Universidade Federal do Ceará
Centro de Ciências/Departamento de Computação
Código da Disciplina: CK0084 Ano: 2021
Professor: Ismayle de Sousa Santos

Aula
04, 05
e 06

Sistemas de Informações e Banco de Dados

Introdução à Programação Orientada a Objetos e Introdução ao Java

rfbrkh3 ismaylesantos@great.ufc.br @IsmayleSantos

Agenda

- Introdução à Programação Orientada a Objetos
  - Programação Estruturada vs Orientada a Objetos
  - Origem
  - Conceito
  - Objetos - Atributos e Métodos
  - Classes
  - Encapsulamento
  - Visibilidade

O que é Programação Estruturada?

- E uma programação desenvolvida por Michael A. Jackson no livro "Principles of Program Design" de 1975
- Exemplo
  - PHP, Cobol, C
- Essa programação preconiza que todos os programas possíveis podem ser reduzidos a apenas três estruturas:
  - Sequência
  - Decisão e
  - Iteração (repetição)

O que é Programação Estruturada?

- Sequência: Uma tarefa é executada após a outra, linearmente
- Decisão: A partir de um teste lógico, determinado trecho de código é executado, ou não
- Iteração: A partir de um teste lógico, determinado trecho de código é repetido por um número finito de vezes

Um programa é tipicamente escrito em uma única função

Quais as Vantagens e Desvantagens da Programação Estruturada?

- Vantagens
  - É fácil de entender!
  - Muito usada em cursos introdutórios de programação
  - Execução mais rápida
- Desvantagens
  - Baixa reutilização de código
  - Códigos confusos com dados misturados com comportamento

O que é Programação Orientada a Objetos?

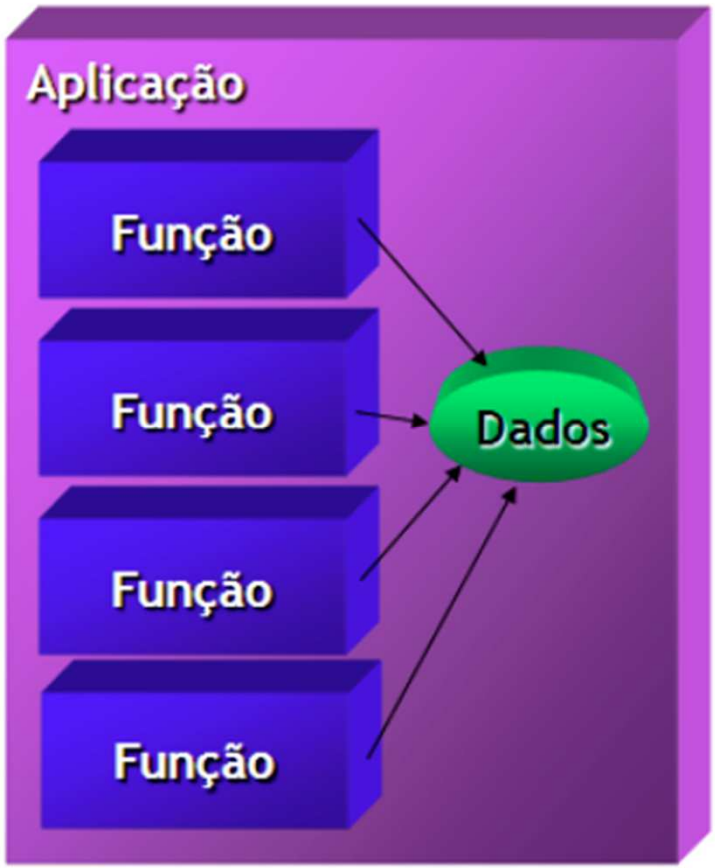
- A programação orientada a objetos (POO) é um modelo de programação onde diversas classes possuem características que definem um objeto

Quais as Vantagens e Desvantagens da POO?

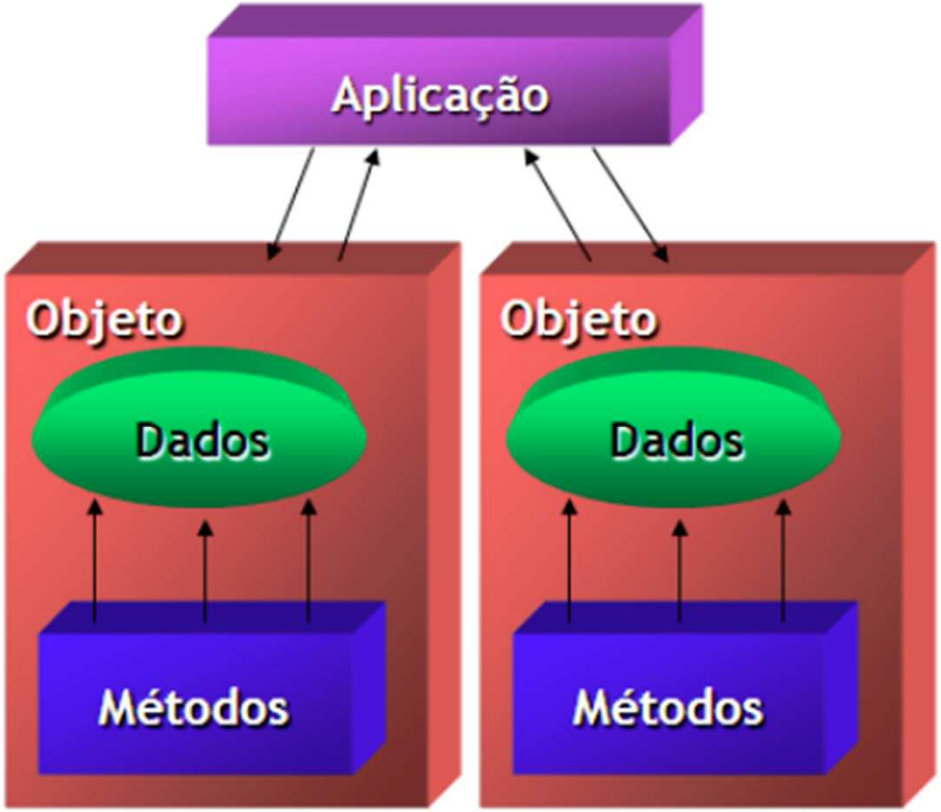
- Vantagens
  - Melhor organização do código
  - Bom reaproveitamento de código
- Desvantagens
  - Desempenho mais baixo que o paradigma estruturado
  - (Pode ser) Mais difícil a compreensão

Programação Estruturada vs POO

ESTRUTURADA



ORIENTAÇÃO A OBJETOS

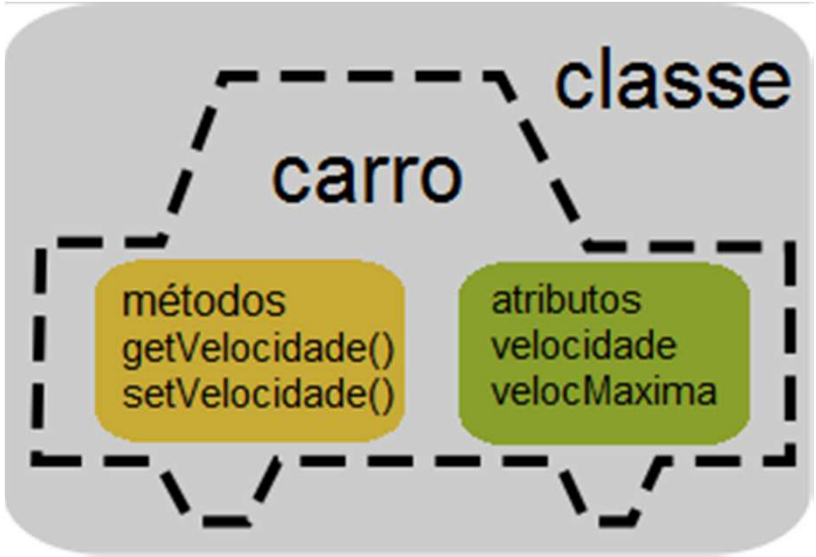


Qual a Origem da POO?

- Nos anos 70 surge Smalltalk, a primeira linguagem totalmente em Orientação a Objeto (O.O)
- C++, evolução de C, já possuía conceitos O.O
- Na década de 80 praticamente todas as linguagens já usavam conceitos O.O
  - Delphi
  - PASCAL
  - Java

Qual o Conceito por trás da POO?

