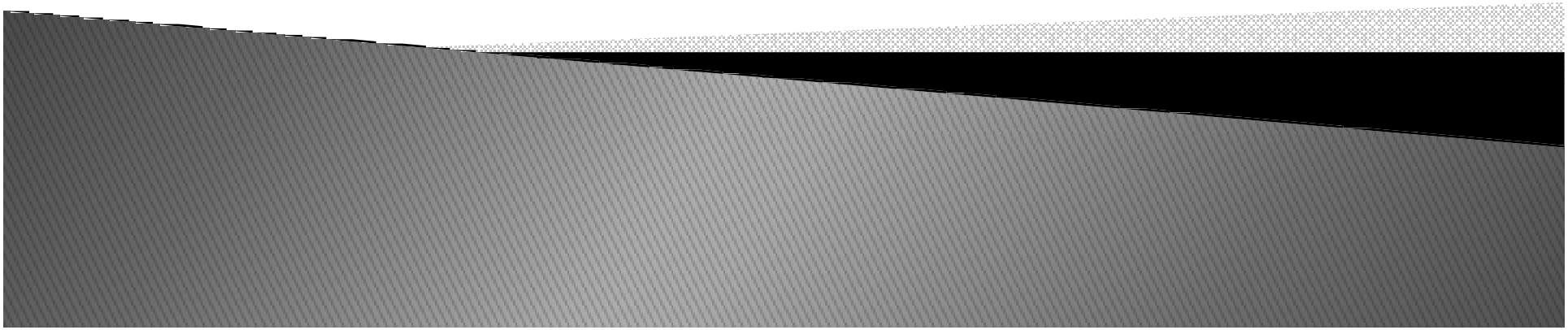
Java

Aula07

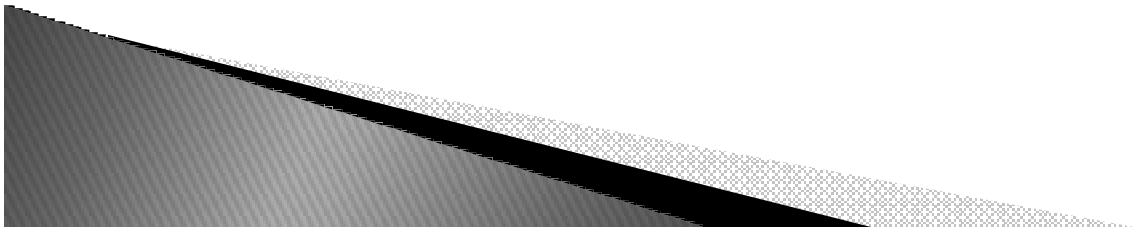
BSI – UFRPE
Prof. Gustavo Callou
gcallou@gmail.com



Primeiro Programa

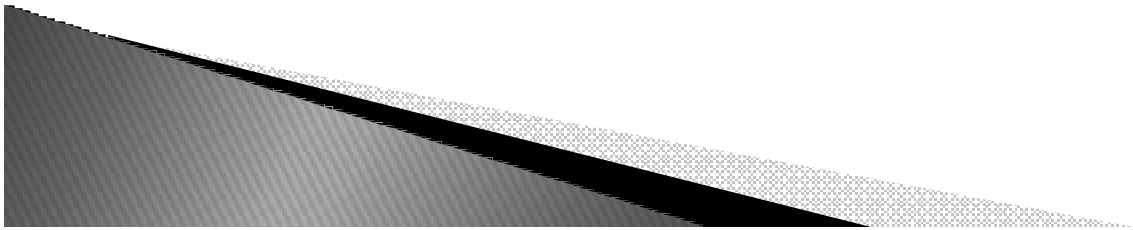
- HelloWorld.java:

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



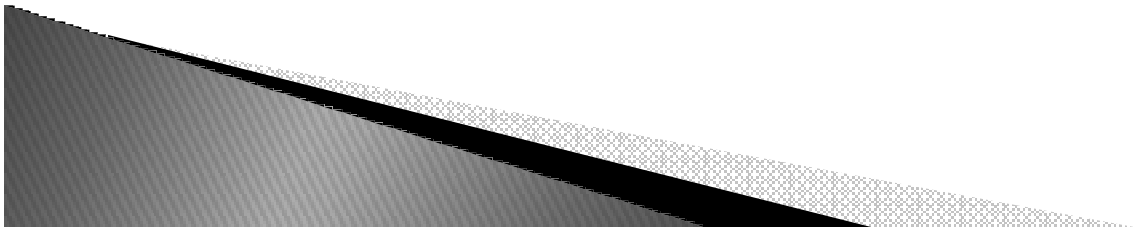
Palavras Reservadas

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	



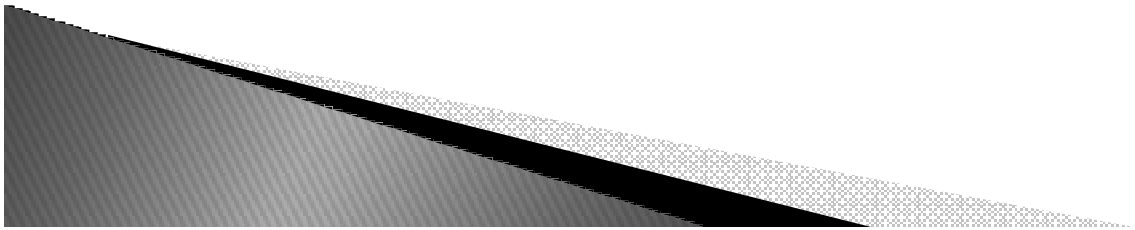
Palavras-Chave e Identificadores

- Identificadores são usados para nomear variáveis, métodos, classes ou rótulos
  - Java diferencia maiúsculas de minúsculas
    - funcionario
    - Funcionario
    - FUNCIONARIO
  - Devem iniciar com letra, sinal de dolar ($) ou underscore seguidos de letras, sinais de dolar ($), underscores ou dígitos
  - Não têm tamanho máximo



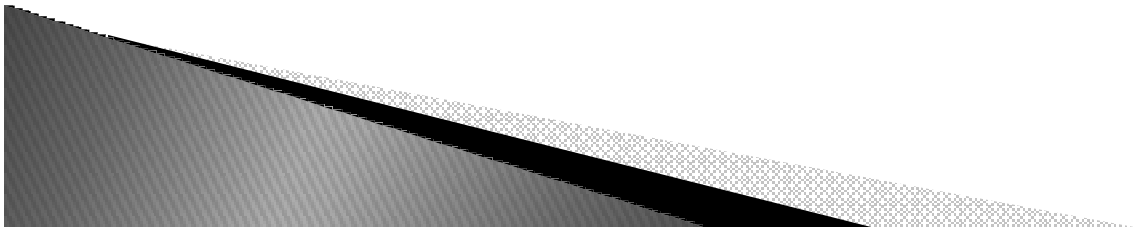
Palavras-Chave e Identificadores

- Palavra-chave não pode ser usada como identificador, mas pode fazer parte de um
- Ex.:
  - Identificadores válidos:
    - a_$, $a1, $b2, _a123, _$1, _class, bytes
  - Ilegais:
    - 1_a1, class, !erro



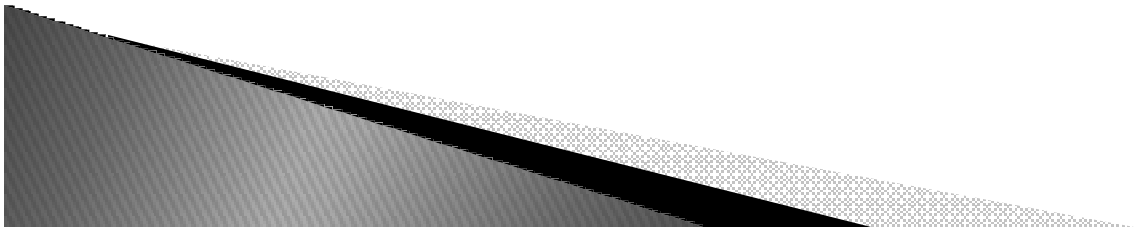
Tipos em Java

- Podem ser divididos em:
  - Tipos primitivos: elementos que não são instâncias de classes, ou seja, não são objetos
  - Tipos referência: elementos são instâncias de classes



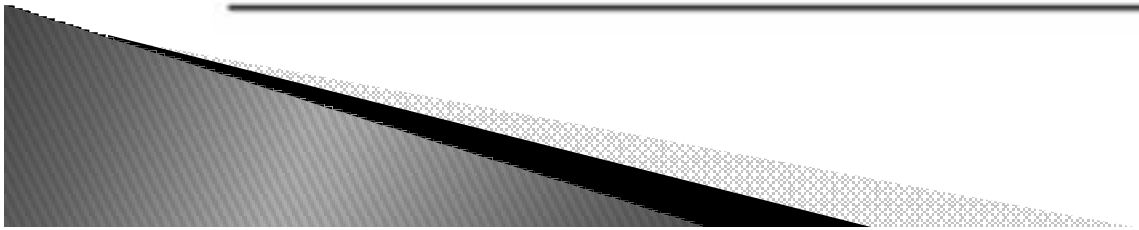
Tipos Primitivos

- Lógico
  - boolean
- Textual
  - char
- Inteiros
  - byte, short, int e long
- Pontos Flutuantes
  - float e double



