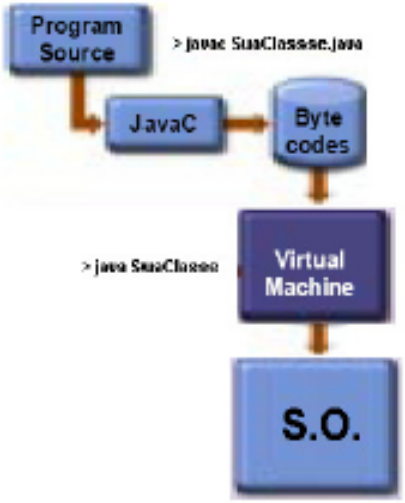
Java J2SE JNI JRE SDK
J2EE JAXP AWT
JSP JDBC
J2ME
CORBA
JVM Swing
XML
API JINI RMI
Servlet

Curso de Introdução ao Java

Fundamentos da Linguagem Java

- Java não necessita de um editor específico (Notepad é o suficiente)
- Existem dezenas de editores completos (IDEs) para Java (livres ou não)
- É portavel para qualquer ambiente/plataforma - "Write once, run everywhere!"
- Java é orientado ao objeto

Os programas em Java, quando compilados, são convertidos para um código intermediário (bytecode), que é verificado, carregado na memória e então interpretado pela JVM (Java Virtual Machine). O Java NÃO gera executáveis, nem código nativo para o SO.



Curso de Introdução ao Java

Primeiro Programa em Java

Arquivo: PrimeiroPrograma.java

```
public class PrimeiroPrograma {
 public static void main( String[] args ) {
   System.out.println( "Meu primeiro programa em Java" );
 }
}
```

Todo programa começa pelo método *main( )*. Que é o seu ponto de partida.

Compilando o código-fonte:

```
javac PrimeiroPrograma.java
```

Executando o programa:

```
java PrimeiroPrograma
```

Saída gerada:

```
Meu primeiro programa em Java
```

Curso de Introdução ao Java

Método *main( )*

A assinatura do método *main( )*, que é o ponto de partida de um programa Java e deve ser feito como abaixo:

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

O parâmetro passado para o método *main( )* é um array de Strings, que contém os valores dos argumentos passados na linha de comando da execução do programa. Exemplo:

```
java PrimeiroPrograma argumento1 argumento2 argumento3
```

Cada palavra passada como argumento é um ítem do array, parâmetro do *main( )*.

Curso de Introdução ao Java

Comentários

Os comentários em Java podem ser por linha ou em bloco:

Por linha:

```
// isto é um comentário e inicia com duas barras.
```

Em bloco:

```
/*
  Comentário em bloco aceita múltiplas linhas
  Não utilize comentários aninhados
*/
```

Curso de Introdução ao Java

Exercícios

1) Declare uma variável que represente um número inteiro e inicie com o valor “10”.

2) Declare três variáveis com tipos diferentes, sem atribuir valor. Depois atribua um valor qualquer a elas.

3) Crie uma variável do tipo *int*, atribuindo um valor a ela. Depois crie uma variável do tipo *double*, atribuindo a ela o valor da primeira variável criada.

Curso de Introdução ao Java

Palavras-Chaves do Java

O Java possui 53 palavras-chaves e palavras reservadas:

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

Nenhuma das palavras acima podem ser usadas como identificadores (nomes de variáveis, atributos, classes), ou para outro propósito, a não ser o especificado para aquela determinada palavra. As palavras *goto* e *const*, apesar de reservadas, não tem utilizadade algum no Java.

Curso de Introdução ao Java

Palavras-Chaves do Java

O Java possui 53 palavras-chaves e palavras reservadas:

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

Nenhuma das palavras acima podem ser usadas como identificadores (nomes de variáveis, atributos, classes), ou para outro propósito, a não ser o especificado para aquela determinada palavra. As palavras *goto* e *const*, apesar de reservadas, não tem utilizadade algum no Java.