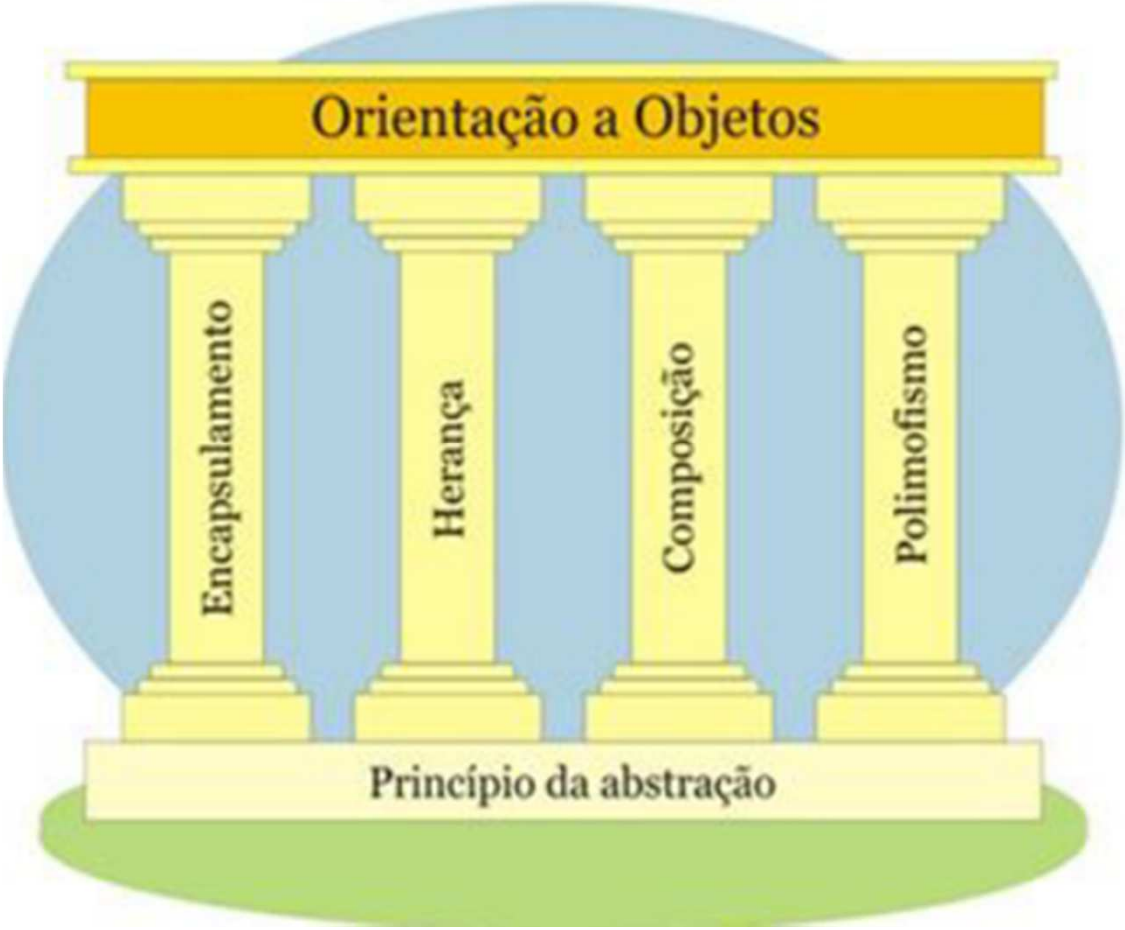
- James E. Rumbaugh afirma que a POO é "Uma nova maneira de pensar os problemas utilizando conceitos do Mundo Real. [...] O componente fundamental é o objeto que combina estrutura e comportamento em uma única entidade"



O que é uma POO?

- Paradigma de programação baseado no conceito de classes e objetos
  - As classes são elementos onde dados e procedimentos são agrupados, segundo seu objetivo, para um determinado sistema
  - Quando uma classe é usada como um tipo de dado para a criação de uma variável, esta é chamada de objeto

O que a POO Engloba?



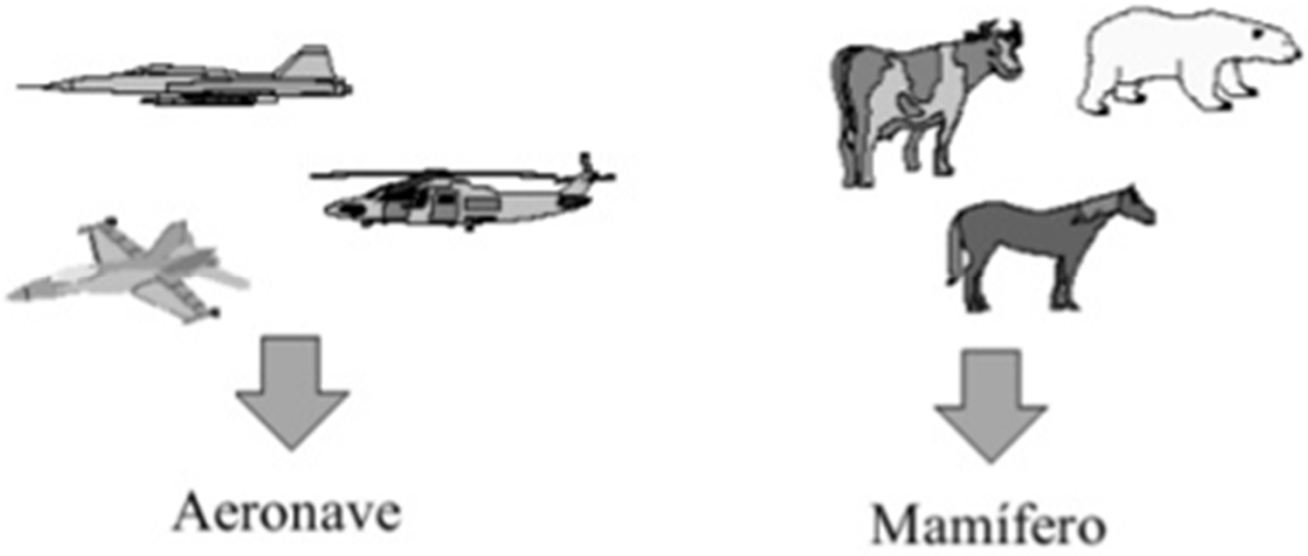
O que é o Princípio da Abstração?

- Habilidade de se concentrar nos aspectos essenciais do sistema, ou um contexto qualquer, ignorando o que é supérfluo
- Ou o processo de identificar os aspectos essenciais de um contexto qualquer, ignorando características menos importantes
  - A abstração é o resultado desse processo
- A abstração transforma aquilo que observamos no mundo real para a virtualidade

Classificar é uma forma de Abstração

- A abstração deve ser feita com algum objetivo para determinar o que é e o que não é importante

EXEMPLO:



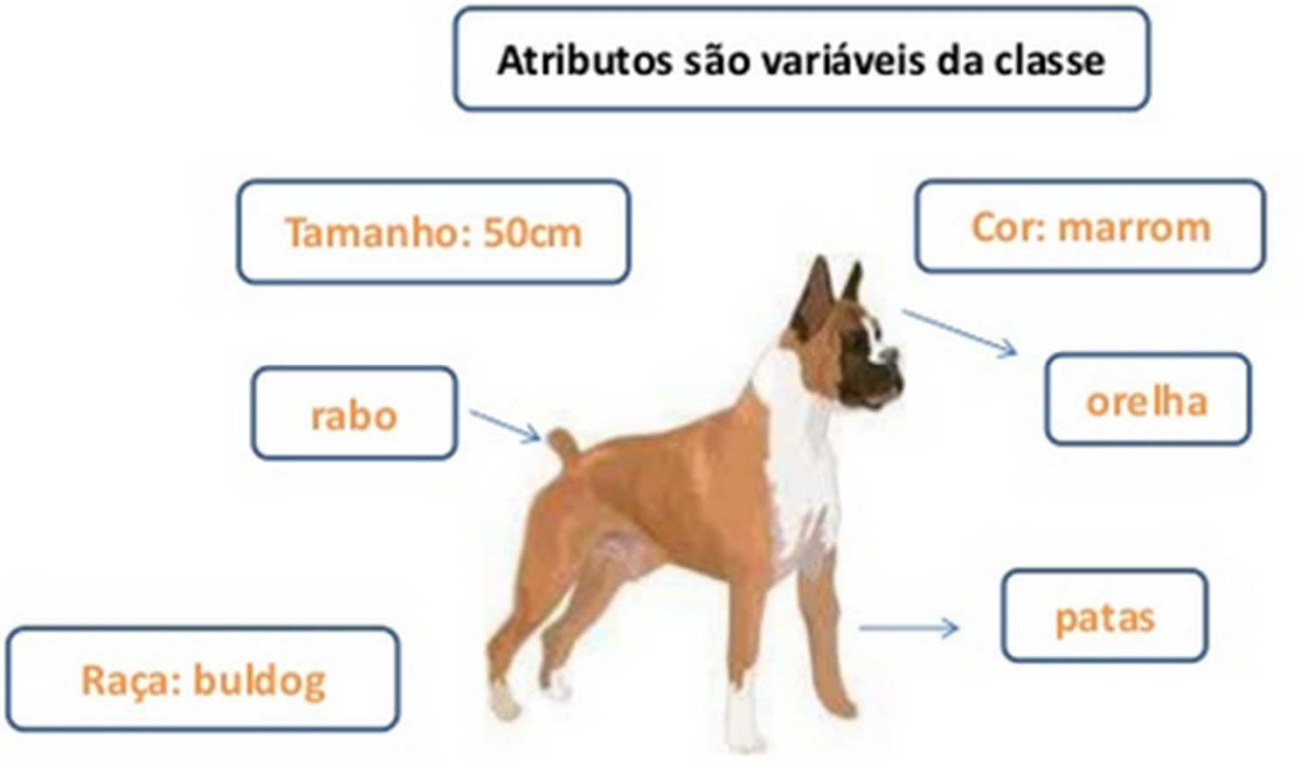
O que é um Objeto?

- A percepção dos seres humanos é dada através dos objetos
  - Um objeto é uma entidade que exibe algum comportamento bem definido
- É a representação computacional de algo do mundo real
  - Concreto = pessoas, avião, carro ...
  - Abstrato = música, operação bancária

O que é um Objeto?

