Aula 01 - Introdução a Programação Java

Prof. Marcelo Nascimento Costa, MSc
marcelo.nascimento@uva.br

Estrutura do Módulo 1 – Java Básico

- Apresentação da Linguagem Java
- Introdução à Linguagem de Programação Java
- A linguagem Java
  - Declarações de classes e interfaces, classes abstratas; modificadores de visibilidade; uso de primitivos, enums, arrays e var-args.

Estrutura do Módulo Java

- Encapsulamento, herança, polimorfismo; overriding/overloading; construtores e instanciação; casting; métodos e variáveis estáticas; acoplamento e coesão.
- Uso de primitivos; passagem de parâmetros em Java; declaração, instanciação e inicialização de arrays; wrappers e autoboxing; garbage collection.
- Operadores de atribuição, relacionais, lógicos, condicionais, aritméticos, instanceof.

Apresentação da Linguagem Java

O que é Java ?

- Java
  - Linguagem de programação orientada a objetos
  - Definida pela *Sun Microsystems*
  - Projetada originalmente para controlar aparelhos eletrônicos

O que é Java?

- Atualmente chamamos Java de tecnologia ou plataforma, por causa de seu suporte a diversos tipos de sistemas de processamento.
  - JSE – **Java Standard Edition**, destinada a computadores pessoais e pequenos servidores.
  - JEE – **Java Enterprise Edition**, para sistemas de grande porte e servidores;
  - JME – **Java Micro Edition**, para sistemas de pequeno porte e embarcados;
  - **JavaCard**, para operação de SIM Cards ou chips passivos.

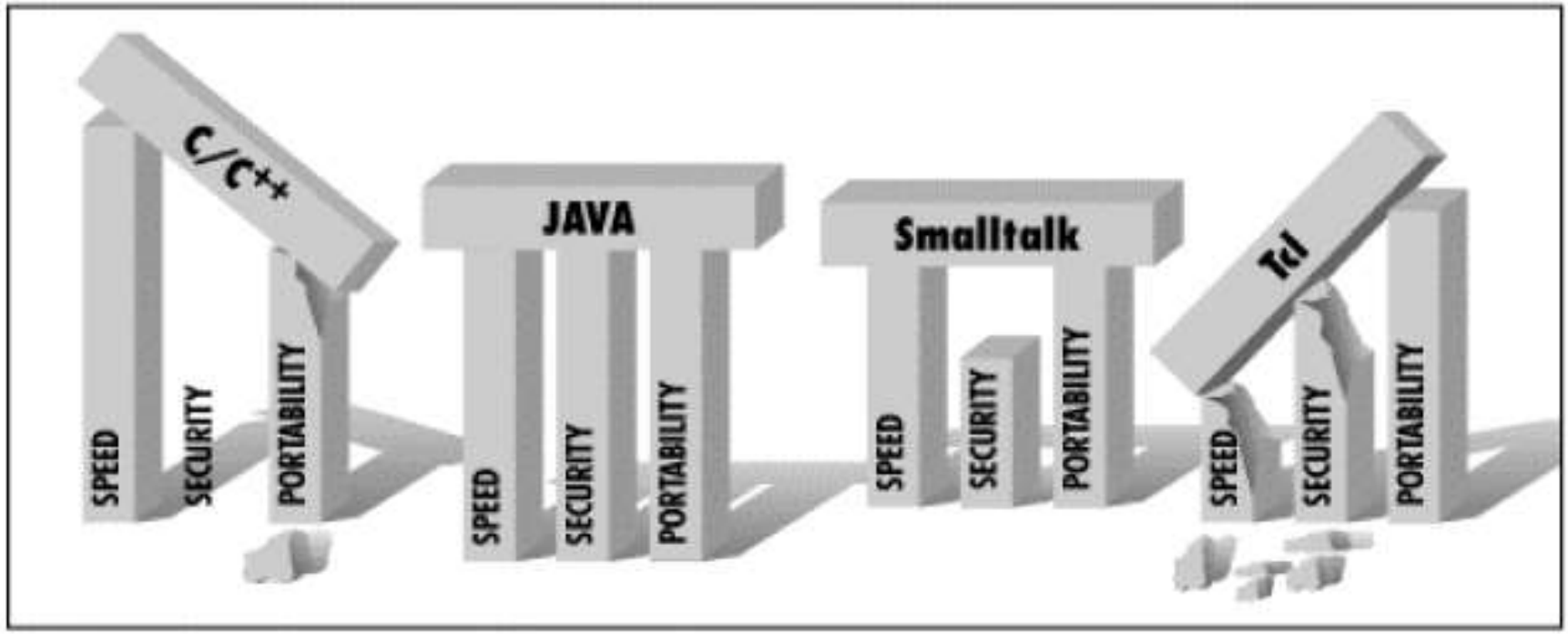
Características da Linguagem

- Orientada a Objetos
  - Java é pura, totalmente OO, e possui grande diversidade de bibliotecas de classes disponíveis;
- Simples
  - Java é mais simples que outras linguagens OO, como C++, e possui facilidades como "*Garbage Collector*";
- Distribuída
  - Suporta aplicações em rede e objetos distribuídos. Java também suporta aplicações multi-tarefa, podendo executar diversas *threads* simultaneamente;

Características da Linguagem

- Independente de Plataforma
  - Java é interpretada, podendo rodar em qualquer plataforma;
  - Para aumentar a segurança da linguagem, o interpretador analisa o código antes de executá-lo;
- Robusta
  - Java suporta o tratamento de exceções. O interpretador não permite que uma aplicação paralise o sistema;
- Performance
  - Mais rápida que linguagens *script*, porém mais lenta que as linguagens compiladas. Passível de compilação *just-in-time*.