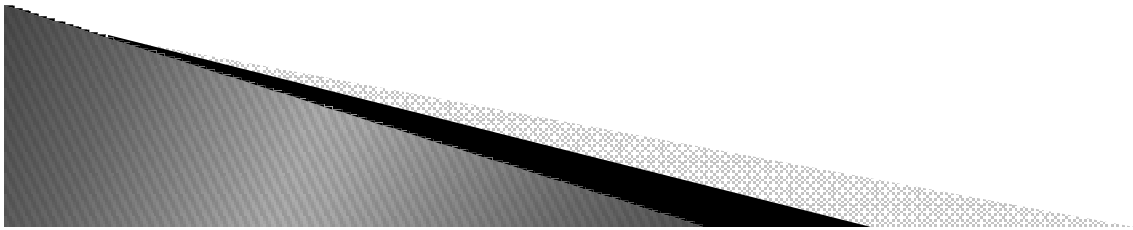
Tamanho

Type	Effective Representation Size (bits)
byte	8
int	32
float	32
char	16
short	16
long	64
double	64



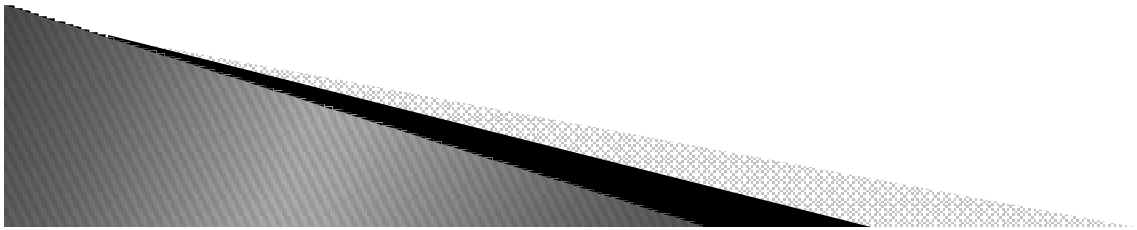
Intervalo

Type	Size	Minimum	Maximum
`byte`	8 bits	-2^7	$2^7 - 1$
`short`	16 bits	-2^{15}	$2^{15} - 1$
`int`	32 bits	-2^{31}	$2^{31} - 1$
`long`	64 bits	-2^{63}	$2^{63} - 1$

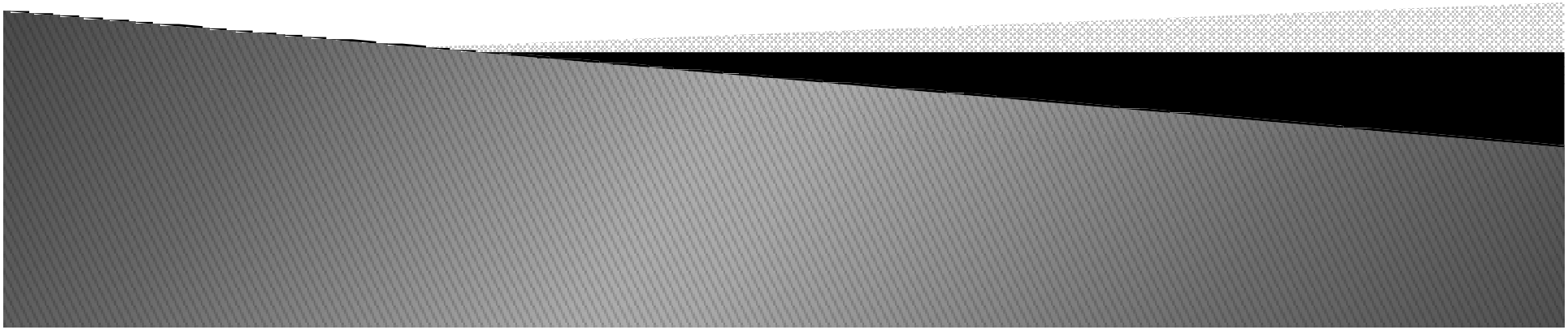


Tipos Referência

- Todos os demais tipos de Java!
- Definidos a partir de declarações de classes
- Quando uma variável em Java é declarada como sendo do tipo de uma classe, esta variável é considerada como sendo do tipo referência
- Declaração e inicialização:
  - String str = “Entendendo Strings”;
  - Banco brasil = new Banco (“Brasil”);



Arrays em Java



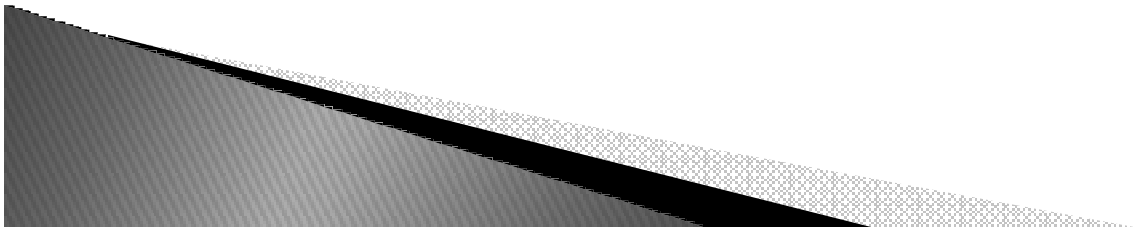
Arrays

- São objetos especiais de Java
- Uma variável do tipo array é definida usando a notação:

  tipo[] arrayNome

- Exemplo:

```
int[] c;                // declara o array
c = new int[ 3 ];       // aloca o array
```



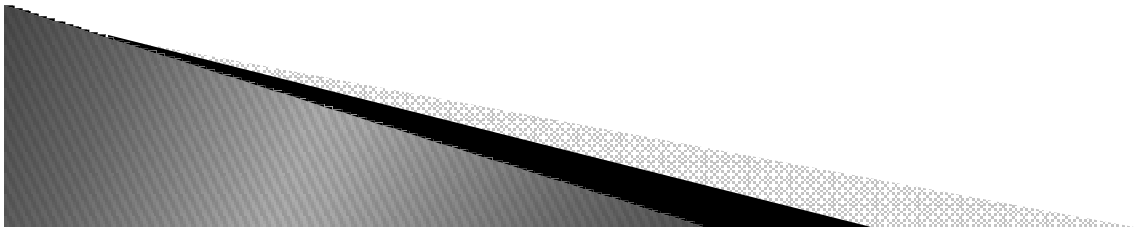
Array

- Para declararmos um array devemos seguir a sintaxe:
  - Tipo[] nome

  Para inicializar todos os valres com o valor default podemos :
- int[] ints = new int[25]

- Array bidimensional:
  - float[][]twoDee

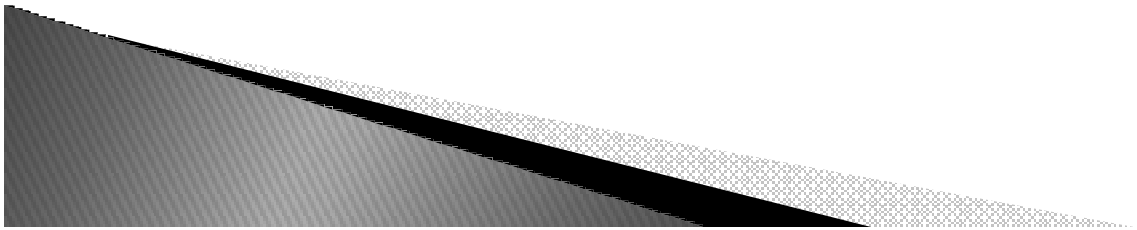
  Exemplo:
  - float[] diametro = {1.1f, 2.2f, 3.3f, 4.4f, 5.5f}



Array

- Exemplo:

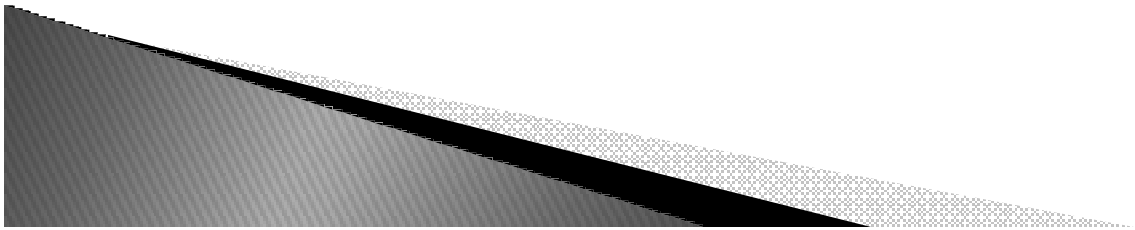
```
long[] squares;
squares = new long[6000]
for (int i=0; i<squares.lenght; i++){
  squares[i]= i*i
}
```



Criação de arrays

- O operador new X[Tamanho] cria um objeto array, não os objetos do tipo X por ele referenciado
- O primeiro elemento do array tem índice 0 e o último tem índice *tamanho*–1
- O comprimento do array é determinado pela seguinte expressão:
  length