Curso de Introdução ao Java

Exercícios

7) Calcula a área de uma circunferência com raio 12, onde PI = 3.1415 e área = PI * r^2.

8) Calcule o resto da divisão de 99 por 4.

9) Divida um número por 2 sem utilizar o operador /.

10) Multiplique um número por 8, sem utilizar o operador *.

Desafio 2:
Declare um inteiro de valor 10 e mostre na tela o valor do terceiro bit mais significativo (da direita para a esquerda).

Operador Condicional: ?

É também conhecido como operador ternário, pois trabalha com 3 operandos.
Ele avalia o primeiro operando.
Caso a avaliação retorne true, ele executa o segundo operando.
Senão, ele executa o terceiro operando.
O segundo e terceiro operandos DEVEM ser do mesmo tipo (senão, use cast).

O código do operador ternário abaixo:

```
int x = 10;
int y = (x > 10) ? x : x+1;
```

é semelhante ao código abaixo:

```
int x = 10;
int y;
if( x > 10 ) {
  y = x;
} else {
  y = x + 1;
}
```

Operadores de Atribuição

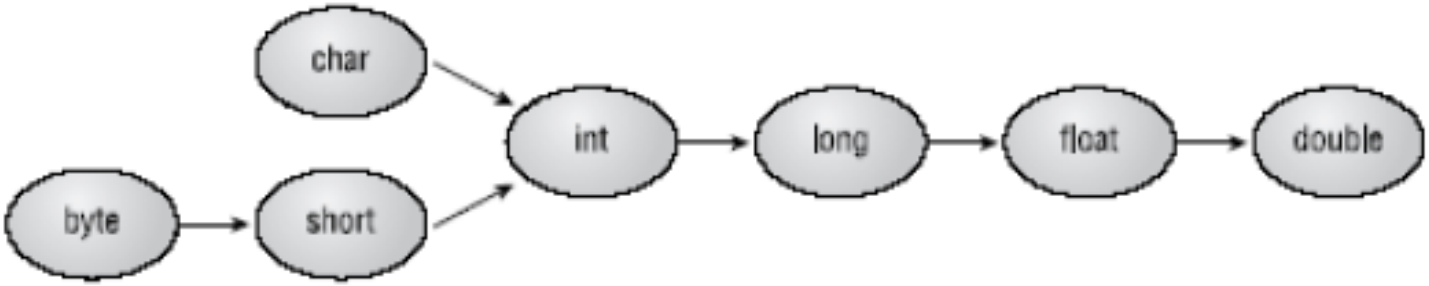
Estes operadores atribuem um novo valor a uma variável ou expressão.
O operador = apenas atribui um valor.
Os operadores +=, -=, *= e /= calculam e atribuem um novo valor.

```
int i = 10;

int dois = 1;
dois += 1;       // dois = dois + 1;

int cinco = 7;
cinco -= 2;      // cinco = cinco - 2;

int dez = 5;
dez *= 2;        // dez = dez * 2;

int quatro = 12;
quatro /= 3;     // quatro = quatro / 3;
```

Conversão de Tipos Primitivos

Permite a conversão entre tipos diferentes
Deve ser explícito quando for de um tipo maior para um menor (narrowing)
Pode ser implícito (ou explícito) de um tipo menor para um maior (widening)
Conversão para tipos menores pode causar perda de precisão e truncamento.

```
double d = 1.99d;
int i = (int) d;      // i recebe o valor 1

short s = 15;
long x = s;           // conversão widening
long y = (long) s;    // não é necessário
```



O widening segue o sentido das flechas. Narrowing é no sentido contrário.