Características da Linguagem



Introdução à Linguagem de Programação Java

Conceitos Fundamentais

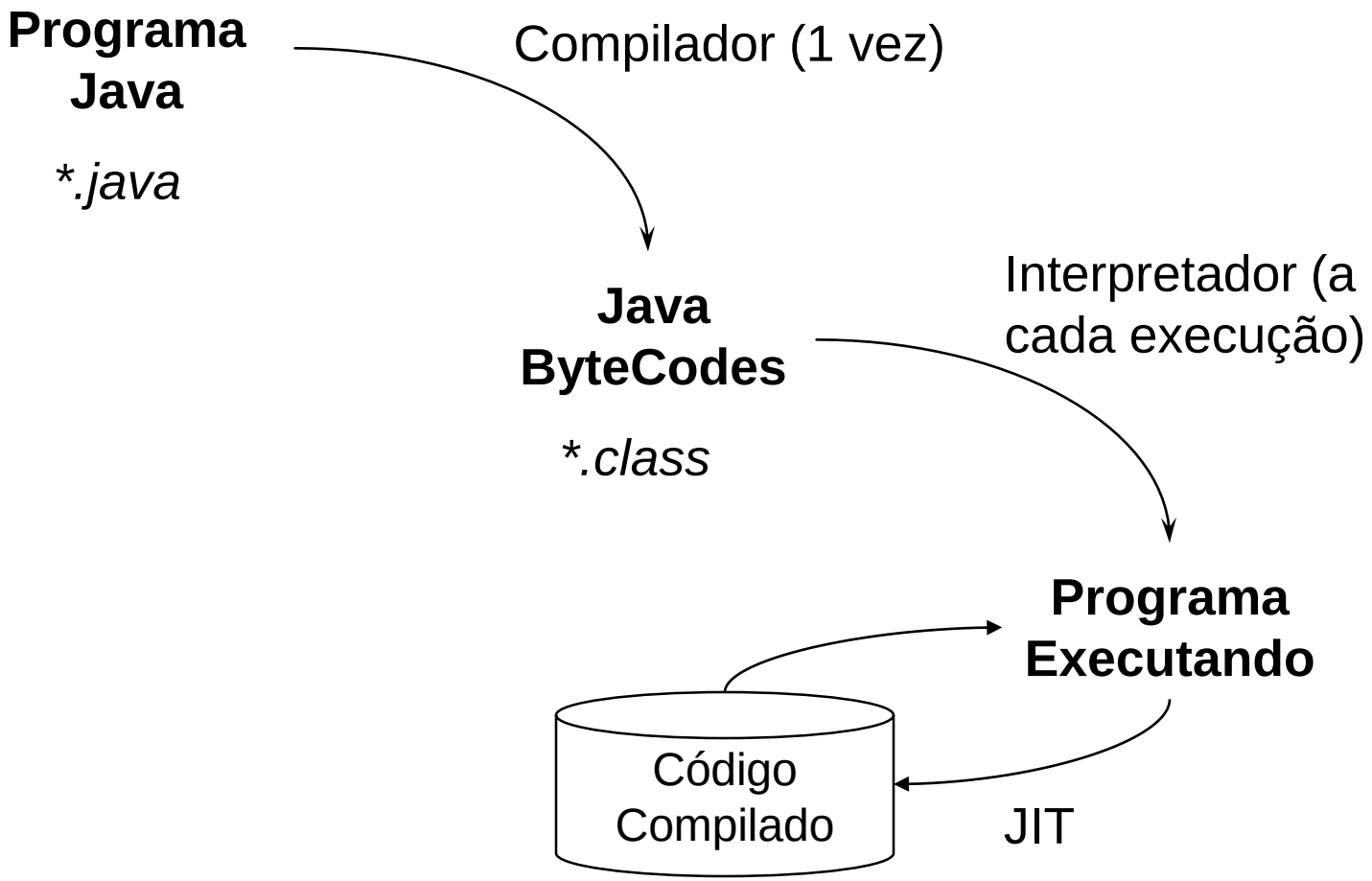
- Elementos básicos da linguagem
- Orientação a Objetos e Java
- Definindo Classes em Java
- Herança
- Polimorfismo e Interfaces
- Exceções
- Pacotes
- Estruturas de dados

Compilação

- Aplicações Java são facilmente portáveis
  - Programa Java pode ser executado em diversas plataformas
  - Programa compilado e interpretado
  - Compilação transforma o programa em *bytecodes*
  - *Bytecodes* são interpretados pela Java *Virtual Machine (VM)*

- *Java Virtual Machine*
  - *Browser Web*, no caso de *applets*
  - Interpretador "*stand-alone*", no caso de aplicações
  - Existem *Java VM's* para diversas plataformas

Compilação


Programa
Java
*.java
Compilador (1 vez)
Java
ByteCodes
*.class
Interpretador (a
cada execução)
Programa
Executando
Código
Compilado
JIT

Java ainda é APENAS interpretado? (1)

- As JVMs atuais executam bytecodes utilizando uma combinação de interpretação e a chamada **compilação Just-In-Time(JTI)**.

- Nesse processo, a JVM analisa os bytecodes à medida que eles são interpretados, procurando **hot spots(pontos ativos)** - partes dos bytecodes que executam com frequência.

- Para essas um (JIT) traduz os bytecodes para a linguagem de maquina de um computador subjacente.

- Quando a JVM encontra novamente essas partes compiladas, o código de linguagem de maquina mais rápido é executado.

Java ainda é APENAS interpretado? (2)

- Os programas java na realidade passam por duas fases de compilação:
  1. Código-fonte é traduzido em bytecodes (para a portabilidade entre JVMs em diferentes plataformas de computador)
  2. Durante a execução, os bytecodes são traduzidos em linguagem de maquina para o computador real em que o programa é executado.

