Pacotes & Comentários

MAT A55

Pacotes

- Muitas aplicações são formadas por diversas classes
- É importante agrupar estas classes para organizá-las de acordo com uma relação específica.
- Ex. Um sistema pode usar classes de outros sistemas (reutilização)

Criação de Pacotes

- Pacotes requerem que as classes que o compõe sejam armazenadas em um mesmo diretório.
- Pode-se criar um pacote criando-se um diretório e armazenando nele todos os códigos fontes das classes que serão agrupadas.
- As classes pertencentes a um mesmo pacote deve ter no início de seu código a declaração *package* seguida do nome do diretório (ou pacote).

Criação de Pacotes

- Um ***package*** é um agrupamento de classes e interfaces que estão logicamente relacionadas.
- No Java é um conjunto de classes e interfaces que estão dentro de um mesmo diretório de arquivos.

Criação de Pacotes

- O Java utiliza os ***packages*** para:
  - garantir a unicidade do nome da classe, ou seja, não permitir que existam nomes de classes iguais em um mesmo pacote
  - e para realizar controles de acesso
    - métodos de uma classe sem a especificação do controle de acesso são considerados membros que podem ser acessados pelas classes do mesmo *package*.

Criação de Pacotes

- Para colocar uma classe em um determinado pacote, deve-se colocar a palavra reservada **package** na primeira linha de código do arquivo fonte (*.java).

```
package java.lang;
```

Exemplo de Pacotes

```
package Calendario;

public class Data{
private byte dia, mês;
private short ano;

public Data(){
}
... gets e sets ....

}
```

```
package Calendario;

public class Hora{
private byte hora, min, seg;

public Hora(){
}
... gets e sets ....

}
```

Criação de Pacotes

Pode-se criar um pacote com o mesmo nome de uma das classes

Classes que estão no mesmo pacote podem acessar umas as outras através de composição (declaração de variáveis) sem a necessidade de nenhuma declaração adicional

Neste caso apenas os métodos públicos podem ser acessados da classe que a importa.

Pacotes

- Para utilizar classes definidas em outros pacotes deve-se importar as classes através da palavra chave **import**.

```
import java.lang.String;
```

ou ainda

```
import java.lang.*;
```

Exemplo de Uso Pacote

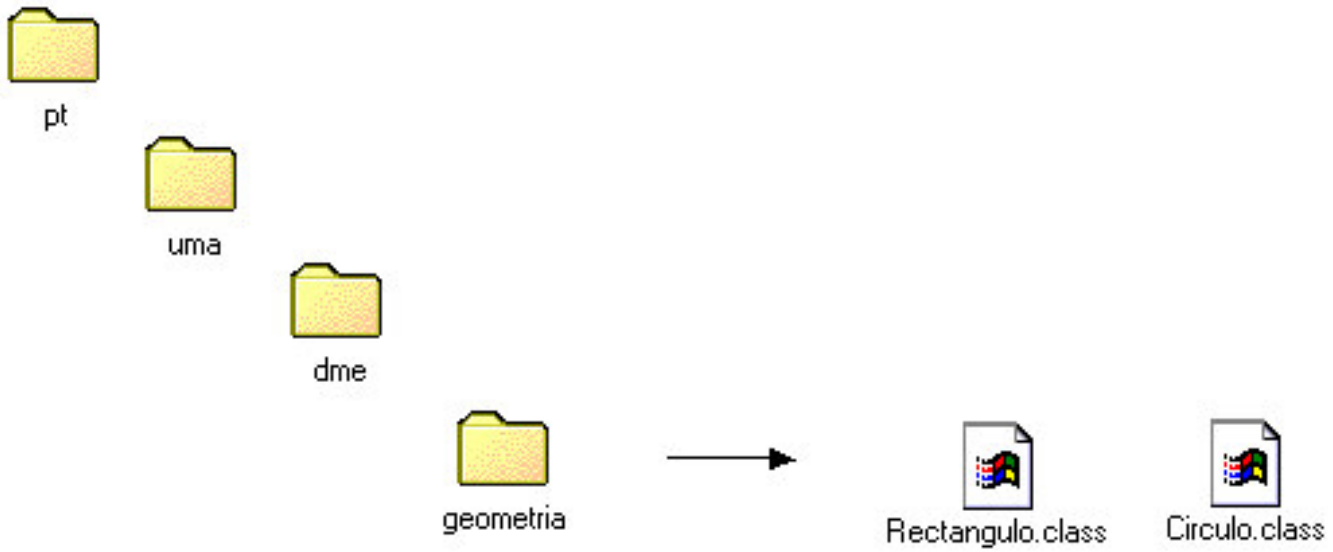
```
Import Calendario.*;

Class DemoCalendario{

Public static void main (String[] args){

Data d1= new Data();
}

}
```