Promoção Aritmética

Ocorre com as operações aritméticas entre tipos primitivos numéricos diferentes.
O menor tipo é automaticamente convertido para o maior tipo.

```
public class PromocaoMatematica {
  public static void main( String[] args ) {
    double d = 100.99;
    int i = 100;
    //aqui ocorre a promoção matemática
    //i é convertido para double e então multiplicado
    d = d * i;
    //ao contrário é necessário informar o casting
    long x = 12345;
    float pi = 3.14f;
    x = x * (long) pi;
    //ou então, converte apenas o resultado
    x = (long) (x * pi);
  }
}
```

Controles de Fluxo do Programa

Cláusula if( ) / else

```
public class ClausulaIf {
  public static void main( String[] args ) {
   int idade = 20;
   if( idade <= 12 ) {
     System.out.println( "Criança" );
   }
   if( idade > 12 && idade <= 19  ) {
     System.out.println( "Adolescente" );
   }
   if( idade > 19 && idade <= 60  ) {
     System.out.println( "Adulto" );
   }
   if( idade > 60 ){
     System.out.println( "Idoso" );
   }
  }
}
```

Controles de Fluxo do Programa

Cláusula if( ) / else

```
public class ClausulaIf {
  public static void main( String[] args ) {
   int idade = 20;
    if( idade <= 12 ) {
      System.out.println( "Criança" );
    }
    else if( idade <= 19  ) {
      System.out.println( "Adolescente" );
    }
    else if( idade <= 60  ) {
      System.out.println( "Adulto" );
    }
    else {
      System.out.println( "Idoso" );
    }
  }
}
```

Curso de Introdução ao Java

Exercícios

11) Crie uma variável inteira com um valor qualquer e verifique se o valor desta variável é menor que 15 ou maior que 100.

12) Crie uma variável com valor de ponto flutuante com um valor qualquer e verifique se o valor desta variável está entre 1.99 e 5.99, inclusive.

13) Agora compare se o valor das duas variáveis acima são iguais.

14) Calcule o valor da multiplicação de um *int* por um *double*, atribuindo o valor a um *int*.

Curso de Introdução ao Java

Exercícios

15) Crie um programa que receba dois argumentos (nomes) e os exiba na ordem alfabética correta.

16) Crie um programa que receba dois argumentos e calcule a área de um quadrilátero e exiba na tela com a seguinte mensagem:

"Lado a = <a>"
"Lado b = <b>"
"A área é = <valor>".

17) Incremente o programa de cálculo de área (16) para exibir ao final a mensagem:

"A figura é um quadrado" caso seja um quadrado,
ou, "A figura é um retângulo", caso seja um retângulo.

Curso de Introdução ao Java

Controles de Fluxo do Programa

Cláusula switch( )

```
public class ClausulaSwitch {
  public static void main( String[] args ) {
    int numero = 1;
    switch( numero ) {
      case 1 :
        System.out.println( "UM" );
        break;
      case 2 :
        System.out.println( "DOIS" );
        break;
      case 3 :
        System.out.println( "TRES" );
        break;
      default :
        System.out.println( "NENHUM" );
        break;
    }
  }
}
```

O switch recebe um argumento do tipo int.

Curso de Introdução ao Java

Controles de Fluxo do Programa

Laço while( )

```
public class LacoWhile {
  public static void main( String[] args ) {
    int i = 0;
    //laço while() com bloco de código definido
    while( i < 10 ) {
      System.out.println( "Linha: " + i );
      i++;
    }
  }
}
```

A expressão é avaliada antes de executar o bloco de código
Ele repete enquanto a expressão for verdadeira (true)

Curso de Introdução ao Java

Controles de Fluxo do Programa

Laço do / while( )

```
public class LacoWhile {
  public static void main( String[] args ) {
    int i = 0;
    //laço do / while() com bloco de código definido
    do {
      System.out.println( "Linha: " + i );
      i++;
    } while( i < 10 );
  }
}
```

O bloco é executado ao menos um vez
Após a primeira repetição é que a expressão é avaliada

Curso de Introdução ao Java

Controles de Fluxo do Programa

Laço for( )

A sua estrutura é definida como a seguir:

```
for( iniciação;  condição;  incremento ) {
     bloco_de_código_a_executar
}
```

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

Curso de Introdução ao Java

Controles de Fluxo do Programa

Laço for( ) avançado (Enhanced for loop)

Foi definido a partir do Java 5, com o intuito de facilitar a vida do desenvolvedor, economizando código e evitando erros ao percorrer arrays e coleções (implementações de java.util.Collection).