Um Pequeno Programa Java

- Passos para criar uma aplicação Java
  - Edição do código fonte
  - Compilação *(javac AloMundo.java)*
  - Execução via interpretador *(java AloMundo)*

```
/* Meu primeiro programa Java */
/* Arquivo AloMundo.java */
class AloMundo {
        public static void main(String[] args) {
                System.out.println("Alô Mundo!");
        }
}
```

Tutorial NetBeans

- Acessar o endereço: https://netbeans.org/kb/docs/java/quickstart_pt_BR.html
- Executar o passo-a-passo da compilação do programa no NetBeans

Reescrita do Programa

- Escreva um programa em Java para imprimir seu nome na tela do computador. Compile e rode esse programa.

Comentários

- Modelo similar a C++

```
// comentário de linha
/* comentário de bloco */
/** comentário de bloco c/ propósito de
    documentação  */
```

- Documentação (Javadoc)
  - Padrão para geração automática (HTML)
  - Marca de início de documentação /**
  - Marca de fim de documentação */

Rotina Principal

- A rotina *main*
  - A rotina principal determina o início do programa Java
  - Um programa pode ser composto por diversas classes
  - Uma e somente uma classe deve definir o método *main*
  - O método *main* possui o seguinte formato:

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

  - O parâmetro **args** indica os argumentos do programa
  - Os argumentos são as palavras da linha de comando

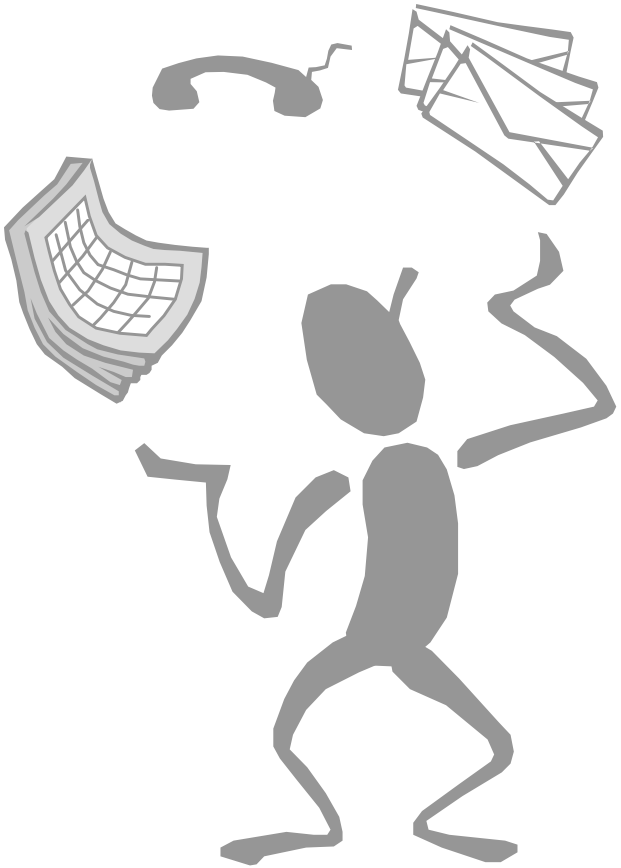
Entrada e Saída

- Classe System
  - Pertence a biblioteca de classes do Java
  - A classe System define os arquivos de entrada e saída padrão
  - O arquivo ***out*** representa a saída de vídeo
  - O arquivo ***in*** representa a entrada de teclado

Bibliotecas Java

- Java API
  - Coleção de classes de objetos
  - As classes são agrupadas em pacotes (*packages*)
  - As classes implementam diversas funcionalidades
    - Estruturas de dados
    - Interface gráfica
    - Comunicação
    - Bancos de dados
    - Segurança
  - Existem diversas versões da API e API's complementares

Orientação a Objetos e Java



- Tópicos
  - Programação OO
  - Objetos
  - Atributos
  - Métodos
  - Mensagens
  - Classes
  - Instanciação de objetos

Programação Estruturada

- Composição dos Programas
  - Um programa é composto por um conjunto de rotinas
  - A funcionalidade do programa é separada em rotinas
  - Os dados do programa são variáveis locais ou globais

- Fluxo de Execução
  - O programa tem início em uma rotina principal
  - A rotina principal chama outras rotinas
  - Estas rotinas podem chamar outras rotinas, sucessivamente
  - Ao fim de uma rotina, o programa retorna para a chamadora