Principais Pacotes em Java

- **java.applet**
  - Classes para criação de Applets.
- **java.awt**
  - Classes para criação de interfaces gráficas.
- **java.io**
  - Classes para gerenciamento de Entradas e Saídas.
- **java.lang**
  - Classes Básicas da linguagem (incluídas automaticamente, não necessita o uso do import).
- **java.net**
  - Classes para trabalho em rede.
- **java.util**
  - Classes de utilitários (Data, Dicionário, Pilha, Vetor, Random, etc.)
- **java.sql**
  - Classes para manipulação de Banco de Dados

Nomeação de Pacotes

- ***Para evitar conflitos de nomes***
  - seu *URL* ou endereço de *email* para identificar o seu *package*.
  - ***package pt.uma.dme.geometria;***



Packages e Visibilidade

- Classes
  - **public** - a classe é acessível dentro do *package* a que pertence e a qualquer código Java que tenha acesso a esse *package*.
  - "default" - quando não é indicada a visibilidade então a classe apenas pode ser acessada do interior do *package* a que pertence.

Packages e Visibilidade

- Métodos

	Acessível dentro de			
Visibilidade:	Classe	Próprio *package*	Subclasses	Outros *packages*
public	**X**	**X**	**X**	**X**
private	**X**	**-**	**-**	**-**
protected	**X**	**X**	**X**	**-**
"default"	**X**	**X**	**-**	**-**

Vantagens de Uso de Pacotes

- Evitam-se conflitos entre nomes de classes
- As classes são mais fáceis de encontrar
- Há um maior controle no acesso às classes

Pacotes

Exemplo: Data, Hora, DataHora

Pacote DataHora

1. Criar diretório DataHora (mesmo nome do pacote) contendo todas as classes do pacote
2. O diretório deve ser criado no caminho de procura de classes java
3. Hierarquia em muitos níveis, refletindo o nome do domínio da instituição em ordem reversa:

Ufba.graduaçãoComputação.LPOO → nome do pacote

Ufba\graduaçãoComputação\LPOO → diretório do pacote

Pacotes

```
package DataHora;
public class Hora  {
byte hora;
byte minuto;
byte segundo;
public Hora(byte h,byte m,byte s)    {
    hora = h; minuto = m; segundo = s;
    }

public String toString()    {
    return hora+":"+minuto+":"+segundo;
    }

 } // fim da classe Hora
```

DataHora/Hora.java

Pacotes

```
package DataHora;
public class DataHora  {
  Data estaData;
  Hora estaHora;
public DataHora(byte h,byte min,byte s,byte d,byte m,short a)    {
    estaData = new Data(d,m,a);
    estaHora = new Hora(h,min,s);    }

public String toString()    {
    ....
    }

 } // fim da classe DataHora
```

Não há conflito: pacote e classe com o mesmo nome.

Usa diretamente as classes do pacote.

