Unidade 1 – Parte 3

PACKAGES

Um pacote (*package*) é uma ferramenta organizacional fornecida com a linguagem Java. Conceitualmente é um agrupamento de classes e interfaces relacionadas. Pacotes são bastante utilitários. Além disso, permitem que sejam definidos os membros que serão protegidos (eles serão públicos às classes dentro do mesmo pacote, enquanto privados para aquelas classes que estão fora do pacote).

Os pacotes são importantes para implementar bibliotecas de código. Depois que temos um pacote, podemos acessar suas classes e interfaces de várias formas: uma delas é usando o nome completo de uma classe. Digamos que implementamos um pacote Animais e que ele contenha duas classes: um cachorro e um gato. Se quisermos criar uma nova instância da classe cachorro, podemos usar o seguinte código:

```
Animais.cachorro lassie = new Animais.cachorro( );
```

O acesso das classes com seus nomes completos não é necessária. A declaração import do Java facilita isso. Quando importa um pacote, você tem uma convenção de atalho para suas classes e interfaces. Vamos importar a classe cachorro do pacote Animais e ver o que ganhamos:

```
import Animais.cachorro;
class Zoologico
{
  public static void main(String args[])
```

```
  {
    cachorro lassie = new cachorro();
  }
}
```

Essa opção é significativamente mais fácil e exige menos digitação. É assim que você vai acessar todo o código que o Java fornece. Primeiro você importa os pacotes que deseja usar e depois usa a forma abreviada para os nomes de classe e interface.

A API Java é uma coleção de várias dezenas de classes, interfaces e exceções prontas. Dentro da API, essas classes, interfaces e exceções estão agrupadas em pacotes, cada um com sua funcionalidade. A seguir tem-se uma descrição de alguns pacotes básicos da API Java:

PACOTE	**DESCRIÇÃO**
Java.io	Manipulação das entradas e saídas do programa.
Java.awt	Permitem escrever Interfaces Gráficas de Usuário
Java.applet	Contém classes e interfaces que ativam as applets
Java.lang	Contém os elementos centrais da linguagem Java, tais como Object, String, Exception e Thread
java.util	Contém várias classes de utilitários que tornam mais fácil a sua vida, tais como as estruturas de dados mais usadas

O pacote java.lang é tão essencial que o compilador sempre o importa.

Abaixo um exemplo da utilização de um pacote da API Java.

```
import java.sql.*;

class TodasTabelas

{

        public static void main(String[] args) throws Exception

        {

                Properties p = new Properties();

                p.put("user", "borg");

        }
}
```

APIs JAVA

As APIs Java vem sendo constantemente atualizadas. Os principais pacotes padrões oferecidos são:

API	DESCRIÇÃO
Java.applet	Classes para a criação de applets.
Java.awt	Classes para a criação de interfaces com o usuário e para criação de gráficos e imagens.
Java.beans	Classes para o desenvolvimento de *beans* (componentes)
Java.io	Classes para tratamento de entrada e saída de dados, serialização de objetos e tratamento de arquivos.
Java.lang	Classes fundamentais para a programação Java (importada por default)

Java.math	Classes para tratamento de precisão matemática.
Java.net	Classes para implementação de programas em rede.
Java.rmi	Classes para programação remota (RMI – puramente Java).
Java.security	Classes para implementar aplicações seguras.
Java.sql	Classes para acesso e processamento de dados armazenados em uma base de dados (geralmente relacional).
Java.text	Classes para formatação de textos, datas, números e mensagens independente da linguagem utilizada.
Java.util	Classes para trabalhar com as várias estruturas de dados existentes, datas e horários, separadores de string e outros.
Javax.crypto	Classes para trabalhar com criptografia.
Javax.print	Classes para trabalhar com impressão.
Javax.sound.midi	Classes para tratamento de MIDI (Musical Instrument Digital Interface)
Javax.swing	Nova implementação da interface gráfica com o usuário.
Javax.xml	Classes para trabalhar com XML.
Org.omg.CORBA	Classes para programação distribuída (CORBA)
Org.w3c.dom	Classes para Document Object Model (DOM)
Org.xml.sax	Classes para Simple API for XML (SAX)

COLLECTIONS

Uma *collection* permite que um grupo de objetos possa ser tratado como uma simples unidade. Objetos arbitrários podem ser armazenados, recuperados e manipulados.

O pacote *java.util* prove a implementação da maioria das estruturas conhecidas, como, por exemlpo, tabelas hash e listas encadeadas.

Lists

Lists é uma collection que mantém seus elementos em ordem (também chamado de sequência) e não contém elementos duplicados. Esta classe possui métodos para utilizar-se das operações necessárias em uma lista como: acesso por posição numérica, procura na lista, operações em partes da lista e outros.
Em uma lista não vazia, o primeiro elemento tem o índice 0 (zero) e o último tem o índice *size( )* – 1. Qualquer acesso fora desse intervalo gera uma exceção do tipo *IndexOutOfBoundException*.

Principais métodos:

- get (int)
- set (int, Object)
- indexOf (Object)
- lastIndexOf (Object)
- add (Object)
- remove (Object)

Exemplo de utilização:

```
List p = new LinkedList();
p.add("Primeiro");
p.add("Segundo");

for (int i=0; i < p.size(); i++)
{
      System.out.println("Valor = "+p.get(i));
}
```

Vector

A classe *Vector* implementa arrays com tamanho dinâmico. A classe vector é thread-safe.

Principais métodos:

- add (Object)
- clear ( )

- contains (Object)
- indexOf (Object)
- insertElementAt (Object, int)
- remove (int)
- addElement(Object)
- elementAt(int)

Exemplo de utilização:

```
Vector v = new Vector();
v.addElement(“Elemento 1”);
v.addElement(“Elemento 2”);
System.out.println(“Elemento 1 ->”+v.elementAt(0));
```

HashMap

Um *map* mapeia valores através de chaves. Um mapa não permite chaves duplicadas, ou seja, as chaves são únicas e cada chave aponta para apenas um valor, implementando o que chamamos de *single-value maps*.

Principais métodos:

- clear ( )
- get (Object chave)
- isEmpty ( )
- put (Object chave, Object valor)
- size ( )

Abaixo se demonstra um exemplo:

```
HashMap telefones = new HashMap();
telefones.put("Pizza","808221122");
telefones.put("Locadora","838383");
String locadora=(String)telefones.get("Locadora");
String numero=(String)telefones.get("Pizza");

System.out.println("locadora->"+locadora);
System.out.println("pizza->"+numero);
```

Os exemplos acima são apenas algumas das *Collections* disponíveis no Java. Mais informações sobre estas e outras *Collections* podem ser acessadas no site (https://www.java.com/pt-BR/).

Acessando dados sem usar índices (ITERATOR)

Uma forma de navegar em coleções sem usar índices é usando um "iterator", que diz: "me dê o próximo, me dê o próximo, me dê o próximo, ..." até acabar.
Abaixo um exemplo:

```
Vector <String> v = new Vector <String>();
v.addElement("Primeiro");
v.addElement("Segundo");
v.addElement("Terceiro");

Iterator it = v.iterator();
   while (it.hasNext())
{
      System.out.println("Elemento = "+it.next());
}
```