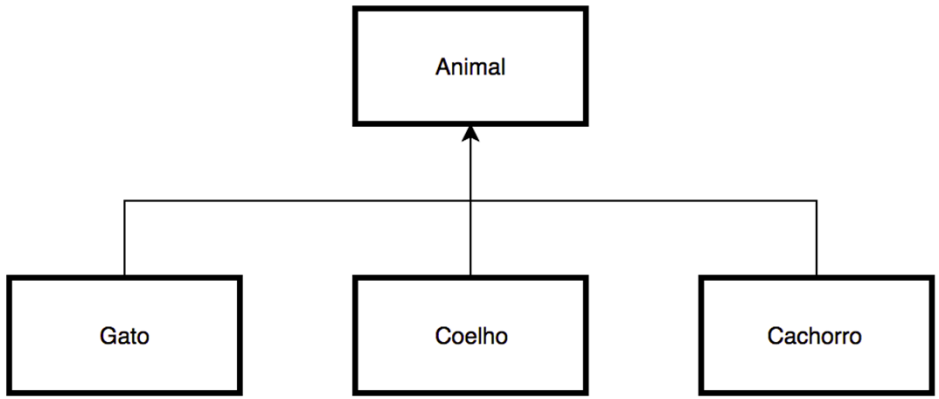
- Estado
  - Atributos (Características)
- Operações
  - Métodos (Comportamentos)
- Identidade
  - Dois objetos com estado e operações precisamente idênticos não são iguais
- Operações podem mudar os valores dos atributos assim mudando o estado de um objeto

Exemplo de Objeto

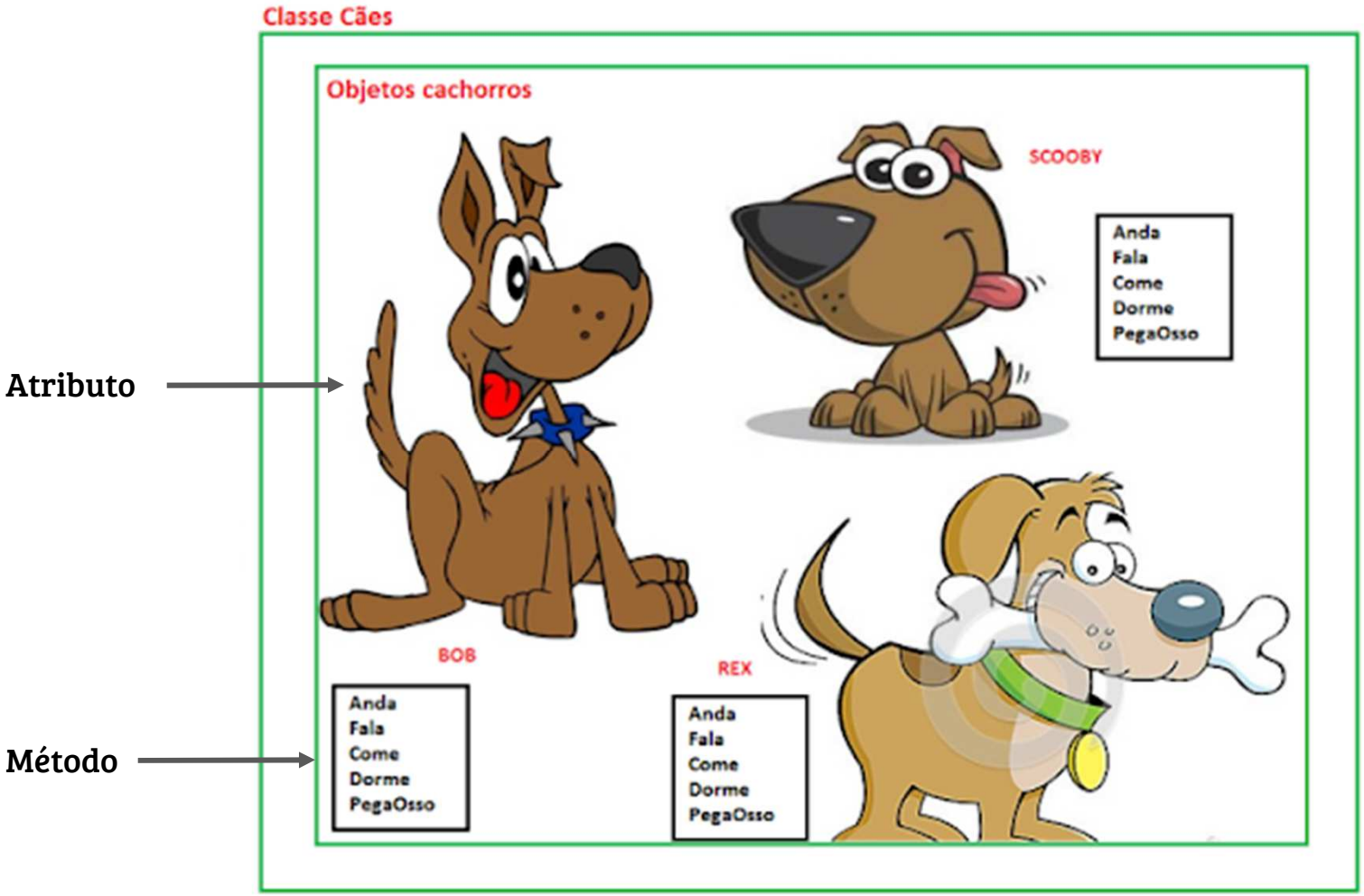


Como Definir um Objeto?

- Objetos parecidos têm a mesma classificação
  - Carro x, cor azul, 2 portas;
  - Carro y, cor verde, 4 portas;
  - Ambos são classificados como carro
- O conhecimento a determinado objeto é dado a partir de sua classificação



Como Definir um Objeto?



O que é um Método?

- São ações que uma classe possui

Exemplo de Métodos e Atributos

- Atributos
  - Raça: Poodle
  - Nome: Rex
  - Peso: 5 quilos
- Método
  - Latir
  - Comer
  - Dormir

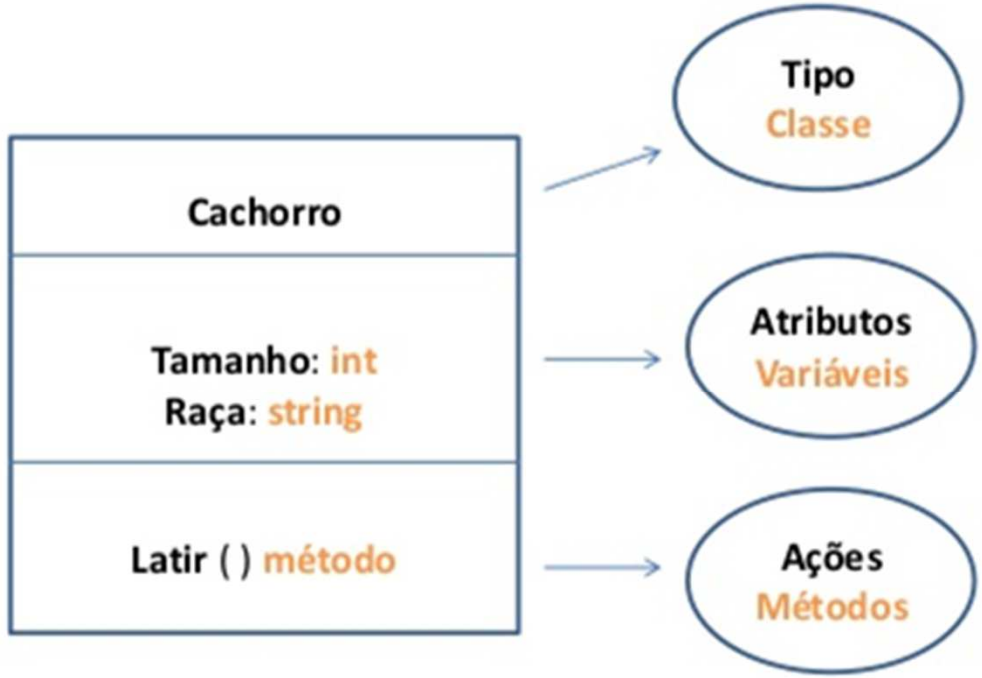
- Potência: 500cc
- Modelo: Honda
- Ano: 1998

- Acelerar
- Frear
- Abastecer

O que é uma Classe?

- É um conjunto de objetos
  - Características semelhantes
  - Comportamento comum
  - Interação com outros objetos
- Uma classe é a forma para criação de objetos
- Objetos são representações concretas (instâncias) de uma classe
- Uma classe serve de modelo para vários objetos semelhantes que possuem os mesmos tipos de informação em seu estado e tem os mesmos comportamentos