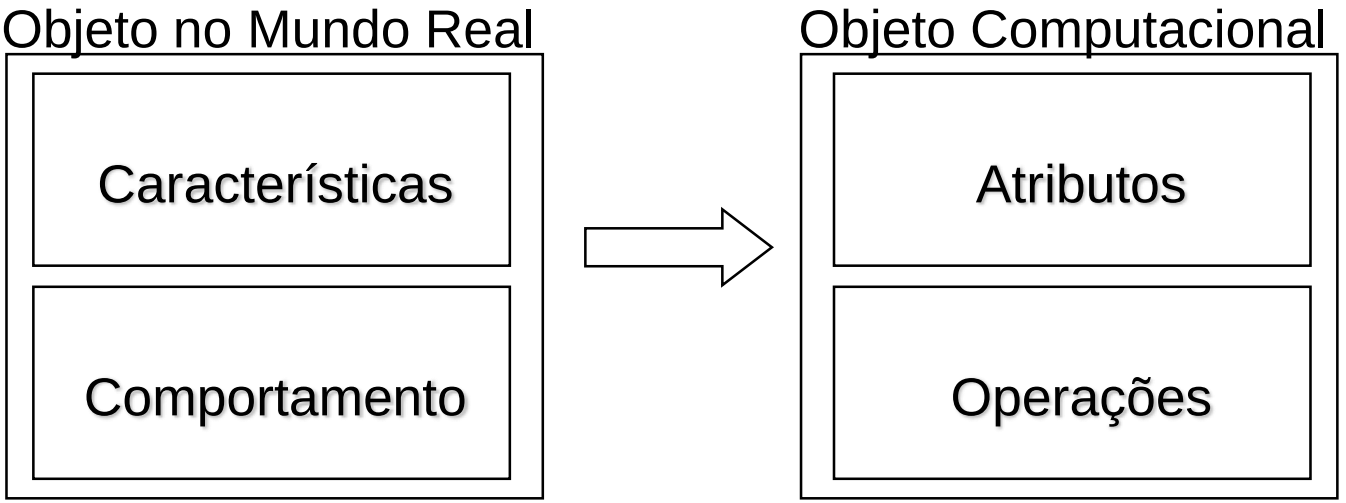
Programação OO

- Composição do programa
  - A funcionalidade do programa é agrupada em objetos
  - Os dados do programa são agrupados em objetos
  - Os objetos agrupam dados e funções correlacionados

- Fluxo de Execução
  - Similar ao anterior
  - Os objetos colaboram entre si para a solução dos objetivos
  - A colaboração se realiza através de chamadas de métodos de objetos.

Objetos

- Definição
  - Um objeto é a representação computacional de um elemento ou processo do mundo real
  - Cada objeto possui suas características e seu comportamento

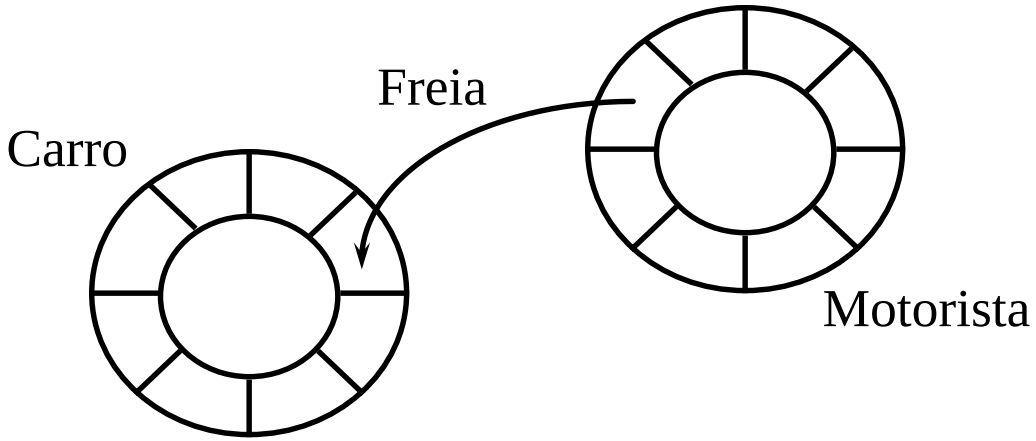


Classes

- A classe descreve as características e comportamento de um conjunto de objetos
  - Cada objeto possui uma única classe
  - O objeto possuirá os atributos e operações definidos na classe
  - O objeto é chamado de instância de sua classe
  - A classe é o bloco básico para a construção de programas OO

Mensagens

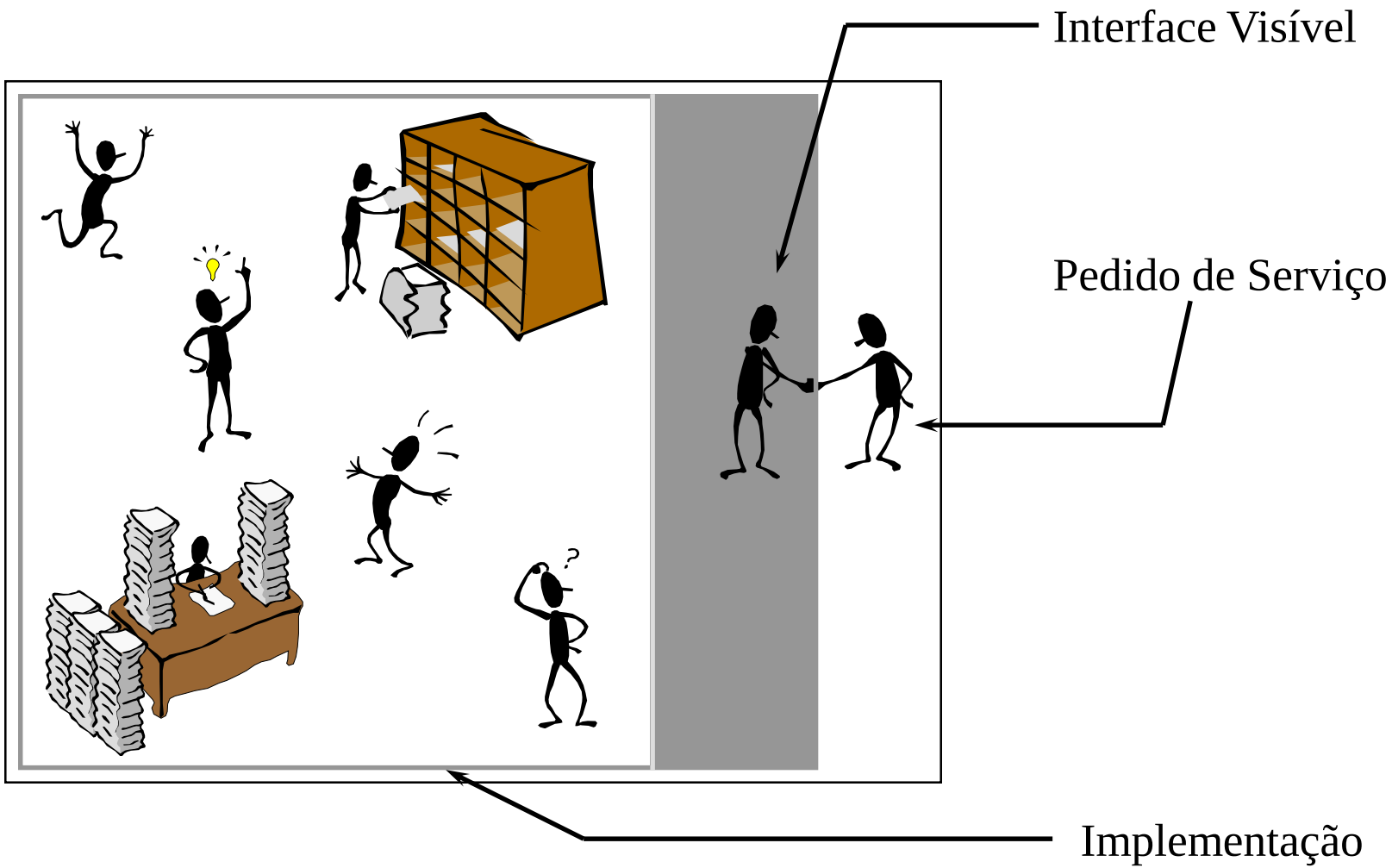
- Colaboração
  - Um programa OO é um conjunto de objetos que colaboram entre si para a solução de um problema
  - Objetos colaboram através de trocas de mensagens
  - A troca de mensagem representa a chamada de uma operação



