

Pacotes

Tipos de Módulos em Java

- Classes e interfaces
 - agrupam definições de métodos, atributos, construtores, etc.
 - definem tipos
- Pacotes
 - agrupam definições de classes e interfaces relacionadas
 - estruturam sistemas extensos, facilitando a localização das classes e interfaces
 - oferecem um nível mais alto de abstração: há mais classes do que pacotes

Pacotes e Diretórios

- As classes de um pacote são definidas em arquivos com o mesmo cabeçalho:
`package nomeDoPacote;`
- Cada pacote é associado a um diretório do sistema operacional:
 - Os arquivos `.class` das classes compiladas do pacote são colocados neste diretório
 - É recomendável que o código fonte das classes do pacote também esteja neste diretório

Nomes de Pacotes

- O nome de um pacote é parte do nome do seu diretório associado: o pacote

`qualiti.banco.conta`

deve estar no diretório

`c:\java\qualiti\banco\conta`

assumindo que o compilador Java foi informado para procurar pacotes em

`c:\java`

Pacotes e Subdiretórios

- Subdiretórios não correspondem a "subpacotes". São subdiretórios como outros quaisquer
- Por exemplo, não existe nenhuma relação entre `exemplos` e `exemplos.banco`:

```
package exemplos;  
import exemplos.banco.*;  
/*...*/
```

```
package exemplos.banco;  
/*...*/
```

Pacotes e modificadores de acesso

- ▶ `public`
 - Elementos com este modificador podem ser utilizados (são visíveis) em qualquer lugar, mesmo em pacotes diferentes
- ▶ `protected`
 - Elementos com este modificador só podem ser utilizados no pacote onde são declarados, ou nas subclasses da classe onde são declarados
- “friendly” (sem modificador)
 - Elementos com nível de acesso default só podem ser utilizados no pacote onde estão declarados
- ▶ `private`
 - Elementos declarados com este modificador só podem ser utilizados na classe onde estão declarados

Reuso de Declarações

- As declarações feitas em um arquivo são visíveis em qualquer outro arquivo do mesmo pacote, a menos que elas sejam **private**
- Qualquer arquivo de um pacote pode usar as definições **visíveis** de outros pacotes, através do mecanismo de importação de pacotes

Importação de Pacotes

► Importação de definição de tipo específica:

```
package segundo.pacote;  
    import primeiro.pacote.NomeDaClasse;  
    /*...*/
```

• Importação de todas as definições de tipo públicas:

```
package segundo.pacote;  
    import primeiro.pacote.*;  
    /*...*/
```


Importação de Pacotes: Detalhes

- Tanto `NomeDaClasse` quanto `primeiro.pacote.NomeDaClasse` podem ser usadas no corpo de classes pertencentes a `segundo.pacote`
- Em `segundo.pacote`, não pode ser definida uma classe com o nome `NomeDaClasse`, caso a importação tenha sido específica

Importação de Pacotes: Mais Detalhes

```
package segundo.pacote;  
  
public class NomeDaClasse {  
    /*...*/  
}  
  
package primeiro.pacote;  
  
public class NomeDaClasse {  
    /*...*/  
}
```

Os exemplos 1 e 3 apresentam
problemas!

1

```
import segundo.pacote.*;  
import primeiro.pacote.*;
```

2

```
import segundo.pacote.NomeDaClasse;  
import primeiro.pacote.*;
```

3

```
import segundo.pacote.NomeDaClasse;  
import primeiro.pacote.NomeDaClasse;
```

Estruturando Aplicações com Pacotes

- Agrupar classes relacionadas, com dependência (de implementação ou conceitual) entre as mesmas
- Evitar dependência mútua entre pacotes:

```
package a;  
import b.*;  
/*...*/
```

```
package b;  
import a.*;  
/*...*/
```

Estruturando Aplicações com Pacotes

- Estruturação típica de um sistema de informação:
 - Vários pacotes para as classes e interfaces de GUI, um para cada conjunto de telas associadas
 - Um pacote para a classe fachada e exceções associadas
 - Um pacote para cada coleção de negócio, incluindo as classes básicas, coleções de dados, interfaces, e exceções associadas
 - Um pacote (sistema).util contendo classes auxiliares de propósito geral


Pacotes da Biblioteca Padrão de Java

- Acesso a Internet e WWW (`java.net`)
- Applets (`java.applet`)
- Definição de interfaces gráficas (`java.awt`)
- Suporte a objetos distribuídos (`java.rmi`)
- Interface com Banco de Dados (`java.sql`)
- Básicos: threads e manipulação de strings (`java.lang`), entrada e saída (`java.io`), coleções e utilitários de propósito geral (`java.util`)
- Dezenas de outros...

APIs de Java

APIs da Plataforma Java 2 Standard Edition v1.4

Swing		AWT	
Sound	Input Methods	Java 2D	Accessibility
RMI	JDBC	JNDI	CORBA
XML	Logging	Beans	Locale Support
Preferences	Collections	JNI	Security
Lang	Util	New I/O	Networking

-  **User Interface Toolkits**
-  **Integration APIs**
-  **Core APIs**

Pacote java.lang

- É o principal pacote de Java
- Contém as classes fundamentais da plataforma
- É importado automaticamente em todas as classes criadas
- Classes essenciais:
 - Object, Throwable, Exception, String, Thread, Runnable, Math, System, Runtime

Classe Object

- É a superclasse de todas as classes de Java
- É a única classe sem superclasse
- **boolean equals(Object obj):** verifica se dois objetos são iguais
- **String toString():** retorna um String descrevendo o objeto
- **Object clone():** retorna uma cópia do objeto (mas só se a interface Cloneable for implementada)

A maioria dos métodos de Object
é redefinida nas subclasses

Throwable e Exception

- Throwable é a superclasse de todas as exceções
 - Exception herda de Throwable. É a classe que deve ser usada como base para a definição de novas exceções
-
- **String getMessage():** retorna a mensagem encapsulada na exceção (retorno pode ser nulo)
 - **void printStackTrace():** mostra a pilha de exceção (a seqüência de chamadas de métodos que resultaram na exceção)

String

- Encapsula cadeias de caracteres e muitas operações de manipulação
- Vista detalhadamente na aula sobre Strings e Arrays

Thread e Runnable

- A classe Thread e a interface Runnable são essenciais para a implementação de aplicações com concorrência em Java

System

► Define uma interface padrão, independente de plataforma, para recursos do sistema operacional:

- **out:** saída padrão
- **in:** entrada padrão
- **currentTimeMillis():** hora atual em milisegundos desde 1 de Janeiro, 1970, GMT
- **arrayCopy(Object origem, int origem_pos, Object destino, int destino_pos, int comprimento):** copia um array ou parte de um array para outro
- **exit():** força a saída da aplicação

Pacote java.util

- Define classes e interfaces de coleções e outras classes utilitárias
- Coleções:
 - Collection
 - List
 - Set, SortedSet
 - Vector, Stack
 - ArrayList, LinkedList
 - Map
 - Hashtable
 - Arrays
 - Iterator

Classes utilitárias

- Date, Calendar e GregorianCalendar
- Timer e TimerTask
- Random
- Properties
- StringTokenizer