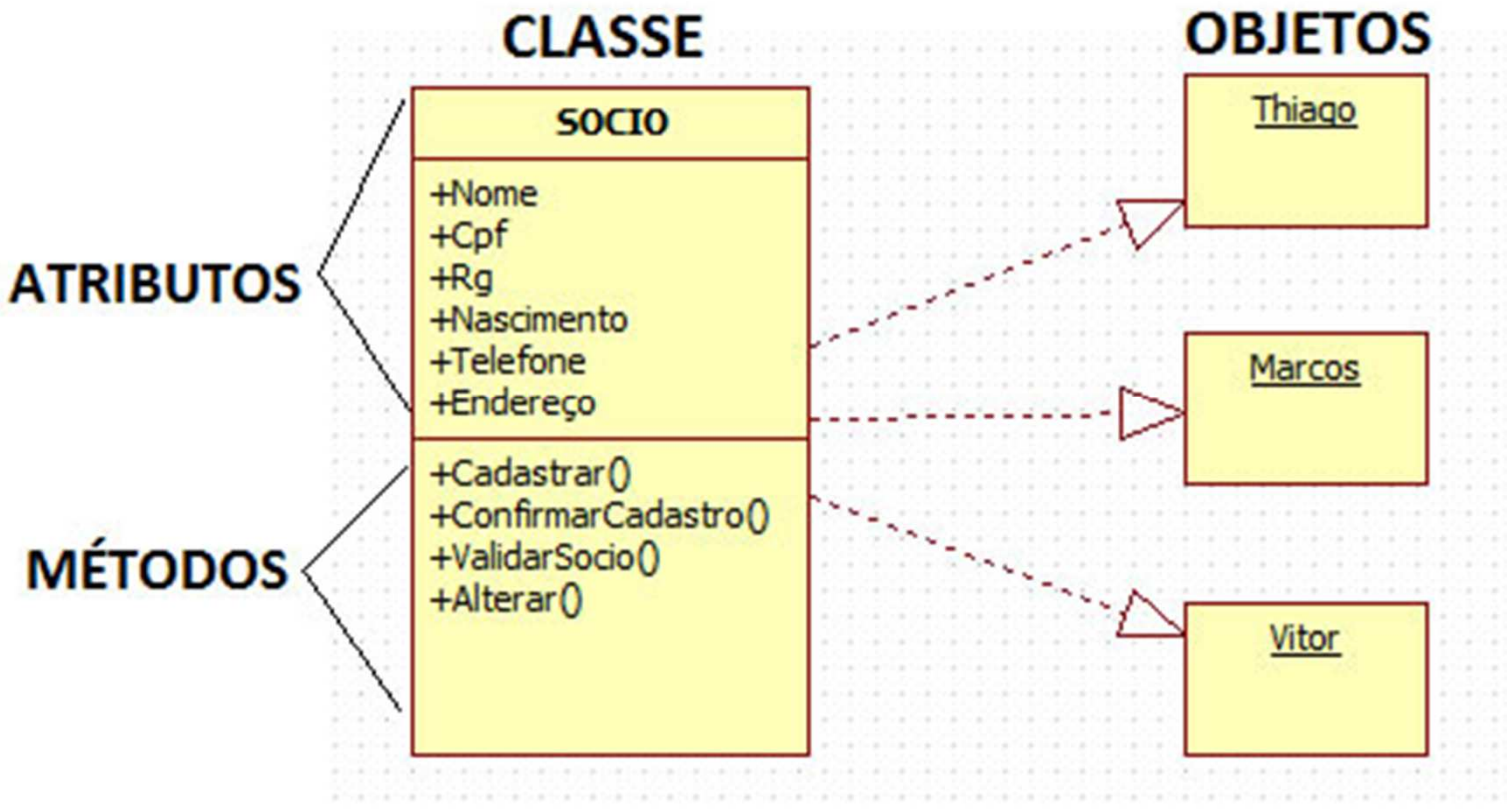
Exemplo de Classe



Sobre uma Classe ...

- É possível criar vários objetos em uma só classe
- Pode ser definido outros objetos com atributos diferentes comportamentos diferentes, mas do mesmo jeito não deixa de ser um objeto
- O conceito disso em orientação a objetos isso é chamado de código reuso, ou seja, reutilização de código
- Objetos trocam mensagem entre si, objetos trocam mensagem entre si e pode trocar atributo de outro objeto

Exemplo de Classe e Objeto



O que é Encapsulamento?

- Um objeto, em um programa, "encapsula" todo o seu estado e o comportamento
- Os dados e as operações são agrupados e a sua implementação é escondida, protegida dos usuários
- É a capacidade de restringir o acesso a elementos de uma classe utilizando qualificadores
  - Um qualificador ou modificador é uma palavra reservada que define a visibilidade de determinado atributo ou método

Exemplo de Encapsulamento

1) Classe Pessoa

```
public class Pessoa{
   private String nome;
   public String getNome(){
      return this.nome;
   }
   public void setNome(String nome){
      this.nome = nome;
   }
}
```

2) Classe Principal

```
public class Principal{
   public static void main(String[] args){
         Pessoa p = new Pessoa();
         p.setNome("Joaozinho");
         System.out.println("Nome :"  + p.getNome()  );
   }
}
```

Visibilidade

- Private (-)
  - Somente a classe tem acesso
  - Não é transmitido por herança
- Protected (#)
  - Visível em toda a classe de um pacote
  - Transmitido por herança
- Public (+)
  - Torna o membro acessível de fora da definição de classe
  - Visível irrestritamente

Visibilidade


```
            Classe Caneta
Publico     Modelo: Caractere
Publico     Cor: Caractere
Privado     Ponta: Real
Protegido   Carga: Inteiro
protegido   Tampada: Logico
Publico     Metodo rabiscar()
                        se (tampada) então
                escreva ("ERRO")
                         senão
                escreva ("RABISCO")
                        fimse
                FimMetodo
Privado     Metodo tampar()
                tampada=verdadeiro
                FimMetodo
            FimClasse
```

Agora vamos falar de Java

Na aula de hoje iremos aprender sobre Tipos primitivos, estruturas de controle, métodos e argumentos

Entendendo a Linguagem Java

- Java
  - É uma linguagem de programação orientada a objetos desenvolvida na década de 90
    - A história dela começa em 1991, quando um grupo de empregados da Sun Microsystems iniciaram o Projeto Green para pequenos dispositivos eletrônicos de consumo, tais como o PDA (Personal Digital Assistant)

Entendendo a Linguagem Java

- Java é tanto compilada como interpretada:
  - O compilador transforma o programa fonte em bytecodes
    - Bytecodes são instruções compreendidas pela Máquina Virtual Java
  - A Máquina Virtual Java (JVM) é um interpretador, que transforma as instruções em linguagem de máquina (permitindo a execução do programa)
  - "Write once, run anywhere" - slogan criado pela Sun, para demonstrar a portabilidade da linguagem (graças aos bytecodes)

Entendendo a Linguagem Java

- Como plataforma, Java compreende uma JVM e uma API (application programming interface)
  - Programas podem ser executados como aplicações tradicionais ou em páginas web
  - Applications - são executados pelo sistema operacional e podem ser:
    - console applications: quando não apresentam saída gráfica, somente textual
    - windowed applications: criam e gerenciam múltiplas janelas, usam mecanismos de GUI (graphical user interface) para a programação

Entendendo a Linguagem Java

- Applets - são programas executados pelo navegador Web, através de uma JVM própria (interna)
  - A característica principal dos applets é a utilização da própria área da página como interface
  - Applets são executados em um ambiente restrito, oferecendo segurança

Características da Linguagem Java

- Concisa e simples
  - Não contém redundâncias e é fácil de entender, implementar e usar
- Orientada a objetos
  - Suporta os principais conceitos de orientação a objetos e favorece reusabilidade
  - A abordagem de OO permite o desenvolvimento de sistemas de uma forma mais natural
- Provê acesso a Internet/WWW
  - Contém bibliotecas especiais que possibilitam o trabalho com protocolos TCP/IP como HTTP e FTP

Características da Linguagem Java

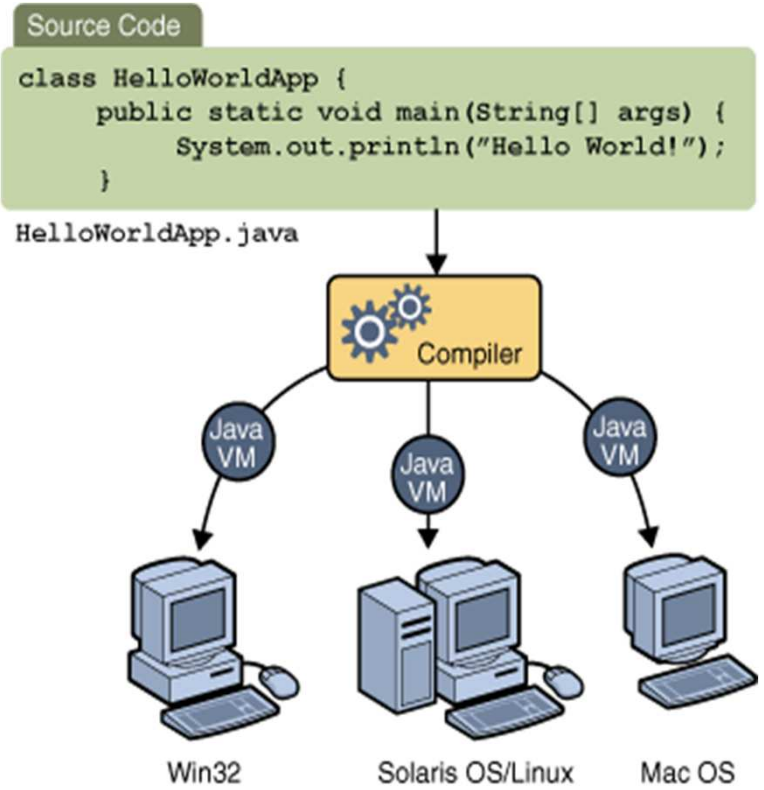
- **Robusta**
  - Reduz imprevistos em tempo de execução: variáveis são automaticamente inicializadas, uso disciplinado de ponteiros, rotinas devem ser chamadas corretamente, etc
- **Portável**
  - Não contém aspectos dependentes da implementação: o tamanho dos tipos é fixo para qualquer implementação, etc

Características da Linguagem Java

- Segura
  - Restrições de acesso a arquivos, manipulação de ponteiros, etc
- Concorrente
  - Suporta aplicações concorrentes: multithreads, monitores, execução atômica
- Também é: interpretada, neutra, portável, dinâmica e multi-thread