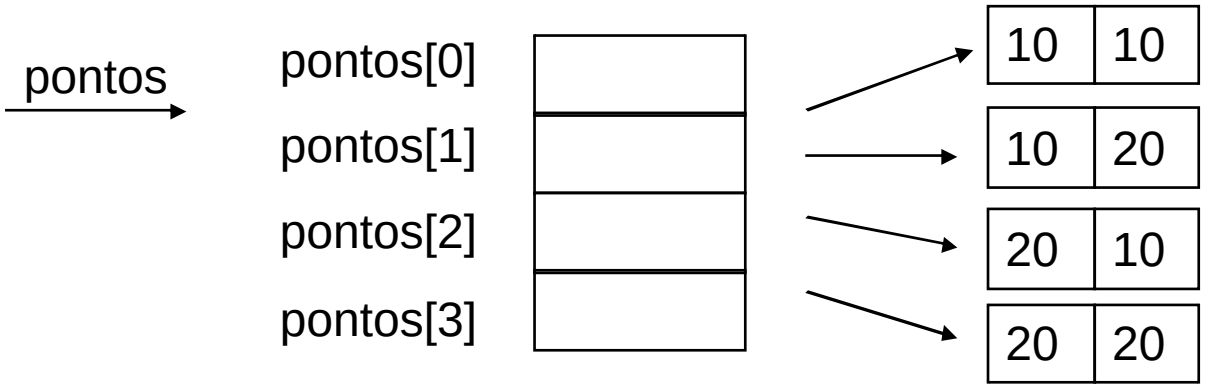
```
int[] c = {8, 5, 7};      // declara e inicializa o array
int tamanho = c.length;
```



Inicializadores

- Exemplo: Declara, cria e inicializa um array de pontos

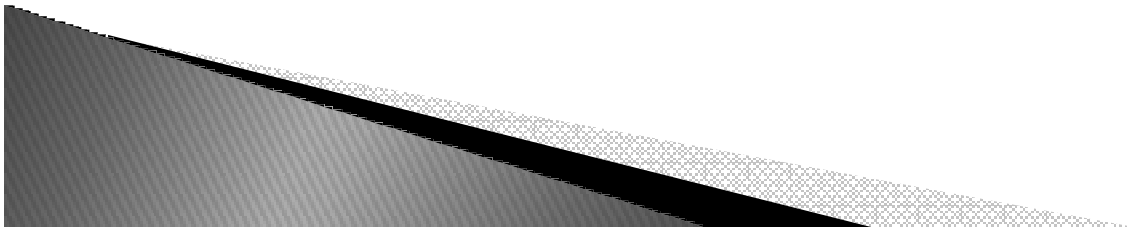
```
int[][] pontos = {{10,10}, {10,20},
                  {20,10}, {20,20}};
```



Acesso

variável[*expressao_inteira*]

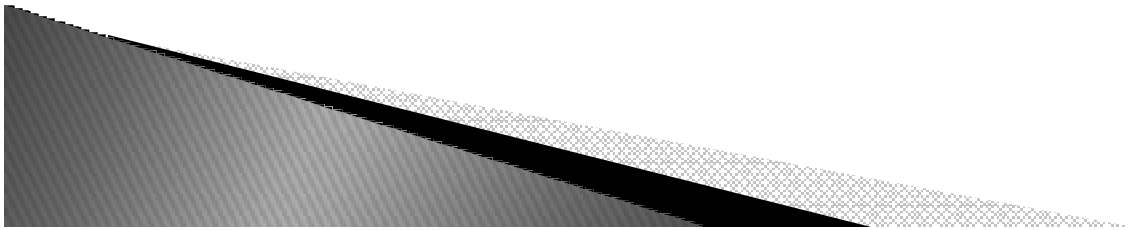
- Acesso a um array é checado em tempo de execução
- A exceção java.lang.IndexOutOfBoundsException é levantada na tentativa de acesso fora dos limites do array (0..TAMANHO-1)



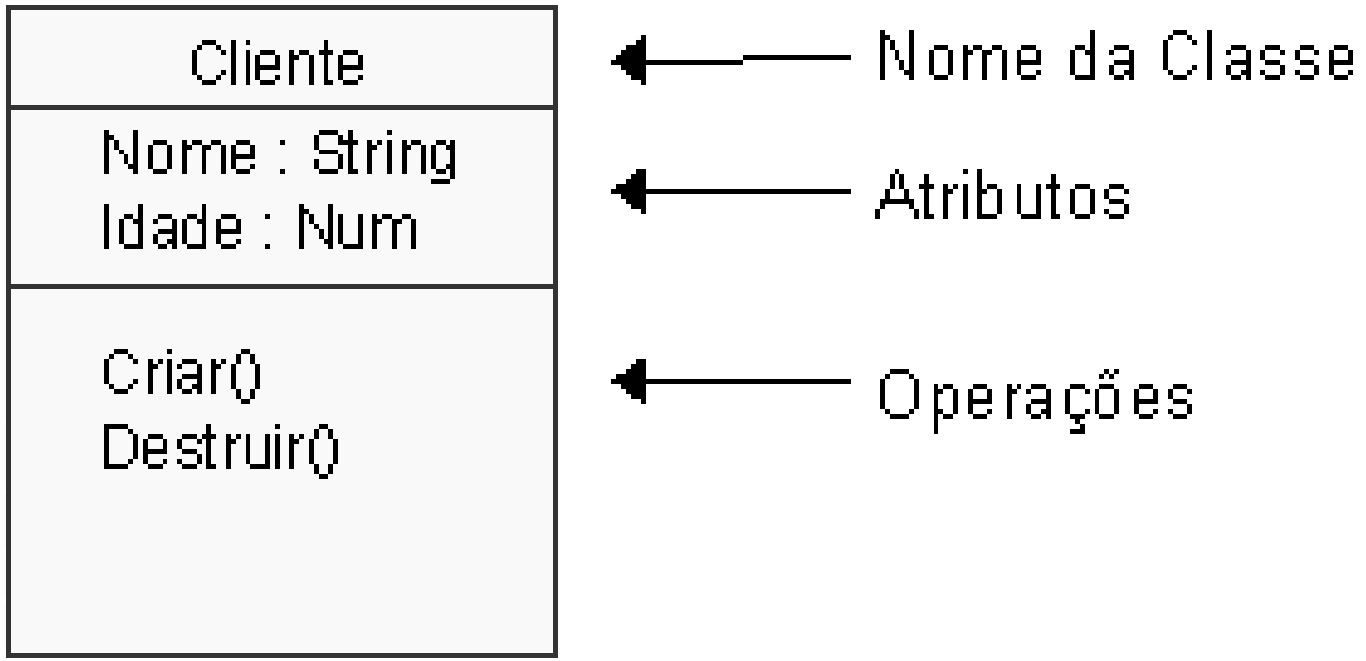
Acesso

- Exemplo:

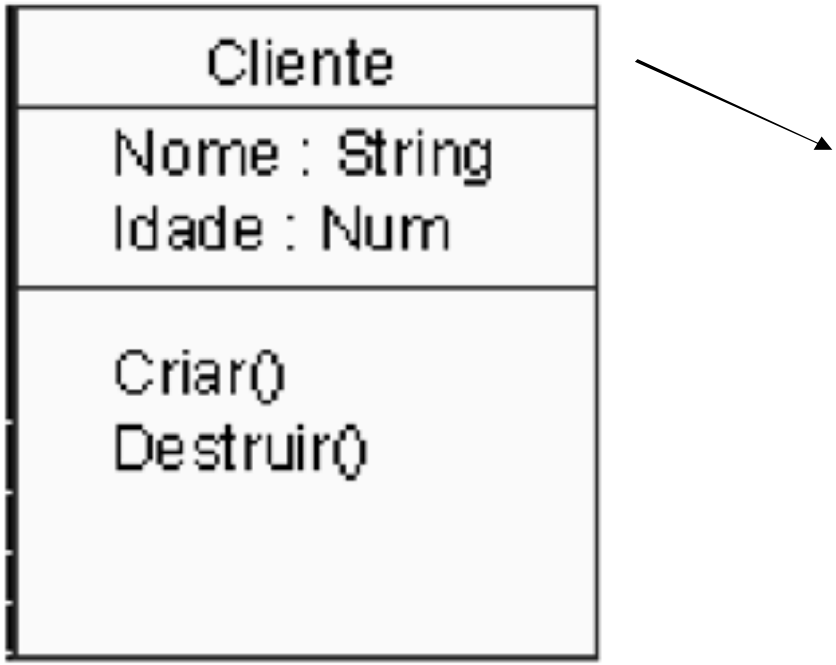
```
Conta[] contas;
if(contas[i].getNumero().equals(numero))
  achou = true;
else
  ...
```