É similar ao ***for each*** de outras tecnologias.

Não é possível controlar o índice utilizado pelo for, mas pode-se contonar este problema.

```
public class LacoForAvancado {
  public static void main( String[] args ) {
    for( String s : args ) {
      System.out.println("Argumento: " + s );
    }
  }
}
```

```
List lista = new ArrayList();
// adiciona itens à lista
for( String s : lista ) {
  System.out.println( s );
}
```

Curso de Introdução ao Java

Controles de Fluxo do Programa

Cláusula break

Aborta a execução de um laço, quando executado.

```
public class ClausulaBreak {
  public static void main( String[] args ) {
    char letras[] = { 'A', 'B', 'C', 'D', 'E' };
    int i;
    for( i=0; i<letras.length; i++ ) {
      if( letras[i] == 'C' ) {
        break;
      }
    }
    System.out.println( "Último índice: " + i );
  }
}
```

Curso de Introdução ao Java

Controles de Fluxo do Programa

Cláusula break rotulada

Aborta a execução de um laço rotulado, quando executado.

```
int j = 0, i = 0;
 principal1: while( true ) {
   for( i=0; i<1000; i++ ) {
     if( j == 10 && i == 100 )
       break principal1;
   }
   j++;
 }
```

Controles de Fluxo do Programa

Cláusula continue

Ignora a execução dos comandos seguintes do bloco, no laço, quando executado.

```
public class ClausulaContinue {
  public static void main( String[] args ) {
    char letras[] = { 'B', 'X', 'R', 'A', 'S', 'I', 'L' };
    int i;
    for( i=0; i<letras.length; i++ ) {
      if( letras[i] == 'X' ) {
        continue;
      }
      System.out.print( letras[i] );
    }
  }
}
```

Curso de Introdução ao Java

Controles de Fluxo do Programa

Cláusula continue rotulada

Ignora a execução dos comandos seguintes do bloco, do laço rotulado, quando executado.

```
int i=0, j=0;
principal2: for( j=1; j<10; j++ ) {
  for( i=1; i<10; i++ ) {
    if( (i % j) == 0 ) {
      System.out.println( "i=" + i + " j=" + j );
      continue principal2;
    }
  }
  j++;
}
```

Curso de Introdução ao Java

Exercícios

18) Faça o cálculo do valor da variável *x* mais *y*, inteiros, sem utilizar o operador +.

19) Verifique o valor de *x*, imprimindo uma mensagem correspondente quando for maior, menor ou igual a 10, sem usar a cláusula *if( )*.

20) Faça a soma de todos os valores (inteiros) entrados como argumento do programa e exiba na tela a mensagem: "A soma dos valores é = <valor>".

Curso de Introdução ao Java

Programação Orientada ao Objeto

O paradígma da Orientação ao Objeto é um mecanismo que ajuda a definir a estrutura de programas, baseado nos conceitos do mundo real, sejam eles reais ou abstratos.

A Orientação ao Objeto permite criar programas componentizados, separando as partes do sistema por responsabilidades e fazendo com que essas partes se comuniquem entre sí, por meio de mensagens.

Os conceitos da OO envolvem: Classes, Objetos e seus Relacionamentos, Herança e Polimorfismo.

Dentre as vantagens que a OO proporciona, podemos destacar o aumento de produtividade, reuso de código, redução das linhas de código programadas, separação de responsabilidades, encapsulamento, polimorfismo, componentização, maior flexibilidade do sistema, dentre outras vantagens.