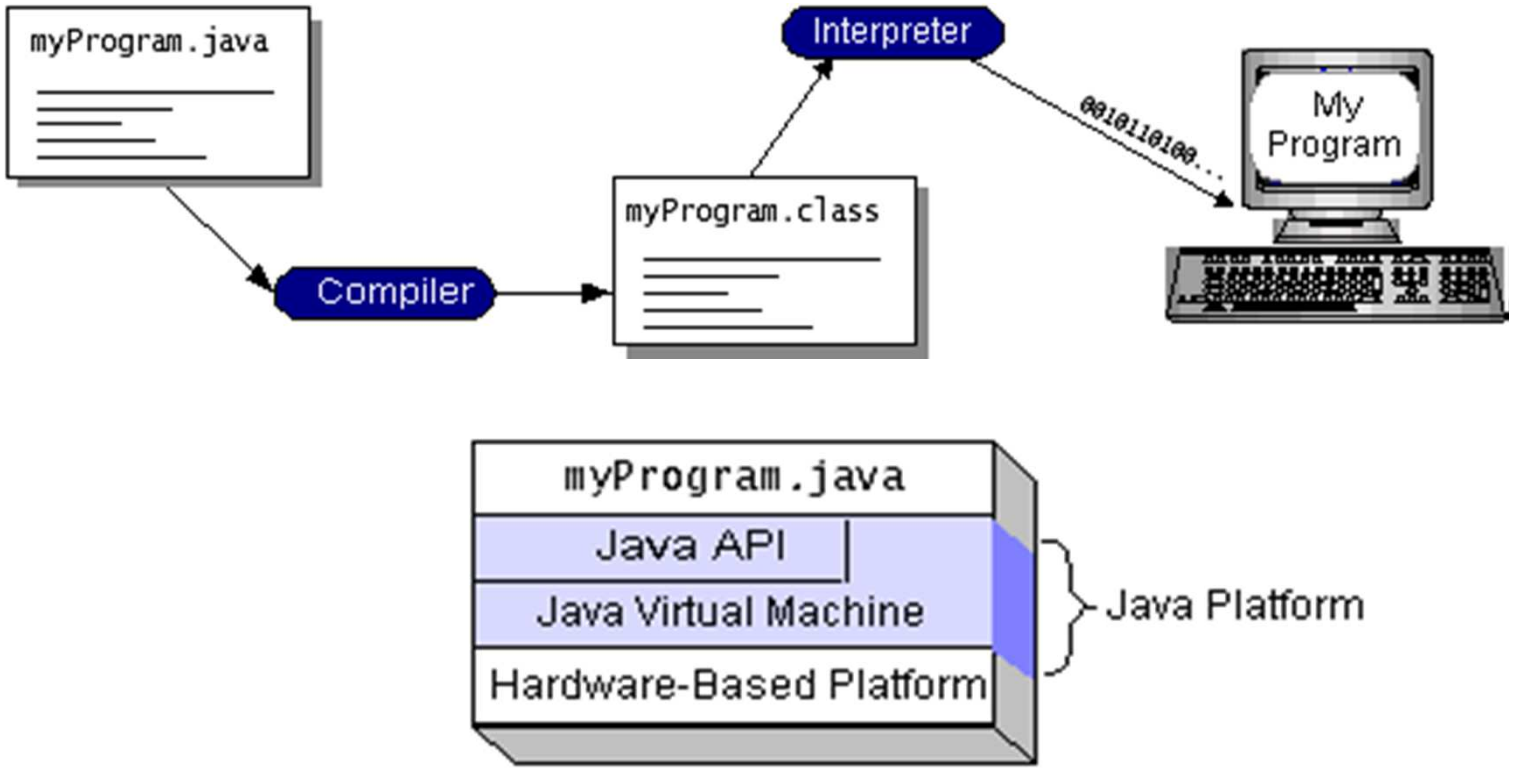
Interpretada, Neutra e Portável

- Bytecodes executam em qualquer máquina que possua uma JVM, permitindo que o código em Java possa ser escrito independente da plataforma
- A característica de ser neutra em relação à arquitetura permite uma grande portabilidade



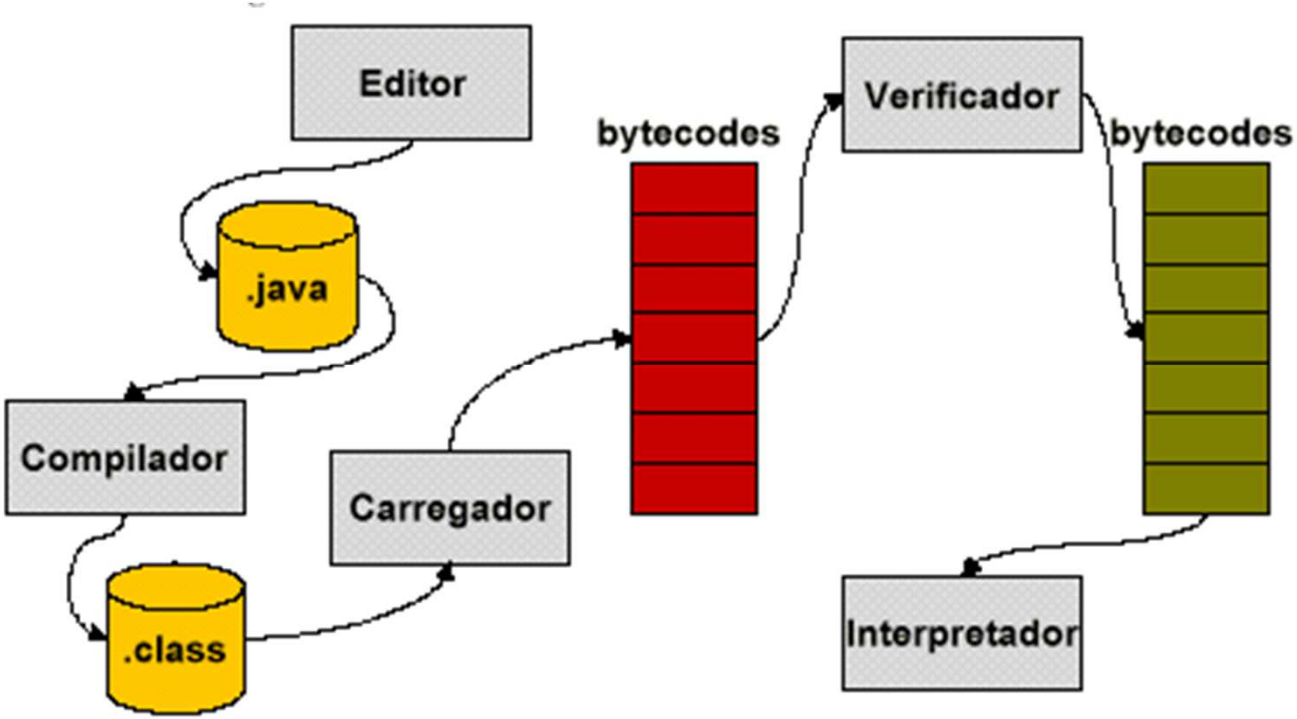
Fonte: MONTENEGRO, Anselmo. Introdução à Linguagem de programação Java - Aula de Programação de Computadores II. Niterói - RJ: Instituto de Computação UFF, s.a..

Interpretada, Neutra e Portável



Fonte: MONTENEGRO, Anselmo. Introdução à Linguagem de programação Java - Aula de Programação de Computadores II. Niterói - RJ: Instituto de Computação UFF, s.a..

O Ambiente Java



Fonte: MONTENEGRO, Anselmo. Introdução à Linguagem de programação Java - Aula de Programação de Computadores II. Niterói - RJ: Instituto de Computação UFF, s.a..

Ambiente de Desenvolvimento em Java

- Java possui um ambiente de desenvolvimento de software denominado Java SDK (Software Development Kit – antigamente denominado JDK)
    - É necessário instalar o kit para desenvolvimento de software Java, ou JDK (Java Development Kit)
- Não é um ambiente integrado de desenvolvimento, não oferecendo editores ou ambiente de programação
- O Java SDK contém um amplo conjunto de APIs (Application Programing Interface)

Ambiente de Desenvolvimento em Java

- Algumas ferramentas do Java SDK:
  - Compilador Java (javac)
    - Gera bytecodes a partir de código-fonte
  - Interpretador de aplicações Java (java)
    - interpreta (ou compila, se suportar JIT) os bytecodes para linguagem de máquina
  - Interpretador de applets Java (appletsviewer)

Ambiente de Desenvolvimento em Java

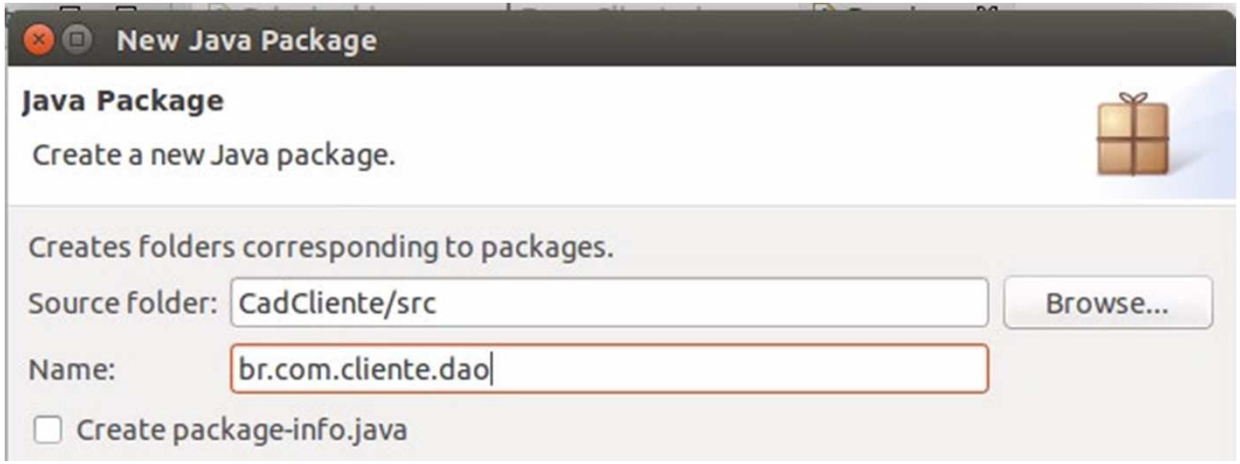
- Ainda existe o:
  - javadoc (um gerador de documentação para programas Java)
  - jar (o manipulador de arquivos comprimidos no formato Java Archive)
  - jdb (um depurador de programas Java), entre outras ferramentas

Estrutura de um programa em Java

- Um programa é composto por uma ou mais classes
- Tipicamente, cada classe é escrita em um arquivo fonte separado, cujo nome deve ser o mesmo da classe, com o sufixo .java
    - Exemplo: a classe “Pilha” deverá estar armazenada no arquivo Pilha.java
- Em geral, todas as classes que compõem um programa deverão estar no mesmo diretório

Packages

- Um pacote ou package na tecnologia Java nada mais é do que um conjunto de classes localizadas na mesma estrutura hierárquica de diretórios
- Com o uso de pacotes podemos organizar de forma física algo lógico (um grupo de classes em comum) que serão armazenados fisicamente em uma pasta