Classes

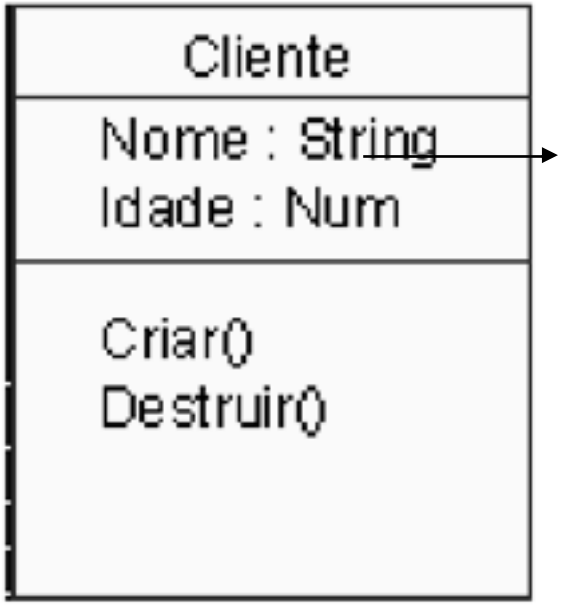

Cliente
Nome : String
Idade : Num
Criar()
Destruir()
Nome da Classe
Atributos
Operações

Declaração de Classe



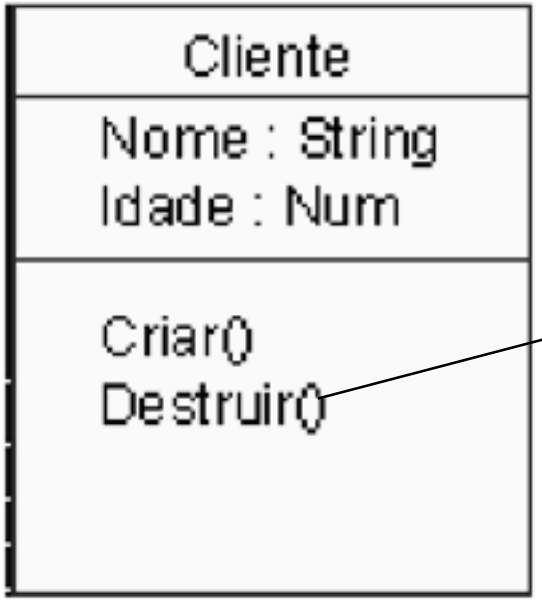
public class Cliente

Declaração de Atributo



private String nome;
private int idade;

Metodos

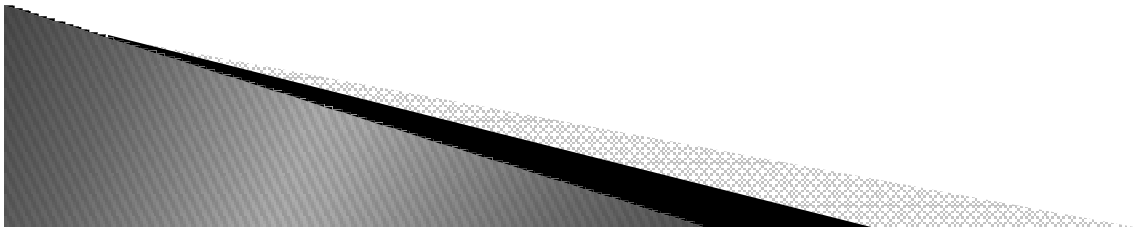



```
Public void criar(){
.
.
.

}
public void destruir(){
.

.}
```

Pacotes

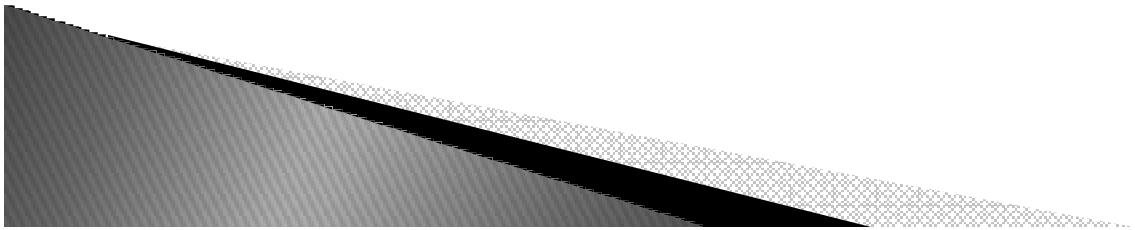
- Mecanismo para Organizar elementos em grupos
- Facilidade de entendimento do sistema
- Favorece modularidade e reuso em larga escala

pakage pessoas;



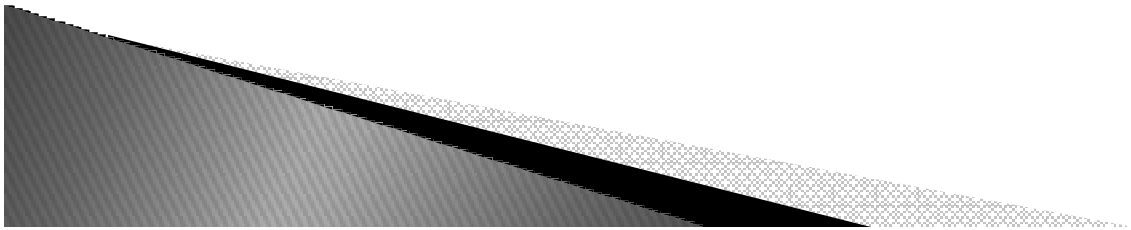
Modificadores

- Private – o atributo ou método so pode ser acessado dentro da sua classe
- Protected –o atributo ou metodo pode ser acessado dentro do seu pacote ou em suas subclasses
- Public – todos podem ter acesso



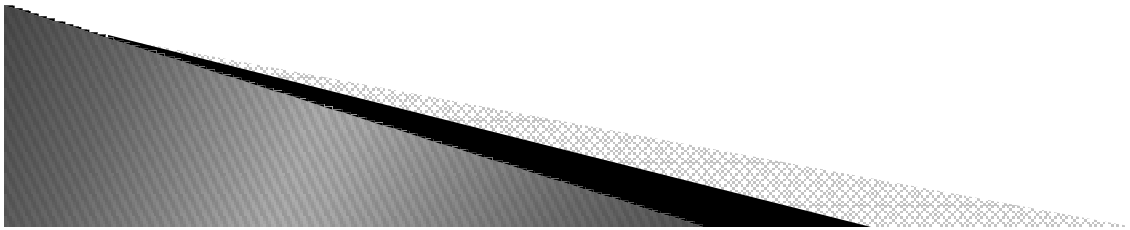
Comparação de Objetos

- O "==" compara as referencias dos objetos
- Quando se deseja comparar se os objetos são iguais usa-se o metodo *equals*



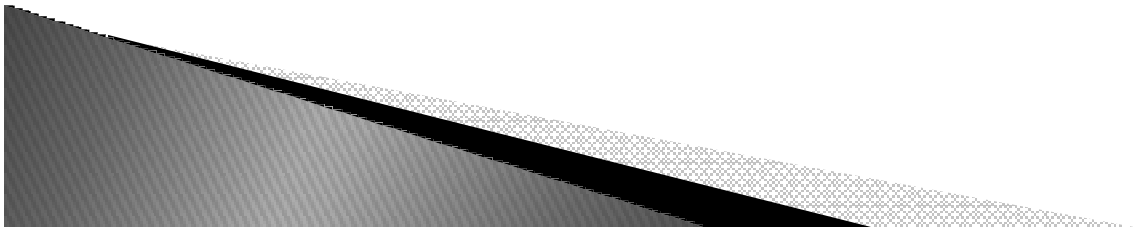
Strings

- Strings
  - São seqüências de caracteres delimitados por “”
  - Representadas por instâncias da classe String (pacotes java.lang)



Strings

- Declaração e Inicialização:
  - String saudacao = “Hello World!”;
  - Variável saudacao conterá um apontador (referência) para um objeto do tipo String que estará em memória e que “armazenará” a informação “Hello World!”