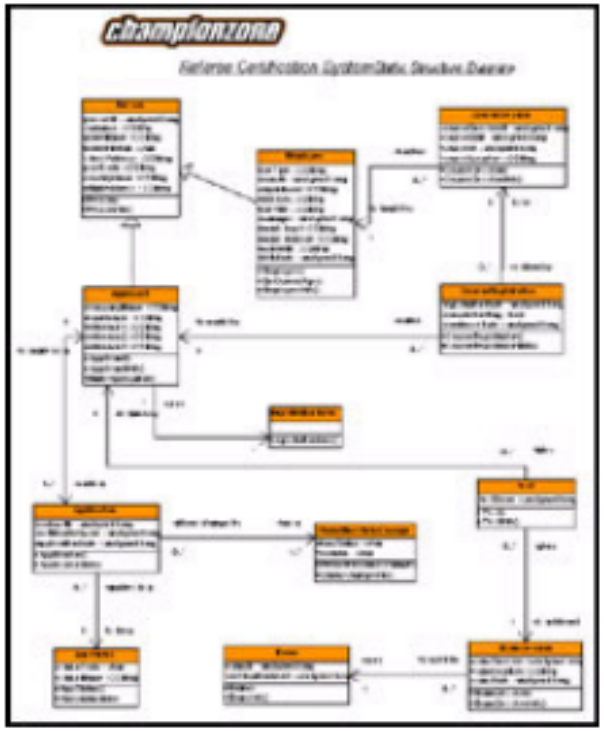
Modelagem Orientada ao Objeto

Os sistemas OO podem ser modelados com auxílio da UML (*Unified Modeling Language*).

UML é uma linguagem de modelagem para especificar, modelar, visualizar e documentar sistemas OO e não-OO, baseando-se em diagramas.

A UML é composta por:

- ✓ Diagrama de Classes
- ✓ Diagrama de Seqüência
- ✓ Diagrama de Objetos
- ✓ Diagrama de Casos de Uso
- ✓ outros....



Curso de Introdução ao Java

Classes

Uma classe, nada mais é, do que a descrição de um conjunto de entidades (reais ou abstratas) do mesmo tipo e com as mesmas características e comportamento. As classes definem a estrutura e o comportamento dos objetos daquele determinado tipo. Podemos dizer que as classes são, na verdade, modelos de objetos do mesmo tipo.

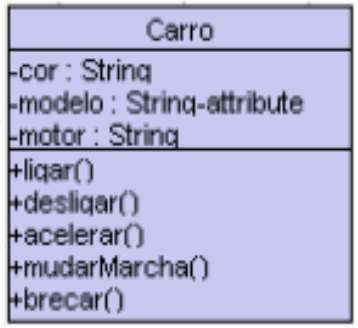
Entidade do mundo real

Propriedades:

- modelo
- cor
- motor

Comportamento:

- liga
- desliga
- muda marcha
- acelera
- breca



Modelo UML da classe

Curso de Introdução ao Java

Classes

```
public class Carro {
   String cor;
   String modelo;
   String motor;

   void ligar() {
     System.out.println( "Ligando o carro" );
   }

   void desligar() {
     System.out.println( "Desligando o carro" );
   }

   void acelerar() {
     System.out.println( "Acelerando o carro" );
   }

   void brecar() {
     System.out.println( "Brecando o carro" );
   }

   void mudarMarcha() {
     System.out.println( "Marcha engatada" );
   }
}
```

Ao lado temos o código Java da classe Carro, definida pelo modelo UML, com base no levantamento da entidade carro do mundo real.

No código definimos:

- declaração da classe
- declaração dos atributos
- declaração dos métodos

Arquivo: Carro.java

O arquivo do código-fonte sempre leva o nome da classe, seguido da extensão .java.

Curso de Introdução ao Java

Objetos

Um objeto, nada mais é do que uma instância particular de um tipo de dado específico (classe), ou seja, em outras palavras, objeto é uma entidade, do mundo computacional, que representa uma entidade do mundo real especificamente. O objeto criado fica armazenado em uma área de memória chamada *heap*.

Os Objetos possuem:

- Estado (atributos/propriedades)
- Comportamento (métodos/ações)
- Identidade (cada objeto é único)

Os Objetos se comunicam entre sim por meio de mensagens (chamadas aos métodos) e devem ter sua responsabilidade bem definida no sistema.

Criando uma instância (objeto) de uma classe:

```
Carro meuCarro = new Carro( );
```

Declaração da variável que vai guardar uma **referência para um objeto** do tipo Carro.