Packages

- Para indicar que as definições de um arquivo fonte Java fazem parte de um determinado pacote, a primeira linha de código deve ser a declaração de pacote:
    - `package nome_do_pacote`
- Caso tal declaração não esteja presente, as classes farão parte do "pacote default", que está mapeado para o diretório corrente

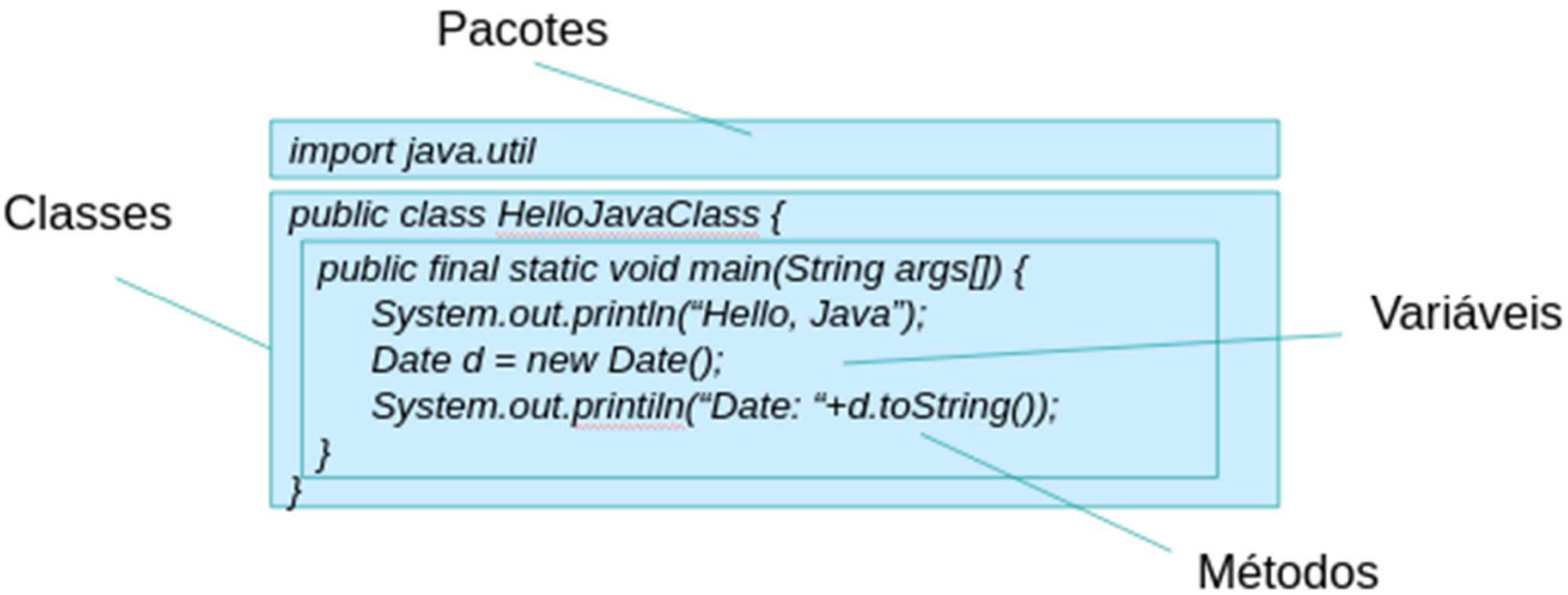
Packages

- Referenciando uma classe de um pacote no código fonte:
    - import nome_do_pacote.xyz ou simplesmente
    - import nome_do_pacote*
- Com isso a classe Xyz pode ser referenciada sem o prefixo nome_do_pacote no restante do código
- A única exceção refere-se às classes do pacote java.lang

Por hoje é só, próxima aula continuamos com JAVA ;)

Os Programa em Java possuem ...

- Todos os programas em Java possuem quatro elementos básicos:



Fonte: MONTENEGRO, Anselmo. Introdução à Linguagem de programação Java - Aula de Programação de Computadores II. Niterói - RJ: Instituto de Computação UFF, s.a..

Tipos Primitivos

- Uma variável de tipo primitivo armazena exatamente um valor de seu tipo declarado por vez
- Quando um outro valor é atribuído a uma dessas variáveis, seu valor anterior é substituído
- Os tipos primitivos são tipos de dados especiais internos à linguagem, não sendo objetos criados a partir de uma classe

Tipos Primitivos: Números Inteiros

- Os números inteiros se diferem nas precisões e podem ser positivos ou negativos

Tipo	Tamanho (bits)	Faixa	Valor Padrão
byte	8	-128 a 127	0
short	16	-32.768 a 32.767	0
int	32	-2^{31} a $2^{31} - 1$	0
long	64	-2^{63} a $2^{63} -1$	0L

Tipos Primitivos: Números de Ponto Flutuante

- Os números reais em ponto flutuante são iguais aos inteiros e também se diferem nas precisões podendo ser positivos ou negativos

Tipo	Tamanho (bits)	Faixa
float	32	IEEE 754 ±1,40129846432481707e-45 a 3,40282346638528860e+38
double	64	IEEE 754 ±4,94065645841246544e-324 a 1,79769313486231570e+308

Tipos Primitivos: Números de Ponto Flutuante

- As variáveis do tipo double armazenam valores com maior magnitude e precisão do que as do tipo float, e devem ser preferivelmente empregadas quando a precisão do valor for um fator importante

	Tipo primitivo	Classe Wrapper	Subclasse
Ponto Flutante	float	Float	
	double	Double	

Tipos Primitivos: Boolean

- Não é um valor numérico, só permite armazenar um valor lógico com valores true ou false (verdadeiro ou falso), ocupando apenas 1 bit de espaço
- Valor padrão para o tipo boolean: false

TIPO	TAMANHO
boolean	1 bit

Tipos Primitivos: Caracteres – char

- O tipo char permite a representação de caracteres individuais
- Caracteres de controle e outros caracteres cujo uso é reservado pela linguagem devem ser usados precedidos por < \ >
- Pode-se armazenar um conjunto de caracteres usando um tipo especial de referência denominado String
- O Java permite colocar acento no nome da variável, mas não é uma boa prática de programação

Tipos Primitivos: Caracteres – char

\b	backspace
\t	tabulação horizontal
\n	newline
\f	form feed
\r	carriage return
\"	aspas
\'	aspas simples
\\	contrabarra
\xxx	o caráter com código de valor octal xxx, que pode assumir valores entre 000 e 377 na representação octal
\uxxxx	o caráter Unicode com código de valor hexadecimal xxxx, onde xxxx pode assumir valores entre 0000 e ffff na representação hexadecimal.

Usar códigos ou caractere entre aspas simples

Fonte: MONTENEGRO, Anselmo. Introdução à Linguagem de programação Java - Aula de Programação de Computadores II. Niterói - RJ: Instituto de Computação UFF, s.a..

Palavras Reservadas

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

- **Além dessas existem outras que embora reservadas não são usadas pela linguagem**

const	*future*	*generic*	*goto*	*inner*	*operator*
outer	*rest*	*var*	*volatile*		

Fonte: MONTENEGRO, Anselmo. Introdução à Linguagem de programação Java - Aula de Programação de Computadores II. Niterói - RJ: Instituto de Computação UFF, s.a..

Declaração de Variáveis

- Sintaxe das variáveis:
  - Tipo nome1 [, nome2 [, nome3 [..., nomeN]]];
- Exemplos de declaração de variáveis:
  - int i;
  - float total, preco;
  - byte mascara;
  - double valormedio;
  - private double valormedio;

Declaração de Métodos

- Sintaxe geral dos métodos:

[visibilidade do método] [tipo de retorno] nome_do_método (tipos parâmetros){

// código do método

}

```
public void printC() {
    char c = 'c';
    System.out.println("c");
}

public int soma(int a, int b) {
    return a+b;
}
```

Método Main

- 1º método executada pela JVM

Nomes em Java

- Embora não seja de uso obrigatório, existe a convenção padrão para atribuir nomes em Java, como:
    - **Nomes de classes** são iniciados por letras maiúsculas
    - **Nomes de métodos, atributos e variáveis** são iniciados por letras minúsculas
    - **Em nomes compostos**, cada palavra do nome é iniciada por letra maiúscula, as palavras não são separadas por nenhum símbolo.
- Mais detalhes:
    - https://www.oracle.com/java/technologies/javase/codeconventions-contents.html

- Projeto
- Pacotes
- Atributos
- Métodos
- Visibilidade

Comentários

- O comentários em java são declarações que não são executadas pelo compilador ou interpretador
- Os comentários podem ser usados para prover informações ou explanações sobre as variáveis, métodos, classes ou alguma alguma declaração
- Existem três tipos de comentários em java:
  - Comentário de linha única
  - Comentário de múltiplas linhas
  - Comentário de documentação

Comentários

- Para fazer um comentário de uma linha é necessário usar //

```
//Isso é uma única linha comentada
```

  - O comentário de linha única é usado para comentar somente uma linha
- Para comentários de múltiplas linhas é necessário usar /* comentário */ :

```
/*
Isso
é
um comentário de
múltiplas linhas
*/
```

Comentários de Documentação

- A documentação comentada é usada para criar API documentada
    - Para criar uma API documentada, você precisa usar a ferramenta javadoc.
- Para fazer um comentário de documentação é necessário usar a seguinte sintaxe:

```
/**
Isto
é
uma documentação
documentada
*/
```

Operadores Aritméticos

Operador	Significado	Exemplo
+	Adição	a + b
-	Subtração	a – b
*	Multiplicação	a * b
/	Divisão	a / b
%	Resto da divisão inteira	a % b
-	Sinal negativo (- unário)	-a
+	Sinal positivo (+ unário)	+a
++	Incremento unitário	++a ou a++
--	Decremento unitário	--a ou a--

Fonte: MONTENEGRO, Anselmo. Introdução à Linguagem de programação Java - Aula de Programação de Computadores II. Niterói - RJ: Instituto de Computação UFF, s.a..