Strings

- Exemplo:
  - String s1 = “string1”;
  - String s2 = “string1”;

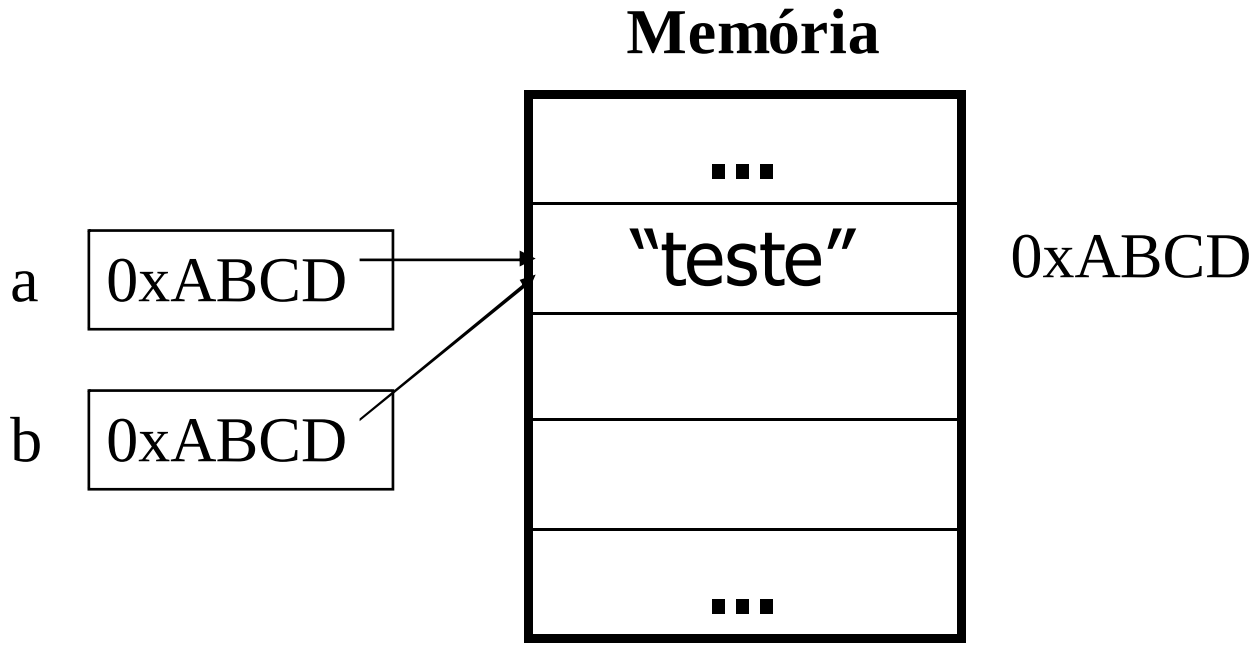
Apontam para mesma posição de memória

  - String s3 = new String(“string1”);

Vai apontar para outra posição de memória. Vai criar um novo objeto

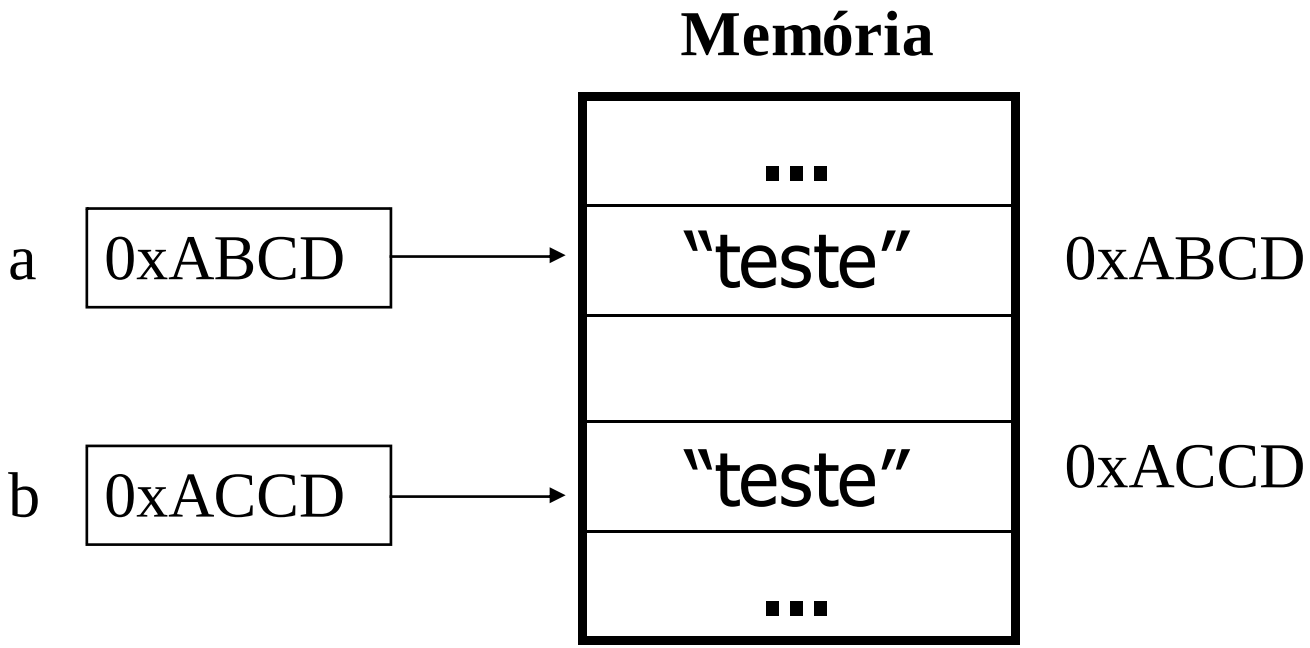
Strings: Criação

String a = “teste”;
String b = “teste”;



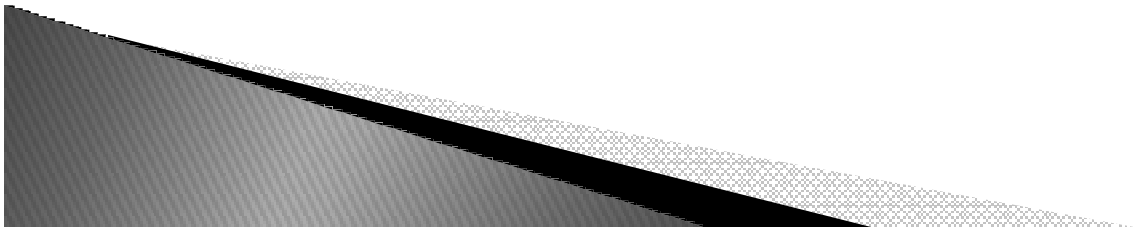
Strings: Criação

String a = “teste”;
String b = new String(“teste”);



Strings: Comparação

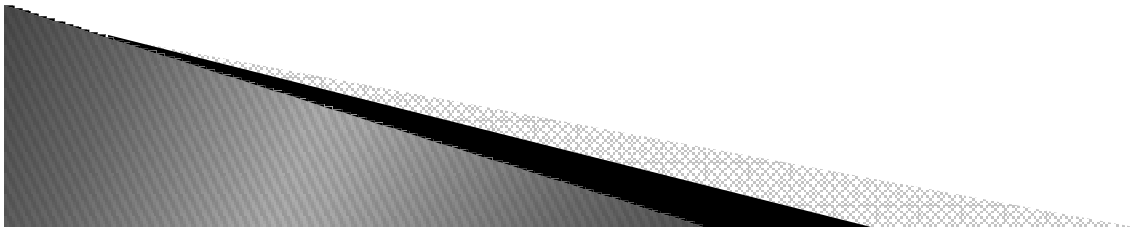
- Não use o operador == para comparar Strings
- O operador == testa referências quando utilizado com objetos, ou seja, testa se os objetos apontam para mesma posição de memória
- Use o método **equals()** da classe String
- Use o método **equalsIgnoreCase()** da classe String se não for importante a caixa (maiúsculas ou minúsculas)
  - String nome = “Lisa”;
  - if(!nome.equals(“Bart”))
  - if(“Lisa”.equals(nome))



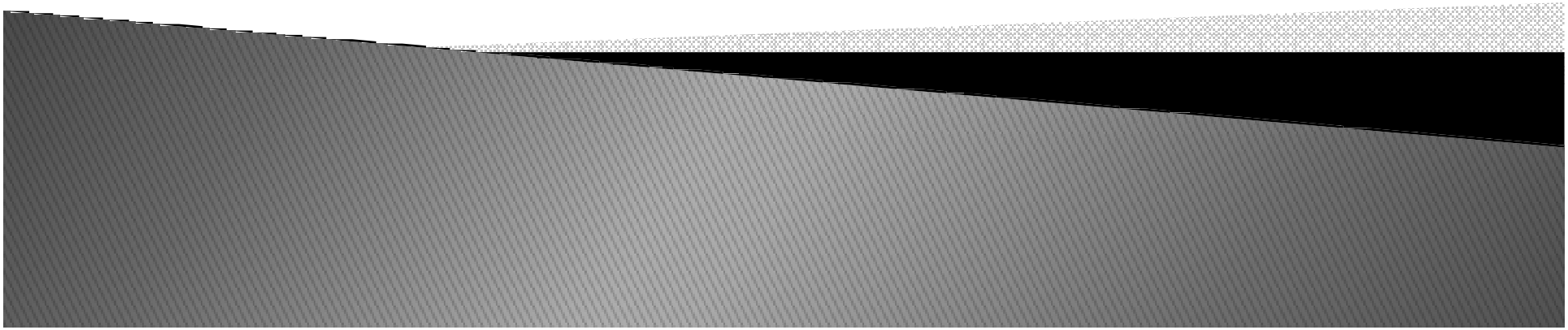
Exercícios

- Crie uma classe e implemente o método main com o seguinte código:

```
String str1 = "Sexta";
String str2 = "Sexta";
String str3 = new String("Sexta");
System.out.println(str1.equals(str2));
System.out.println(str1.equals(str3));
```



Classes



Declarando Classes

Syntax:

```
<modifiers> class <class_name> {
    [<attribute_declarations>]
    [<constructor_declarations>]
    [<method_declarations>]
}
```

Examplo:

```
public class Veiculo {
    private double maxVelocidade;
    public void setMaxVelocidade(double valor) {
            maxVelocidade = valor;
}}
```