Encapsulamento

- O encapsulamento é um princípio do desenvolvimento orientado a objetos
- O encapsulamento determina que a implementação de um objeto somente deve ser acessada através de uma assinatura de método e bem definida.

Encapsulamento

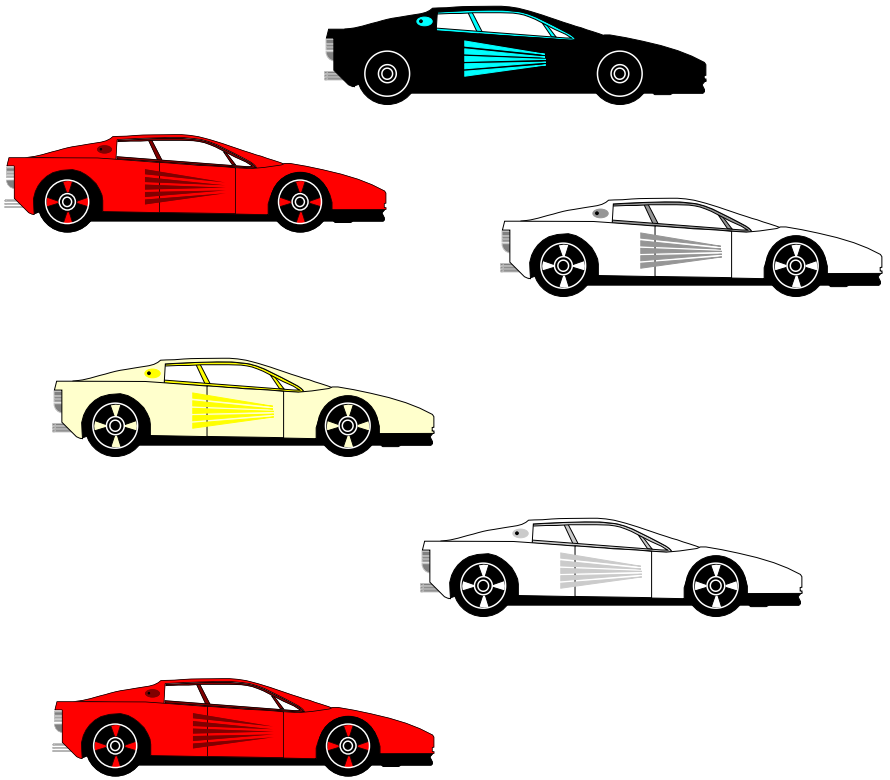


Classe & Objetos

Classe Carro

Carro
Número de Rodas Cor Cor Lateral
Anda Para Acelera Estaciona

Objetos da classe Carro



Métodos Especiais

- Criação de Objetos
  - A classe é responsável pela criação de seus objetos
  - Esta criação é realizada através de um método especial, chamado de **construtor**

Instanciação de Objetos

- Objetos devem ser instanciados antes de utilizados
- O comando ***new*** instancia um objeto
- O comando ***new*** chama o construtor da classe (operação com o mesmo nome da classe).
- Os parâmetros do construtor devem ser passados entre parênteses

```
String x;

x = new String(“Curso Java");
```

Objetos não Inicializados

- Valor ***null***
  - Utilizado para representar um objeto não inicializado
  - Quando um método retorna um objeto, ele pode retornar ***null*** para indicar, por exemplo, que o objeto não foi encontrado
  - Podemos atribuir ***null*** para descartar um objeto previamente instanciado

```
Carro    MeuFusca;
...
MeuFusca = null;
```