Operadores Relacionais

Operador	Significado	Exemplo
==	Igual	a == b
!=	Diferente	a != b
>	Maior que	a > b
>=	Maior ou igual a	a >= b
<	Menor que	a < b
<=	Menor ou igual a	a >= b

Fonte: MONTENEGRO, Anselmo. Introdução à Linguagem de programação Java - Aula de Programação de Computadores II. Niterói - RJ: Instituto de Computação UFF, s.a..

Operadores Lógicos

Operador	Significado	Exemplo
&&	E lógico (*and*)	a && b
||	Ou Lógico (*or*)	a || b
!	Negação (*not*)	!a

Fonte: MONTENEGRO, Anselmo. Introdução à Linguagem de programação Java - Aula de Programação de Computadores II. Niterói - RJ: Instituto de Computação UFF, s.a..

- Comentários
- Operadores Aritméticos e Lógico
- Métodos com argumentos

Controle do Fluxo de Execução

- Controle de fluxo é a habilidade de ajustar a maneira como um programa realiza suas tarefas
    - Permitem modificar a ordem sequencial de execução
- Se não houvesse o controle de fluxo, um programa poderia executar apenas uma única seqüência de tarefas, perdendo a característica mais interessantes da programação: a dinâmica

Controle do Fluxo de Execução

- Podemos classificar os comandos aceitos pela linguagem Java em basicamente quatro categorias:

Comando	Palavras-chave
Tomada de decisões	if-else, switch-case
Laços ou repetições	for, while, do-while
Tratamento de exceções	try-catch-finally, trow
outros	break, continue, return

Tomada de Decisões

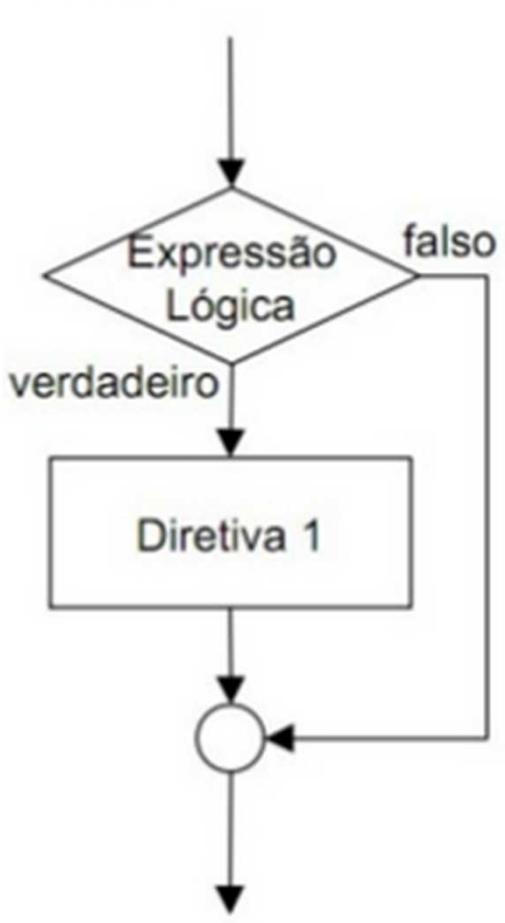
- As tomadas de decisões são baseadas no valor de uma variável
    - Quais partes do programa e quantas vezes serão executadas
- As tomadas de decisões englobam as seguintes estruturas de controle:
    - Execução Condicional
    - Execução Seletiva de Múltiplos Comandos
    - Execução Seletiva por Valores

Execução Condicional if-else

- A forma mais simples de controle de fluxo é o comando if-else
- Ele executa seletivamente ou condicionalmente um outro comando mediante um critério de seleção
- Esse critério é dado por uma expressão, cujo valor resultante deve ser um dado do tipo booleano (true ou false)
    - Se esse valor for true, então o outro comando é executado
    - Se for false, a execução do programa segue adiante

Execução Condicional if-else

- A declaração if especifica que uma instrução ou bloco de instruções seja executado se, e somente se, uma expressão lógica for verdadeira
- Quando existe apenas uma instrução após o if não precisamos abrir um bloco com as chaves



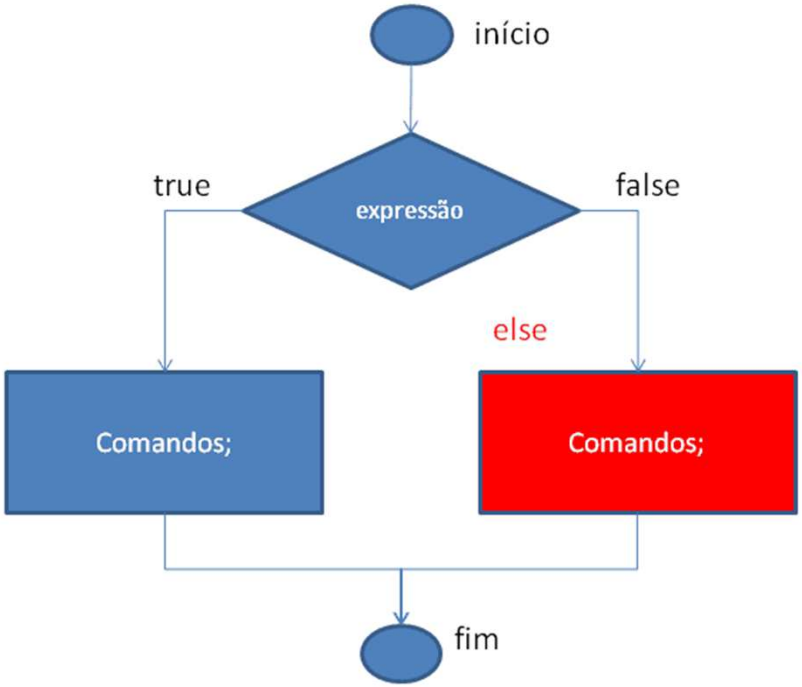

```
if (expressão_lógica)
    instrução;
```

ou

```
if (expressão_lógica) {
    instrução1;
    instrução2;
    ...
}
```

Execução Condicional if-else

- Se houver mais de uma condição usa-se o else



Fluxo da estrutura condicional if...else

```
if (expressão) {
 comando;
 comando;
 comando;
}
else {
 comando;
 comando;
}
```

Sintaxe if...else

Execução Seletiva de Múltiplos Comandos if-else-if

- A declaração else pode conter outra estrutura if-else
- Este cascateamento de estruturas permite ter decisões lógicas muito mais complexas
- A declaração if-else-if possui a seguinte sintaxe:

```
if (expressão_lógica1)
    instrução1;
else if(expressão_lógica2)
    instrução2;
else
    instrução3;
```

Execução Seletiva de Múltiplos Comandos if-else-if

- A presença do último else, juntamente com seu comando, é opcional
- Neste código, a [intrução1] será executada (e os demais saltados) caso a primeira condição seja true,a [intrução2] será executada (e os demais saltados) caso a primeira condição seja false e a segunda condição seja true, e assim sucessivamente
- A [instruçãoN] (se houver) somente será executado (e os demais saltados) caso todas as condições sejam false

Execução Seletiva por Valores switch-case

- Outra maneira de indicar uma condição é através de uma declaração switch
- A construção switch permite que uma única variável inteira tenha múltiplas possibilidades de finalização
- O comando switch pode ser executado com operando do tipo char, byte, short ou int

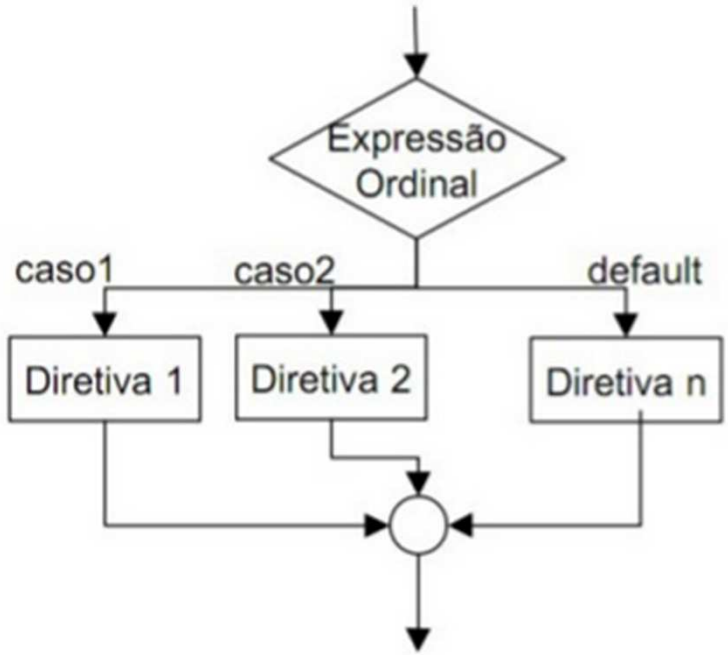
```
switch (variável_inteira) {
   case valor1:
      instrução1; //
      instrução2; // bloco 1
      ... //
      break;
   case valor2:
      instrução1; //
      instrução2; // bloco 2
      ... //
      break;
   default:
      instrução1; //
      instrução2; // bloco n
      ... //
      break;
}
```

Execução Seletiva por Valores switch-case

- Assim como no caso execução seletiva de múltiplos comandos, há situações em que se sabe de antemão que as condições assumem o valor true de forma mutuamente exclusiva
    - Apenas uma entre as condições sendo testadas assume o valor true num mesmo momento
- A [variável ou expressão] pode ser qualquer expressão válida

Execução Seletiva por Valores switch-case

- Se o valor for diferente de todas essas constantes, então o comando presente sob o rótulo default: é executado (e todos os demais são saltados), caso este esteja presente



Laço ou Repetição (Loop)