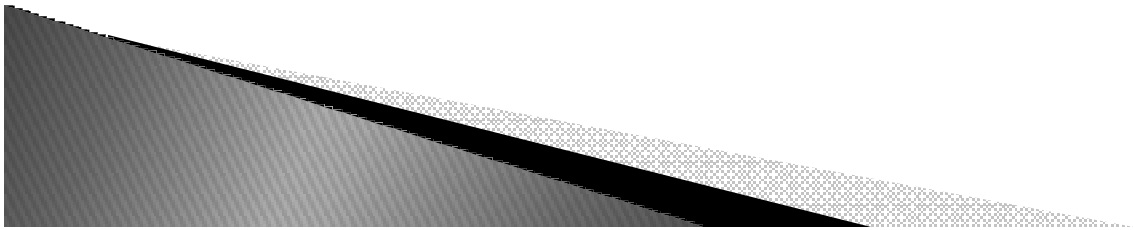
Declarando Atributos

Syntax:

[<modifiers>] <type> <name> [= <initial_value>];

Exemplo:

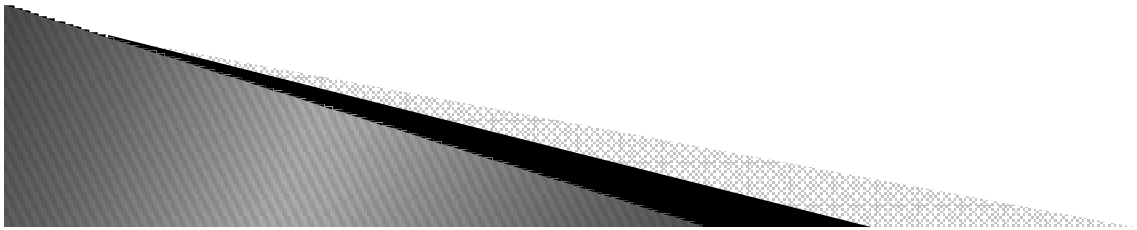
```
public class Casa {
    private int numQuartos = 3;
    private String tamanho = “media";
}
```



Declarando Metodos

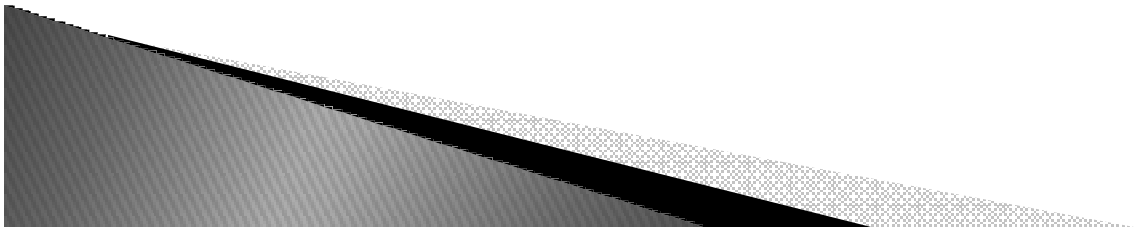
Syntax

[<modifiers>] <return_type>
  <name>([<argument_list>]) {
  [<statements>]
}



Exemplo

```
public class Cachorro {
    private int peso;
    public int getPeso() {
            return peso;
    }
    public void setPeso(int novoPeso) {
            peso = novoPeso;
    }
}
```



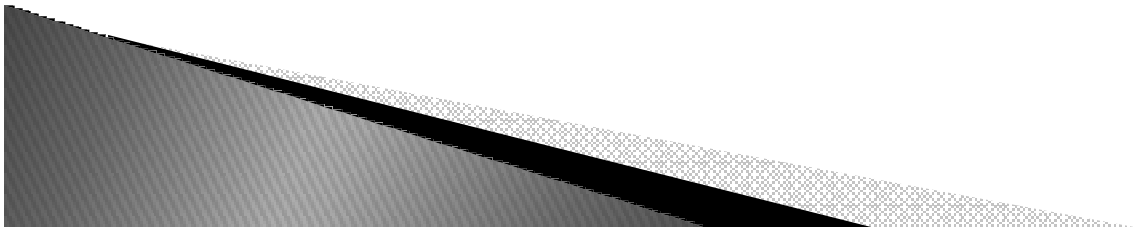
Acessando o Objeto

<object>.<member>

Exemplos:

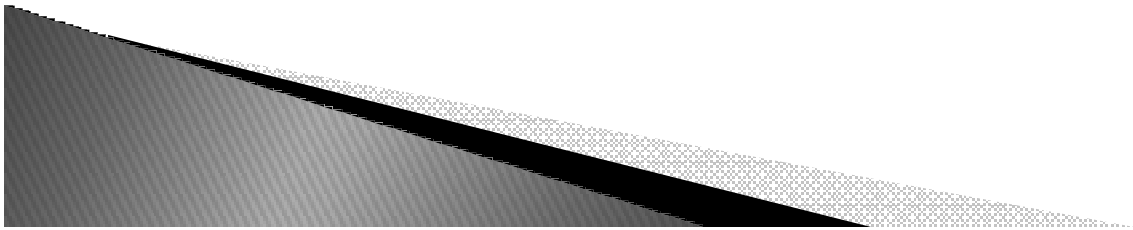
Cachorro rex = new Cachorro();

rex.setPeso(42);



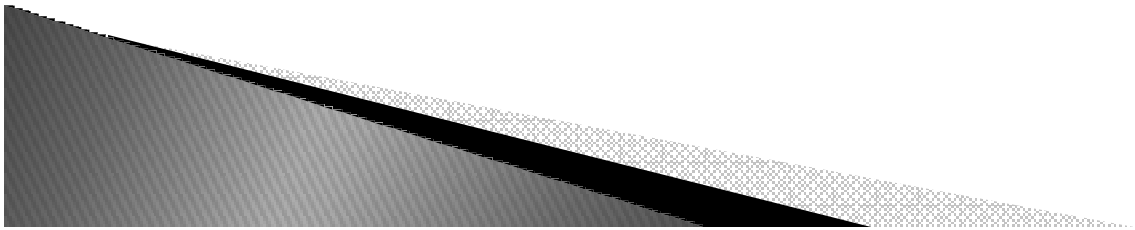
Referencia THIS

- Acessa membros da própria classe
  this.<atributo>
  this.<metodo>



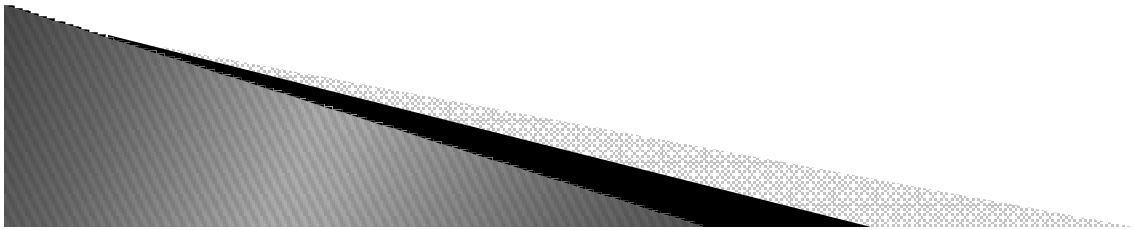
Criando Objetos

- Os objetos são criados utilizando-se a palavra reservada new.
  - Ex : Carro c1 = new Carro ();
  - Observe que :
    - c1 é uma variável, que será utilizada para referenciar uma instância da classe Carro.
    - O objeto só passa a existir depois de executa a instrução new Carro()
    - c1 é um ponteiro para a área de memória que irá armazenar as informações a respeito do objeto.



Exemplo

```
class Principal {
    public static void main (String[ ] args) {
        Carro c1 = new Carro();
        System.out.println("Id do objeto c1 = " + c1);
        c1. ano = 2005;
        Carro c2 = new Carro();
        System.out.println("Id do objeto c2 = " + c2);
        c2 = c1; // Cuidado !
        System.out.println("Id do objeto c2 = " + c2);
        c1 = new Carro(); // Cuidado !
        System.out.println("Id do objeto c1 = " + c1);
    }
}
```



Operadores

Operators	Precedence
postfix	*expr++ expr--*
unary	*++expr --expr +expr -expr* ~ !
multiplicative	* / %
additive	+ -
shift	<< >> >>>
relational	< > <= >= instanceof
equality	== !=
bitwise AND	&
bitwise exclusive OR	^
bitwise inclusive OR	|
logical AND	&&
logical OR	||
ternary	? :
assignment	= += -= *= /= %= &= ^= |= <<= >>= >>>=

Herança em Java

```
public class Poupanca extends Conta {
  //atributos especificos de poupanca
  //metodos especificos de poupanca
}
```

- Restrição de Java
  - Uma classe pode estender apenas uma superclasse diretamente

Construtores e subclasses

```
public class Poupanca extends Conta {
  public Poupanca (String num,double saldo,
                          Cliente cliente){
     super(num, saldo, cliente);
  }
}
```