Invocando métodos de um objeto

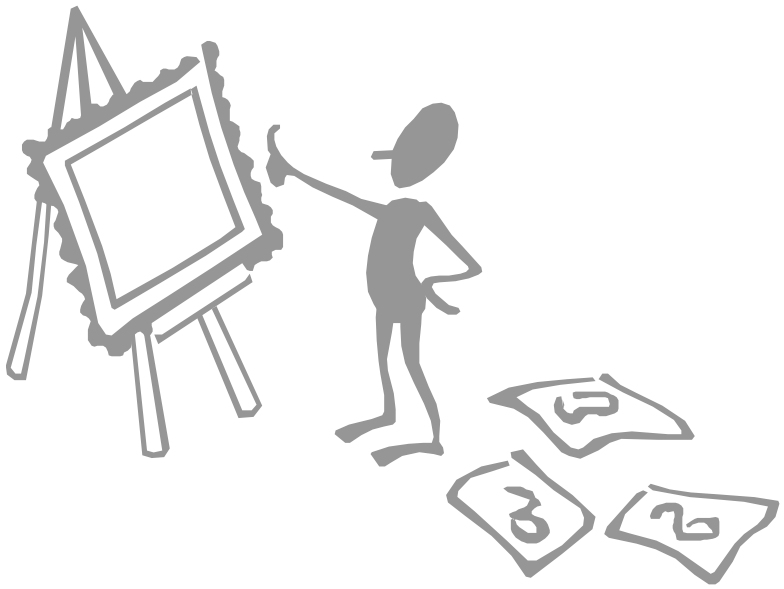
- **objeto.metodo (argumentos);**

  String x = new String(“Nova string”);

  String y;

  y = **x.substring**(0, 4);

Definindo Classes em Java



- Tópicos
  - Declaração de classes
  - Visibilidade
  - Atributos
  - Métodos

Classes

- Regras para a criação de um arquivo .java
  - Só pode haver uma única classe pública por arquivo
  - Caso haja uma classe pública no arquivo (pode não haver!), o nome do arquivo deve bater com o nome da classe. Se no arquivo existe uma classe declarada como `public class Animal { }` o nome do arquivo deve ser `Animal.java`
  - Um arquivo sem classes públicas pode ter qualquer nome (não precisa bater com o nome de nenhuma classe)
  - Se a classe está em um pacote (package), a declaração do pacote deve ser a primeira linha de código do arquivo, antes de qualquer import

Classes

- Regras para a criação de um arquivo .java
  - Se há algum comando import, ele deve vir entre a declaração do pacote e a declaração da classe. O import deve ser a primeira linha de código do arquivo se não houver declaração de pacote. Se não houver nem declaração de pacote nem import, a declaração da classe deve ser a primeira linha de código do arquivo
  - A declaração do pacote e os imports são globais, afetando todas as classes presentes no arquivo
  - Um arquivo pode conter mais de uma classe não pública

Classes

- Declaração
  - Declaração básica:

```
class MyClass { }
```

  - Incluindo os modificadores de visibilidade:

```
public class MyClass { } // OK
protected class MyClass { } // erro de compilação!
private class MyClass { } // erro de compilação!
```

  - Incluindo outros modificadores permitidos:

```
strictfp class MyClass { }
final class MyClass { }
abstract class MyClass { }
```

Classes

- Modificadores de visibilidade
  - Controlam o acesso às classes
  - Há quatro níveis de controle:
    - Default
    - Public
    - Protected
    - Private
  - ATENÇÃO: Há apenas três modificadores!
  - Apenas public e default se aplicam a classes

Classes

- Acesso a classes
  - Se o código em uma classe A acessa uma classe B (A “enxerga” B), a classe A pode fazer as seguintes operações com B:
    - Criar uma instância de B
    - Estender B (tornar-se subclasse de B)
    - Acessar métodos e variáveis da classe B, desde que permitido pelos respectivos modificadores

Classes

- Acesso default
  - Ocorre quando não é especificado nenhum modificador
  - Uma classe com acesso default somente pode ser “vista” por classes do seu mesmo pacote
  - O exemplo a seguir não compila:

Arquivo Bebida.java

```
package cert;
class Bebida { }
```

Arquivo Cafe.java

```
package test.stuff;
import cert.Bebida; // erro de compilação!
class Cafe extends Bebida { }
```

Classes

- Acesso default
  - Exercício: Corrigir o exemplo anterior
  - Dica: há duas soluções possíveis

Classes

- Acesso public
  - Uma classe declarada como public permite que qualquer outra classe dentro de qualquer pacote a acesse
  - Se as classes estiverem em pacotes diferentes, é necessário o comando import
  - Exemplo:

Arquivo Bebida.java

```
package cert;
public class Bebida { }
```

Arquivo Cafe.java

```
package test.stuff;
import cert.Bebida;
class Cafe extends Bebida { }
```

Classes

- O modificador `final`
  - Uma classe declarada como final não pode ser estendida
  - Se a classe A é final, a linha a seguir gera erro de compilação:

```
class B extends A { }
```

  - O uso de final garante que todos os métodos da classe A nunca poderão ser redefinidos (impede o override)
  - Desvantagem: sem herança não há possibilidade de especialização
  - Exemplo de classe final presente na API Java : String

Classes

- O modificador `final`
  - Exemplo:

Arquivo Bebida.java

```
package cert;
public final class Bebida { }
```

Arquivo Cafe.java

```
package test.stuff;
import cert.Bebida;
class Cafe extends Bebida { } // erro de compilação!
```

Classes

- O modificador `abstract`
  - Uma classe marcada com o modificador `abstract` (classe abstrata) nunca pode ser instanciada
  - A classe abstrata existe apenas para ser estendida
  - `final` e `abstract` são inimigos!
  - Aplica-se quando a classe é muito genérica

Classes

- O modificador `abstract`
  - Exemplo:

```
abstract class Automovel {
   private double preco;
   private String marca;
   private String modelo;
   private String ano;

   public abstract void acelera(); // métodos abstratos terminam em ';'
   public abstract void acendeFarois();
   public abstract void buzina();
}
```

  - Se a classe Locadora tentar instanciar um objeto da classe Automovel…

```
Automovel a = new Automovel(); // erro de compilação!
```

Classes

- O modificador `abstract`
  - A presença de um único método abstrato requer que a classe seja abstrata
  - Uma classe pode ser abstrata sem que tenha qualquer método abstrato
  - Uma classe abstrata pode ter métodos não abstratos
  - Exemplo:

```
abstract class Automovel {
   private double preco;
   private String marca;
   private String modelo;
   private String ano;

   public String getModelo() { return modelo; }
}
```