- Ao usarmos laços ou repetições uma tarefa é executada repetidamente por um programa enquanto uma dada condição seja verdadeira
- O Java apresenta as seguintes estruturas de loop:
    - while
    - do-while
    - for

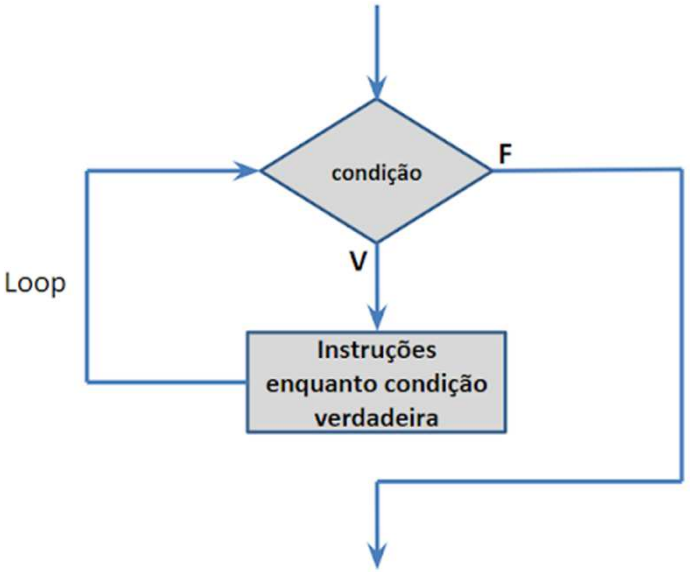
Operadores de incremento e decremento

Operadores
++
--

Comando while

- O termo while pode ser traduzido para o português como “enquanto”
- Este termo é utilizado para construir uma estrutura de repetição que executa, repetidamente, uma única instrução ou um bloco delas “enquanto” uma expressão booleana for verdadeira

```
while (condição)
{
Instrução ou bloco de instruções;
}
```



Comando while

- Em um laço while, a condição é testada antes da primeira execução das instruções que compõem seu corpo
- Se a condição for falsa na primeira vez em que for avaliada, as instruções desse laço não serão executadas nenhuma vez

(3 == 4) resulta em false

(3 != 4) resulta em true

Comando do-while

- A estrutura de repetição do-while é uma variação da estrutura while
- Em um laço do-while a condição somente é avaliada depois que suas instruções são executadas pela primeira vez, assim, mesmo que a condição desse laço seja falsa antes de ele iniciar, suas instruções serão executadas pelo menos uma vez

```
do {
    comando
} while (condição);
```

ou

```
do {
    comando1;
    comando2
    comando3;
} while (condição);
```

Diferença entre os Comandos while e do-while

while (c < a)
do {
a=a-1;
b=b+1;
c= c+b;
}
	do{
	a=a-1;
	b=b+1;
	c= c+b;
	}
	while (c<a)

Comando for

- O laço for é uma estrutura de repetição compacta
- Seus elementos de inicialização, condição e iteração são reunidos na forma de um cabeçalho e o corpo é disposto em seguida
- Quando o número de iterações de um loop é conhecido a priori, podemos usar uma forma mais simples de comando de repetição

```
for (inicialização; condição; incremento)
    instrução;
```

Comando for

- O comando for é executado do seguinte modo:
  - O valor de <início> é atribuído à variável i
  - Testa-se se i <= <fim>
  - Se for, o <bloco for> é executado, a variável i é incrementada de <incr> e volta-se ao passo 2
  - Se não for, o comando for termina

Curiosidade sobre o Comando for e while

- O laço for e o laço while são apenas formas diferentes de uma mesma estrutura básica de repetição
- Qualquer laço for pode ser transcrito em termos de um laço while e vice-versa
    - Do mesmo modo que em um laço while, se a condição de um laço for já é falsa logo na primeira avaliação que se fizer dela, as instruções contidas em seu corpo jamais serão executadas

Comando break

- O comando break é usado para interromper a execução de um dos laços de iteração ou de um comando switch
- Este comando é comumente utilizado para produzir a parada de um laço mediante a ocorrência de alguma condição específica, antes da chegada do final natural do laço
- E se isto se der dentro de um laço duplo?
    - O comando break provocará a interrupção apenas do laço em que o comando é imediatamente subjacente
    - Os outros laços continuam normalmente

- Controle do Fluxo de Execução

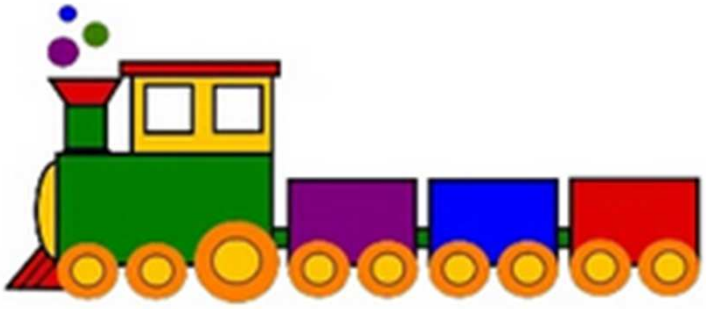
Arrays

- O propósito de um array é permitir o armazenamento e manipulação de uma grande quantidade de dados de mesmo tipo
- Exemplos de dados armazenados através de arrays:
    - Notas de alunos
    - Nucleotídeos em uma cadeia de DNA
    - Frequência de um sinal de áudio
- Arrays são especialmente importantes quando é necessário o acesso direto aos elementos de uma representação de uma coleção de dados

Fonte: MONTENEGRO, Anselmo. Introdução à Linguagem de programação Java - Aula de Programação de Computadores II. Niterói - RJ: Instituto de Computação UFF, s.a..

Arrays

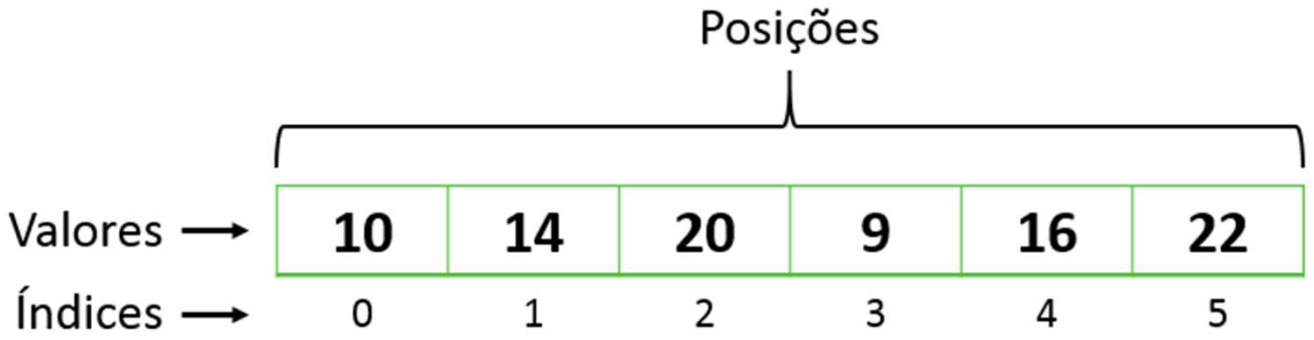
- Arrays são relacionados ao conceito matemático de função discreta, que mapeia valores em um conjunto finito de índices consecutivos em um conjunto qualquer de objetos de mesmo tipo
  - Por exemplo, um subconjunto de inteiros não negativos
- $F(x) \rightarrow S, x \in U$ tal que U é um conjunto finito de valores



Fonte: MONTENEGRO, Anselmo. Introdução à Linguagem de programação Java - Aula de Programação de Computadores II. Niterói - RJ: Instituto de Computação UFF, s.a..

Arrays Unidimensionais

- Os elementos de um array são identificados através de índices
- Arrays cujos elementos são indicados por um único índice são denominados arrays unidimensionais
- Um elemento em uma posição indicada por um índice i, em um array A, é acessado através do identificador do array seguido do índice i (entre chaves ou parênteses)



Fonte: MONTENEGRO, Anselmo. Introdução à Linguagem de programação Java - Aula de Programação de Computadores II. Niterói - RJ: Instituto de Computação UFF, s.a..

Arrays Unidimensionais em Java

- A criação de um array em Java requer 3 passos:
  - Declaração do nome do array e seu tipo
  - Criação do array
  - Inicialização de seus valores
- O número de elementos de um array em Java pode ser determinado através do nome do array seguido de .length( )
  - Exemplo: a.length()

Fonte: MONTENEGRO, Anselmo. Introdução à Linguagem de programação Java - Aula de Programação de Computadores II. Niterói - RJ: Instituto de Computação UFF, s.a..

Arrays Unidimensionais em Java

- Arrays em Java são objetos
- Arrays em Java tem índice base igual a zero
- Arrays em Java podem ser inicializados em tempo de compilação
- Exemplos:
  - string[ ] naipe = {“copas”,”ouros”, “paus”,”espadas”};
  - double[ ] temperaturas = {45.0,32.0,21.7,28.2,27.4};

Fonte: MONTENEGRO, Anselmo. Introdução à Linguagem de programação Java - Aula de Programação de Computadores II. Niterói - RJ: Instituto de Computação UFF, s.a..

Arrays Multidimensionais em Java

- Arrays multidimensionais representam agregados homogêneos cujos elementos são especificados por mais de um índice
- Em Java é muito simples especificar um array multidimensional
  - Exemplo: array contendo as notas de 3 provas de 30 alunos
    - `int[ ][ ] notas = new int[30][3];`

Fonte: MONTENEGRO, Anselmo. Introdução à Linguagem de programação Java - Aula de Programação de Computadores II. Niterói - RJ: Instituto de Computação UFF, s.a..

- Arrays

Obrigado!

Por hoje é só pessoal...

Dúvidas?

 rfbrkh3

 ismaylesantos@great.ufc.br

 @IsmayleSantos