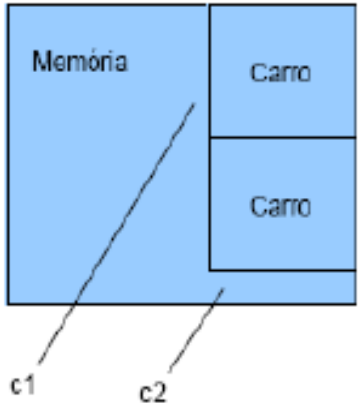
Operador **new** instancía o objeto.

Construtor da classe Carro

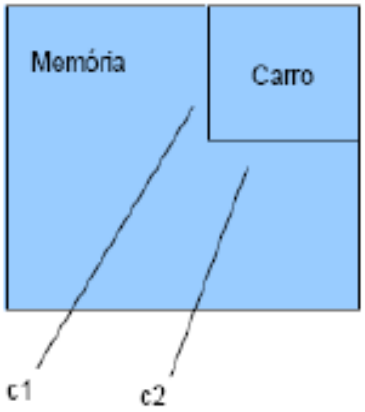
Objetos

As variáveis não guardam os objetos, mas sim uma referência para a área de memória onde os objetos estão alocados.

Se criarmos duas instâncias da classe Carro e atribuírmos cada instância para cada uma das duas variáveis diferentes, *c1* e *c2*, temos a situação ao lado.




```
Carro c1 = new Carro( );
Carro c2 = new Carro( );
```



Imagine, agora, duas variáveis diferentes, *c1* e *c2*, ambas referenciando o mesmo objeto. Teríamos, agora, um cenário assim:

```
Carro c1 = new Carro( );
Carro c2 = c1;
```

Curso de Introdução ao Java

Objetos

Utilizando a classe Carro:

```
class ExemploCarro {
  public static void main( String[] args ) {
    //criando uma instância da classe Carro
    Carro umCarro = new Carro();
    //atribuindo os valores dos atributos
    umCarro.modelo = "Gol";
    umCarro.cor = "preto";
    umCarro.motor = "1.0";
    //executando os métodos do objeto
    umCarro.ligar();
    umCarro.mudarMarcha();
    umCarro.acelerar();
    umCarro.brecar();
    umCarro.desligar();
    //atribuindo null para a variável diz que
    //agora ela não aponta para lugar nenhum
    umCarro = null;
  }
}
```

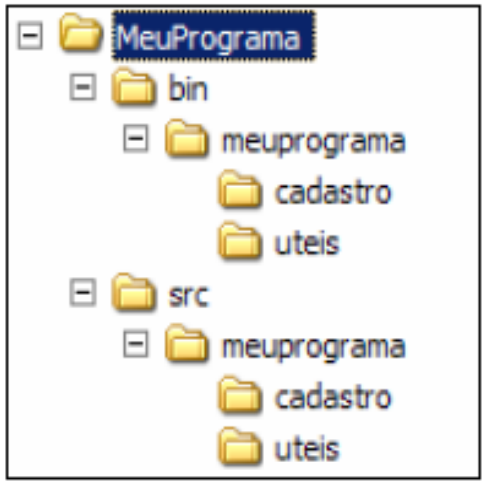
Package e Import

Package

Os pacotes (packages) servem para organizar e agrupar classes por funcionalidade.
Os pacotes são divididos por uma estrutura de diretórios.

```
package meuprograma.uteis;

public class ValidacaoCPF {
  //...
}
```

```
package meuprograma.cadastro;

public class CadastroUsuario {
  //...
}
```



Curso de Introdução ao Java

Package e Import

Import

O import deve ser usado para declarar que usaremos classes de outro pacote.
É parecido com o *include* de outras linguagens, como o C/C++, por exemplo.

```
package meuprograma.cadastro;

import meuprograma.uteis.ValidacaoCPF;
import java.sql.*; //importa classes JDBC

public class CadastroUsuario {
  //...
  public void cadastrar( Usuario u ) {
    //...
    if( ValidacaoCPF.validar( u.cpf ) ) {
      cadastrar( u );
    } else {
      throw new Exception("CPF Inválido");
    }
    //...
  }
}
```

```
package meuprograma.uteis;

public class ValidacaoCPF {
  public static boolean
     validar (String cpf) {
    //fazer a validação
  }
}
```