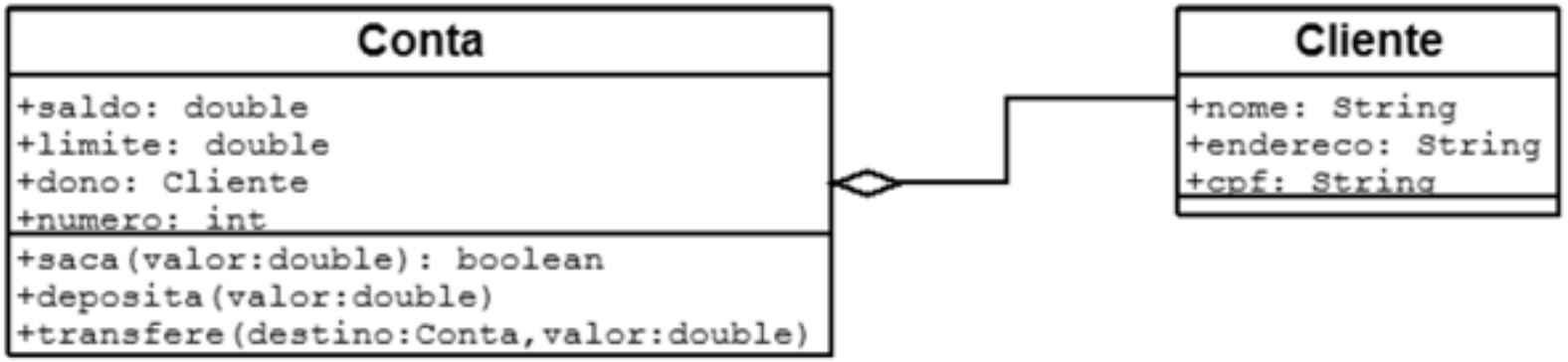
super chama o construtor da superclasse

se **super** não for chamado, o compilador acrescenta uma chamada ao construtor default: **super()**

se não existir um construtor default na superclasse, haverá um erro de compilação

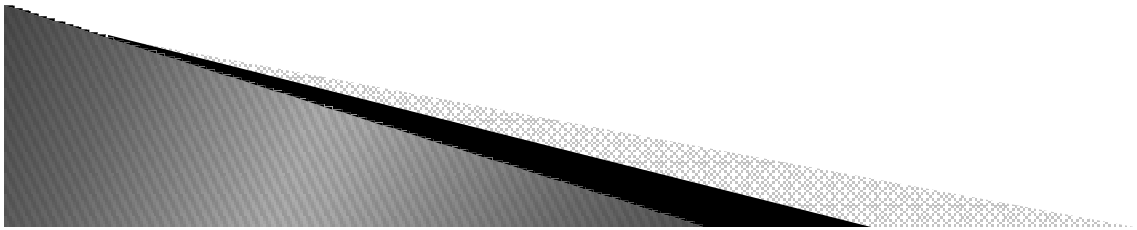
Exercício

- 1 – Faça em java o código correspondente a:



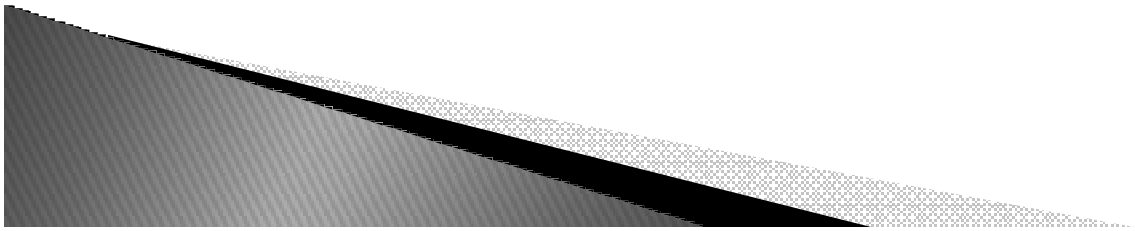
Exercicio

- 2-Defina em Java a classe Anexo para representar os anexos (attachments) que podem ser associados a um e-mail. Os objetos desse tipo devem ter o método getName, que retorna o nome de um anexo, setName, que modifica o nome de um anexo, e contains, que verifica se uma dada string faz parte do texto que constitui um anexo. Defina apenas a assinatura do último método.



Exercicio

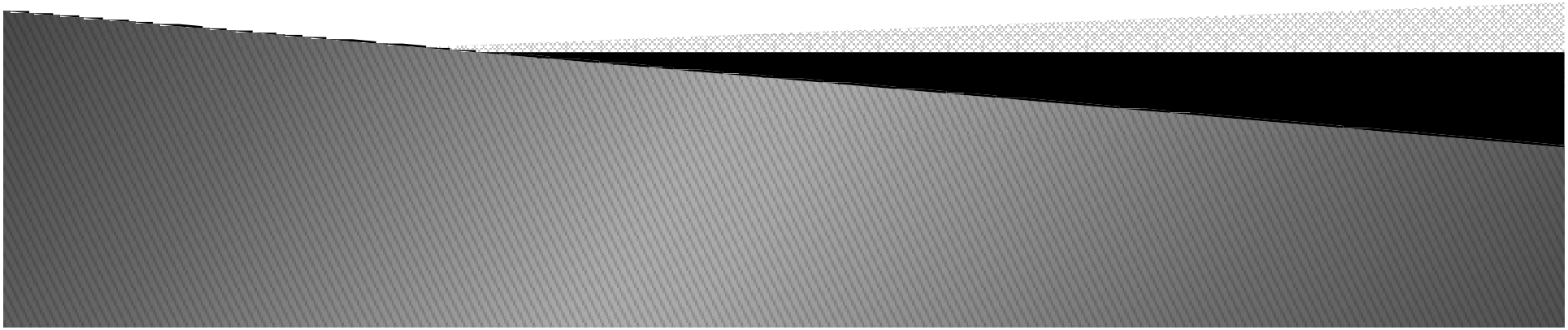
- 3-Defina em Java uma classe EMail cujos objetos têm como atributos um endereço de origem, um endereço de destino, uma mensagem, uma data e 3 anexos. Considere que todos os atributos são privados e que a data pode ser representada por uma string. Defina métodos get e set para cada um dos atributos. Na definição dos métodos set para os anexos, garanta que uma mensagem não terá dois anexos iguais.



Exercício

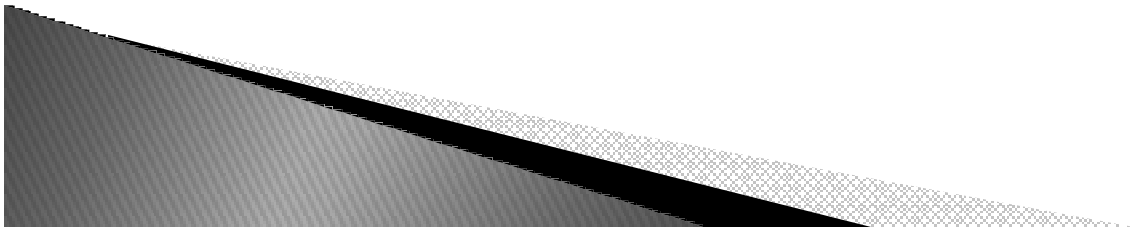
- 4 -Crie uma classe para representar uma conta-corrente, com métodos para depositar uma quantia, sacar uma quantia, sacar uma quantia e obter o saldo. Para cada saque será debitado também uma taxa de operação equivalente à 0,5% do valor sacado.

Arquivos Fonte e Pacotes



Arquivos Fonte

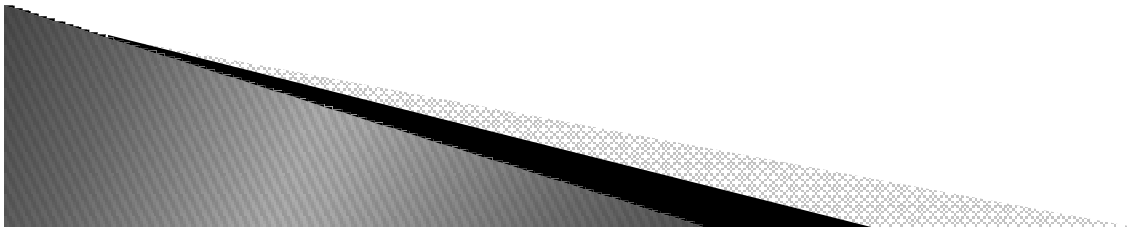
- Extensão .java
- Pode conter diversas classes
  - Uma ÚNICA pode ser pública (public)
- Mesmo nome que a classe pública mais a extensão .java



Arquivos Fonte

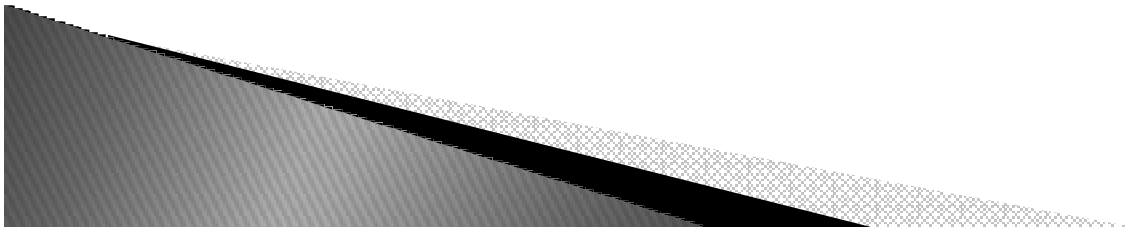
- Pode conter 3 elementos:
  1) Declaração de pacote
  2) Sentenças de importação
  3) Definição de classes