DataHora/DataHora.java

Pacotes

```
import DataHora.*;

class DemoDataHora  {

  public static void main(String[] argumentos){
    Hora meiodia = new Hora((byte)12,(byte)00,(byte)00);
    Data hoje  = new Data((byte)11,(byte)5,(short)2016);
    DataHora agora = new DataHora((byte)22,(byte)35,(byte)00,
                                  (byte)11,(byte)5,(short)2016);
    System.out.println(meiodia);
    System.out.println(hoje);
    System.out.println(agora);
    ....
    } // fim do método main

  } // fim da classe DemoDataHora
```

Usa o pacote DataHora: não precisa estar no mesmo diretório

Importa todas as classes do pacote

DemoDataHora.java

Exercício

- Considere uma classe Colaborador com 3 atributos: nome, matricula e salario. Seu construtor recebe estes 3 parâmetros e ela possui todos os métodos acessores (***get*** e ***set***) para seus atributos.
- Existe um método público *aumentarSalario()* que aumenta em 5% o valor do salário.
- Faça duas subclasses de Colaborador: Gerente e Operador. Que utilizam, como base, o mesmo construtor da superclasse.
  - O método *getSalario()* da classe Gerente será sobrescrito, retornando o salário e mais 10% do seu valor a título de gratificação de chefia.
  - Para a classe Operador, o *getSalario()* também será sobrescrito e retornará o valor do salário adicionado de 30% a título de periculosidade.

ATENÇÃO: Todas estas classes devem estar no mesmo pacote.

CONTINUA → →

Exercício Cont

- Escreva um programa em uma **classe dentro de um outro pacote** que será dividido seu processamento em 3 etapas:
  I. Vai receber valores de nome, matrícula e salário para 10 funcionários. Para cada funcionário, deve ser informado o seu cargo (se colaborador, gerente ou operador)
     - Todos os valores tem que ser informados EXCLUSIVAMENTE através do ***nextLine()*** da classe Scanner
       - Não utilize *nextDouble()*, *nextInt()* ou método similar
  II. Após a entrada de todos os 10, execute o método *aumentarSalario()* para todos
  III. Por último imprima cargo (se colaborador, gerente ou operador), a primeira letra do nome e o salário atual de todos esses 10 funcionários.

Comentários

- Documentação do código
- Detalhes de implementação internos do software:
  - Dirigida para os programadores;
    - Normalmente colocado linha a linha

Comentários

- Quando se deve comentar um código?
  - O que for relevante.
  - Quando alguma ideia obscura surgir na sua mente.

- Quantas linhas de comentários devem ser adicionada para cada linha de código?
  - Quantas forem necessárias para que o leitor precise ler a menor quantidade de código possível.
  - Em média, uma ou duas linhas de comentário.
- Note
  - Comentários não são uma redação

Comentários X Documentação

- Documentação de Software
  - Mais ampla que comentários
  - Aspectos Gerais
    - Externos ao código
  - Integradores de software;
  - Distribuidores de componentes.
  - Arquitetos

Documentação em Java

- Javadoc é uma ferramenta que utiliza anotações e tags HTML para organizar gerar a documentação de um programa.

- Tags são adicionadas aos comentários para transformá-los em documentação

Documentação em Java

- Quem já usou o JavaDoc?
  - Exemplo Java doc ArrayList

Documentação em Java

- Para inserir comentário utiliza-se:
  - Comentário de linha
    - // (...)
  - Comentário de bloco
    - /* (...) */
- Para inserir documentação utiliza-se:
  - /** (...) */

Documentação em Java

- A primeira frase de um bloco de documentação é utilizado como um sumário de todo o bloco.

- Tags em HTML
  - Amplo suporte
  - Devem ser utilizadas durante a documentação:
  - Adicionar imagens, mudar cores, alterar fonte/formatação, etc...

Anotações em Java

Tag	Objetivo	Obrigatório	Exemplo
@author	Autor da classe ou interface	Classes e interfaces	@author John Logan
@version	Versão da classes ou interface	Classes e interfaces	@version 2.1
@param	Descreve um parâmetro de método ou construtor	Métodos e construtores	@param nome Nome do Cliente.
@return	Descreve o retorno de um método	Métodos	@return Uma lista com todos os clientes.
@throws	Descreve as exceções geradas pelo método	-	@throws IOException Caso o arquivo não exista
@see	Indica classe, método, etc. possui informações relacionadas	-	@see ContaBancaria
@since	Indica desde qual versão do software essa classe, método, etc. está presente	-	@since 1.1