Curso de Introdução ao Java

Package e Import

Import estático

A partir do Java 5 é possível fazer o import estático, ou seja, dar import apenas nos métodos ou atributos estáticos oferecidos por uma classe, e usá-los como se fossem métodos ou atributos locais à sua classe.

```
import static java.lang.Math.*;

public class TesteImportEstatico {
  public static void main(String[] args) {
    double d = sin(1);
    // O método sin() pertence
    // à classe Math
  }
}
```

```
import static java.lang.Math.PI;

public class Calculos {
  public double areaCircunferencia(double r) {
    return PI * r * r;
  }
}
```

Curso de Introdução ao Java

Package e Import

Algo importante a se destacar é que as classes possuem um nome e um nome completo (ou *fully qualified name*).

O nome da classe é aquele informado na definição da classe. Exemplo:

```
public class MinhaClasse { }
```

Neste caso o nome da classe é MinhaClasse.

O nome completo da classe compreende o nome da classe mais a sua hierarquia completa de packages a qual pertence. Exemplo:

```
package meu.pacote;
public class MinhaClasse { }
```

Neste caso o nome completo da classe é meu.pacote.MinhaClasse.

Curso de Introdução ao Java

Atributos

Os atributos de uma classe são variáveis com o escopo do objeto.
São acessíveis e estão disponíveis enquanto o objeto estiver disponível.
Os atributos são iniciados durante a criação do seu objeto.

Durante a criação do objeto, os atributos de:

- tipos primitivos numéricos recebem 0 (zero) na iniciação;
- tipo *char* recebe o valor '\u0000';
- tipo *boolean* recebe false;
- referência a objetos recebem *null* na iniciação.

Porém, os atributos podem receber um valor padrão, definido na sua declaração, como no código abaixo:

```
class UmaClasse {
    String valorInicial = "um valor qualquer";
    int i = 1000;
}
```

Curso de Introdução ao Java

Atributos Estáticos

Atributos estáticos não precisam de uma instância da classe para serem usados
Eles são compartilhados por todas as instâncias da classe
Não são *thread-safe* (cuidado ao usá-los)

```
class Contador {
    static int count = 0;

    void incrementar() {
        count++;
    }
}
```

```
public static void main( String[] args ) {
    Contador c = new Contador();
    c.incrementar();
    System.out.println( Contador.count );
    Contador.count++;
    System.out.println( c.count );
}
```

Curso de Introdução ao Java

Constantes

Constantes são atributos de uma classe que não mudam de valor
O modificador *final* indica que o atributo é imutável

```
public class Matematica {
    static final double PI = 3.14159265;

    static double areaCircunferencia( double r ) {
        return PI * r * r;
    }

    static double perimetroCircunferencia( double r ) {
        return PI * r;
    }
}
```

Curso de Introdução ao Java

Métodos

A utilidade dos métodos é a de separar, em pedaços de códigos menores, uma determinada função.

É aconselhável criar e manter métodos pequenos, seguindo uma regrinha básica: Se o método tem scroll na tela, quebre-o em métodos menores. Isso facilita a leitura e o entendimento do código e a manutenção.

Regras para se criar métodos:

- Serem bem claros e ter uma função bem definida
- Serem pequenos e fáceis de entender
- Serem reaproveitáveis ao máximo

Curso de Introdução ao Java

Métodos

A sintaxe para a declaração dos método é a seguinte:

```
<tipo de retorno> <nome do método>( [lista dos atributos] ) {
    // implementação do método
}
```

O tipo de retorno informa qual o tipo de dados o método retorna. Pode ser um tipo primitivo ou um tipo de um classe. Caso o método não retorne nada, ele deve ser *void*.