- Devem obedecer essa ordem



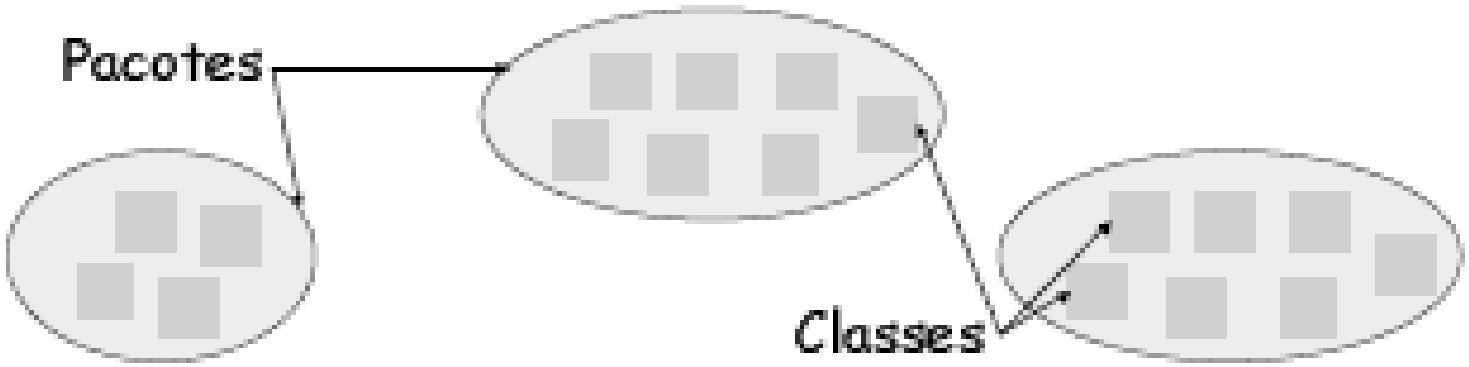
Tipos de módulos em Java

- Classes
  - Agrupam definições de métodos, atributos e construtores
  - Definem tipos
- Pacotes
  - Agrupam definições de classes relacionadas



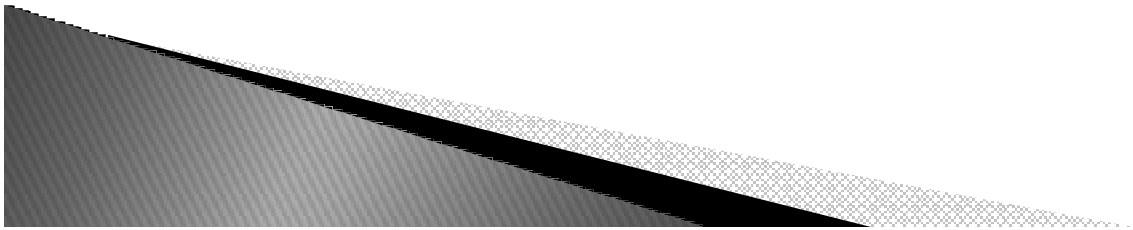
Classes e pacotes

- Pacotes facilitam a localização das classes



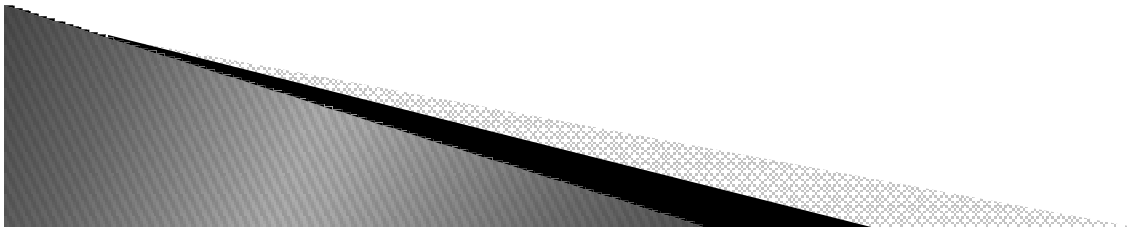
Pacotes: Definição

- Coleção de unidades de compilação relacionadas, permitindo:
  - Controle de acesso
    - Um atributo ou método protegido só fica visível dentro do pacote
  - Gerência do espaço de nomes
    - Duas classes no mesmo pacote não podem ter o mesmo nome



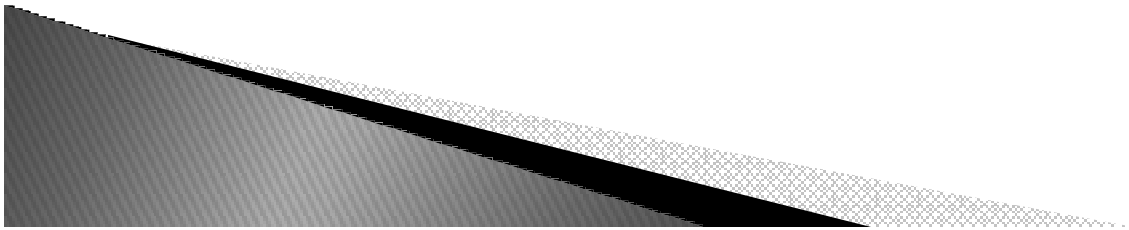
Pacotes: Declaração

- Para se declarar que um arquivo fonte pertence a um dado pacote usa-se a palavra reservada **package**:
  - package <nome do pacote>
  - Onde:
    - <nome do pacote>: série de elementos separados por "."
  - Ex.: package java.io;
- Se presente, a sentença package deve aparecer no topo do arquivo!!!
- Só pode ter um package por arquivo fonte
- Nome completo de uma classe: nome do pacote + nome da classe
  - Ex.: java.lang.String -> nome completo da classe String



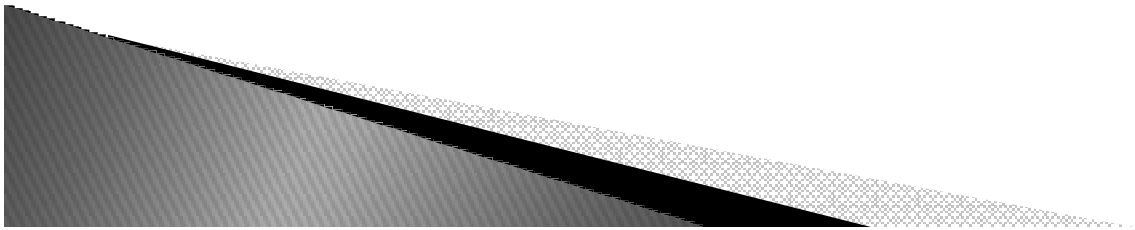
Pacotes: Importação

- Sentença import:
  - Permite a utilização de membros de pacotes sem a necessidade de referenciá-los a partir do seu nome completo
- A importação pode ser de:
  - Apenas um membro do pacote
    - Ex.: import java.awt.Button;
  - Todos os membros do pacote
    - Ex.: import java.util.*;



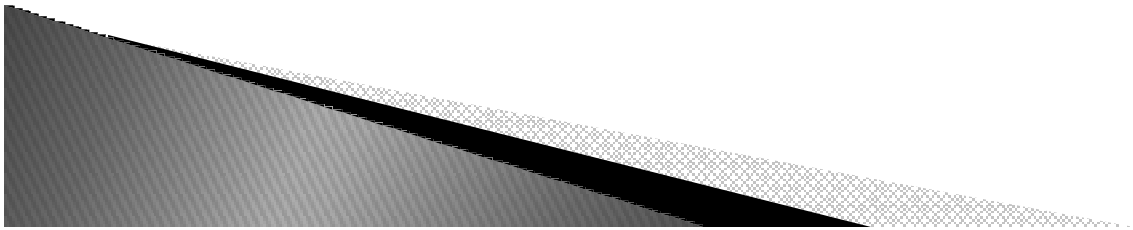
Pacotes: Importação

- Quando fazer?
  - Se a classe que precisa ser usada não estiver no mesmo pacote da classe que quer usá-la, deve-se importá-la, usando o comando:
    - import <nome completo da classe>
- import java.util.*;
  - Se não forem usadas todas as classes desse pacote, o compilador vai descobrir quais foram as classes usadas e as importá-las em tempo de compilação



Pacotes: Observações Finais

- Nome dos pacotes tem relação direta com a hierarquia do sistema de arquivos
  - Cada pacote é associado a um diretório do sistema operacional
  - Os arquivos .class das classes do pacote são colocados neste diretório
- Classe que não declara o pacote pertence ao pacote **default**
- O pacote **java.lang** é sempre importado
  - import java.lang.*;



Message Dialog

- import javax.swing.JOptionPane;
- ...
- String a= JOptionPane.showInputDialog(“nome”)
- JOptionPane.showMessageDialog(null, “welcome”)