Classes

- O modificador `abstract`
  - Exercício: Criar uma classe abstrata e uma classe concreta que a estende

Declaração - Resumo

- Declaração
  - Uma classe é declarada através da palavra chave ***class***
  - Esta palavra reservada deve ser seguida do nome da classe
  - O nome da classe é seguido do corpo da classe

- Corpo de uma classe
  - O corpo de uma classe contém os atributos e métodos da classe
  - O corpo de uma classe é delimitado por um par de chaves

```
public class Carro {
    // declaração de atributos
    // declaração de métodos
}
```

Visibilidade da classe

- Visibilidade da classe: public ou package (default)
  - Classes com visibilidade package somente podem ser manipuladas por outras classes definidas no mesmo módulo de código (package).
  - Classes públicas podem ser manipuladas por qualquer outra classe da mesma aplicação.

- Classes Públicas
  - O modificador ***public*** deve ser apresentado antes da palavra reservada ***class***, nas declarações de classes públicas.
  - Em Java, toda classe pública deve ser declarada em um arquivo com extensão "**.java"** com o mesmo nome da classe
  - 1 arquivo "**.java**" somente pode ter 1 classe com visibilidade ***public***

Analista de Sistemas - 2009 – Tribunal de Justiça/Paraná

1 - Com base no trecho de um programa em Java, abaixo, podemos afirmar que:

```
public class HelloWorld
{
        public static void main(String Args[])
            {
                    System.out.println(“Hello World”);
            }
}
```

a) O nome do arquivo poderá ser OlaMundo.

b) O nome do arquivo deverá ser diferente do nome da classe.

c) Este é um exemplo de programa que não possui classes.

d) O nome do arquivo deverá ser o mesmo nome da classe.

Analista de Sistemas - 2009 – Tribunal de Justiça/Paraná

2 - Assinale a sigla correspondente à plataforma Java para dispositivos compactos, como celulares, PDAs, controles remotos e uma outra gama de dispositivos.

a) Eclipse

b) JVM

c) J2ME

d) JSP

Analista de Sistemas Eletronorte – 2005 – NCE/UFRJ

O nome que se dá a um aplicativo Java que é executado em um browser cliente carregado através de uma página HTML é:

A) Applet;

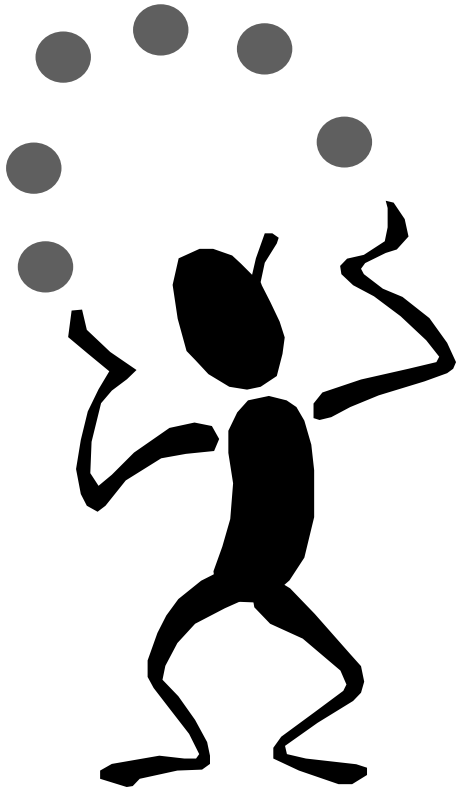
B) JavaBeans;

C) JavaDoc;

D) JSP;

E) Servlet.

Atributos em Java



- Atributos
  - Declaração
  - Modificadores
  - Tipos
  - Nomes
  - Inicialização

Declaração

- A declaração de um atributo é composta de
  - Modificadores de atributos
  - Tipo do atributo
  - Nome do atributo
  - Valor inicial do atributo (opcional)

- Exemplo

```
public class Carro {
        private double velocidade = 0.0;
}
```

Modificadores de Visibilidade

- Indicam os objetos que poderão manipular o atributo
  - O modificador ***public*** indica que qualquer objeto pode manipular o atributo
  - O modificador ***private*** indica que somente objetos da mesma classe podem manipular o atributo
  - O modificador ***protected*** indica que somente objetos da mesma classe ou classes descendentes (em qualquer pacote) ou classes do mesmo pacote podem manipular o atributo
  - default: visibilidade **package**

Modificador de Escopo

- Indica se o atributo pertence ao objeto ou à classe
  - Um atributo de objeto possui um valor distinto para cada objeto da classe
  - Um atributo de classe possui um valor único para todos os objetos da classe
  - Sempre que um objeto altera o valor de um atributo de classe, a alteração se reflete em todos os objetos da classe
  - O modificador ***static*** indica que o atributo pertence à classe
  - Por default, o atributo pertence ao objeto

Manipulação de atributos estáticos

```
public class carroPasseio {
    static int valorImpostoIPVA = 4;
    public static int getValorImpostoIPVA() {
        return valorImpostoIPVA;
    }
}
```

Exemplo de execução do código:

```
carroPasseio c1 = new carroPasseio();
carroPasseio c2 = new carroPasseio();
carroPasseio.valorImpostoIPVA = 2;
System.out.println(c1); //imprime 2
System.out.println(c2); //imprime 2
```

Modificador de Redefinição

- Indica se o valor do atributo pode ser alterado
  - O modificador **final** indica que o valor do atributo não poderá ser alterado após a primeira atribuição
  - Muito utilizado na definição de constantes
  - Por default, o valor do atributo sempre poderá ser alterado

```
public class carroPasseio {
    final static int valorImpostoIPVA = 4;
    public static int getValorImpostoIPVA() {
        return valorImpostoIPVA;
    }
}
```