O nome do método pode ser qualquer. Prefira seguir os padrões de nomenclatura e dar nomes significativos, de preferência verbos no infinitivo.

A lista de atributos não precisa ser informada se não há passagem de argumentos. Caso haja, os argumentos devem ser informados com seu tipo e nome, separados por vírgula se houver mais de um.

Retorno dos Métodos

A palavra reservada *return* causa o retorno do método.
Quando os métodos são declarados com o tipo de retorno *void*, então o método não pode e nem deve retornar nada.
Os métodos que retornam algum valor, devem retornar dados do tipo de retorno declarado, ou de tipos compatíveis.
Veja o exemplo:

```
public class TesteRetorno {
  public void naoRetornaNada() {
    int i = (int)(Math.random() * 100);
    if( i > 50 ) {
      return;  //aborta o método
    }
    System.out.println("OK");
  }

  int somar( int a, int b ) {
     return a + b;
  }

  Carro criarUmCarro() {
    Carro c = new Carro();
    c.modelo = "Ferrari";
    c.cor = "vermelha";
    c.motor = "5.0 V12";
    return c;
  }
}
```

Curso de Introdução ao Java

Métodos

```
public class DeclaracaoDeMetodo {
  public static void main( String[] args ) {
    DeclaracaoDeMetodo dm = new DeclaracaoDeMetodo();
    dm.fazerAlgo();
    dm.imprimirNaTela( "Daniel" );
    int soma = dm.somar( 2, 3 );
    Carro meuCarro = dm.criarUmCarro();
  }

  void fazerAlgo() {
    //este método não faz nada
  }

  void imprimirNaTela( String nome ) {
     System.out.println( "Meu nome é " + nome );
  }

  int somar( int a, int b ) {
     return a + b;
  }

  Carro criarUmCarro() {
    Carro c = new Carro();
    c.modelo = "Ferrari";
    c.cor = "vermelha";
    c.motor = "5.0 V12";
    return c;
  }
}
```

Curso de Introdução ao Java

Métodos Estáticos

Métodos estáticos não precisam de uma instância da classe para serem usados
Métodos estático NÃO podem chamar métodos não-estáticos sem uma instância
Não são *thread-safe* (cuidado ao usá-los)

```
class MetodoEstatico {
    public static void main( String[] args ) {
        MetodoEstatico me = new MetodoEstatico();
        me.metodoNaoEstatico();
        me.metodoEstatico();
        MetodoEstatico.metodoEstatico();
        metodoEstatico();
    }

    static void metodoEstatico() {
        //metodoNaoEstatico(); //ERRADO
        // (new MetodoEstatico()).metodoNaoEstatico();  //OK
    }

    void metodoNaoEstatico() {
        metodoEstatico();  //OK
    }
}
```

Construtores

Construtores não são métodos, são construtores.
Eles fazem a função de iniciação (start up) do objeto criado.
Se nenhum construtor for declarado, um construtor *default* será criado.
Múltiplos construtores podem ser declarados (overloading).

```
public class MinhaClasse {

  //sem construtor default

}
```

```
public class MinhaClasse {
  public MinhaClasse() {
    //Construtor Default
  }
}
```

```
public class NotaFiscal {
  private int numero;

  public NotaFiscal() {
    //Construtor Default
    this( novoNumero() );
  }

  public NotaFiscal( int numero ) {
    this.numero = numero;
  }

  public int novoNumero() {
    int i;
    //gera novo numero em i
    return i;
  }
}
```

Construtores

Usando os diferentes construtores:

```
public class Venda {

  public Venda() {
    //Construtor Default
 }

  public void fecharVenda() {
    //cria uma NF com um numero gerado
    NotaFiscal nf = new NotaFiscal();

    //cria uma NF com um numero definido
    NotaFiscal nf2 = new NotaFiscal( 12345 );
  }
}
```

Curso de Introdução ao Java

Exercícios

21) Crie classes que descrevam objetos que representem diferentes figuras geográficas, por exemplo: círculo, quadrado e retângulo.

22) Adicione métodos nas classes para calcular e retornar a área da própria figura.