Tipos Primitivos

- Inteiros

  byte: 8-bits

  short: 16-bits

  int: 32-bits

  long: 64-bit

- Números Reais
  - **float**: precisão simples 32-bits (IEEE 754 SPFP)
  - **double**: precisão dupla 64-bits (IEEE 754 DPFP)
- Outros
  - **char**: caractere 16-bit (Unicode)
  - **boolean**: pode receber dois valores (true ou false)

Arrays

- Java permite a definição de arrays de atributos
  - Um array deve ser declarado e inicializado antes de ser utilizado
  - A declaração indica o tipo dos elementos e o nome do array
  - A inicialização indica o número de elementos do array
  - Um array pode ser inicializado logo após a sua declaração

```
int[ ]          numeros;
double [ ]      itens = new double[40];
int[]           valores = { 2, 3, 4, 5, 10 };

numeros = new int[10];
```

Acesso em Arrays

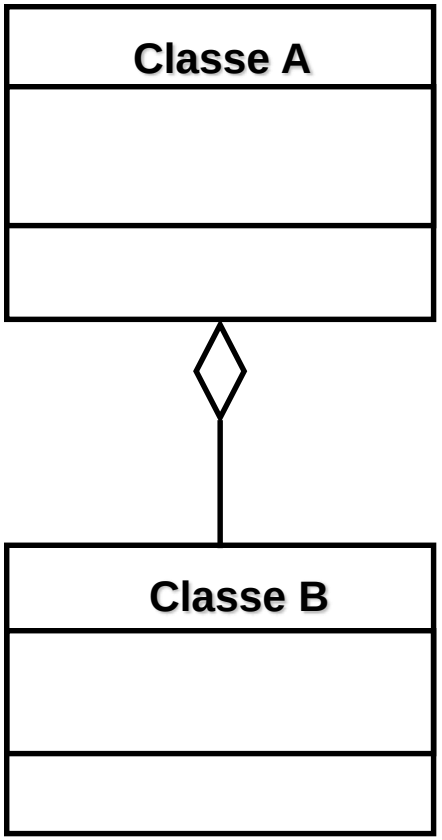
- Itens de arrays são acessados pelo operador **[ ]**
  - O número do item desejado é apresentado entre colchetes
  - O primeiro item do array possui número zero
  - O número do último item é igual ao tamanho menos 1
  - O atributo *length* contém o número de elementos do array

```
int [ ]     numeros = new int [10];

…
for ( j = 0; j < numeros.length; j++)
            numeros [ j ] = j;

numeros [10] = 10;          // Erro: área não reservada
```

Outras Classes

- Associações entre Classes
  - Outras classes podem ser utilizadas como tipos dos atributos de uma determinada classe
  - Neste caso, o atributo representa uma associação entre objetos das duas classes.
  - O desenvolvedor deve definir a visibilidade da associação, ou seja, quais classes conhecem a associação.

Objetos Atributos




```
class A
{
        private B       b;
        …
}

class B
{
        private A       a;
        …
}
```

ANALISTA DE
DE SISTEMAS JR / ENG. DE SOFTWARE – Petrobras
2010 - Cesgranrio

6 - Considere o seguinte trecho de código em Java:

// **Arquivo C1.java**

```
package br.com.pk1;
public class C1 {
int x;
public int y;
protected int z;
private int w;
}
```

// **Arquivo C2.java**

```
package br.com.pk2;
public class C2 extends C1 {
}
```

A Classe C2 pode manipular os atributos

(A) x, y, z (B) y, z

(C) x, y (D) y

(E) x, y, z, w

ANALISTA DE SISTEMAS JÚNIOR – COPEL – 2010 – PUC/PR

7 - Encapsulamento é um conceito da orientação a objetos que tem por objetivo proteger atributos e métodos de um objeto. Baseado nesta premissa, qual trecho de código *Java deixará o atributo "saldo" com* acesso mais restritivo?

A) *Protected double saldo.*
B) *Restricted double saldo.*
C) *Double saldo.*
D) *Private double saldo.*
E) *Readonly double saldo.*

PRODAUB - Analista de sistemas Senior - TRADE CESUS

8 - Em Java, a palavra reservada (modificadores) utilizada para um método que não pode ser sobreposto (overriden) é conhecida como:

A)Static;

B)Private;

C) Final;

D) Native;

E) Protected.

Analista de Tecnologia da Informação 2009 UFF COSEAC

10 - Em Java, das opções abaixo, aquela que indica o modificador que só pode ser executado em um atributo ou método de uma classe, ainda que para qualquer um dos casos sua presença indique que o alvo estará acessível para qualquer subclasse ou classe pertencente ao mesmo pacote da classe identificada, é:

(A) Private;

(B) Public;

(C) Protected;

(D) Static;

(E) Abstract.

Exercício 1

1 - Determine qual é a idade que o usuário faz no ano atual. Para isso solicite o ano de nascimento do usuário e o ano atual

- Dicas:
  - Importar a classe Scanner
  - Ler os atributos com os seguintes comandos:

```
System.out.println(String)
anoAtual = new Scanner(System.in).nextInt();
```

Exercícios 2

2 - Escreva um programa que receba do usuário o tamanho de cada lado do triângulo e imprima seu perímetro.

3 – Implemente um programa para calcular a área de um trapézio, onde:

h = altura

b = base menor

B = base maior

Área = (h . (b + B)) / 2

4 - Crie um programa que receba como parâmetro um valor em reais e converta para dólares (Estados Unidos) e yenes (Japão). Deve ser lida a cotação de cada moeda.