DANIEL ABELLA
www.daniel-abella.com

Daniel Abella

Profissional PMP, PMI-ACP e CSM, Gerente de Projetos no Virtus & UniFacisa e Professor do Curso de Sistemas de Informação da UniFacisa. Autor dos livros Gestão A3 (Ágil) e Scrum Arretado.

Conteúdo:

Capítulo 5

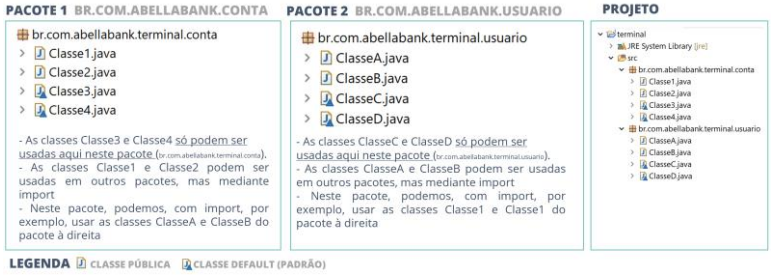
Pacotes, Listas e Enums

Livro Java do Zero (Uma Viagem ao Mundo Java)

01 Pacotes

- Imagine centenas de classes (que no fim, são arquivos com extensão .java para o código fonte e .class para os compilados) sem nenhuma organização. **É nesta situação que entra os pacotes (em inglês, package), atuando como pastas para organizar todos componentes de um programa em Java.**
- Para definir o pacote, na primeira linha do código, colocamos package <nome-do-pacote>; No quadrante à esquerda intitulado "código fonte", temos as duas classes do nosso sistema.

USO DO IMPORT



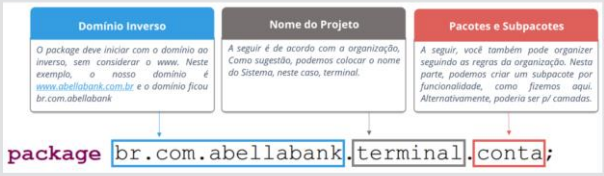
Memorize!

- Classes com modificador default (package) só podem ser usadas (em geral, instanciadas) de classes do mesmo pacote, porém podem usar outras classes default e public do mesmo pacote sem import e classes public de outros pacotes mediante import;
- Classes com modificador public podem ser usadas (em geral, instanciadas) por classes do mesmo pacote (independente do modificador) sem import e podem usar outras classes default e public do mesmo pacote sem import e classes public de outros pacotes mediante import.

CONVENÇÃO PARA PACKAGES

De acordo com a convenção Java, os nomes dos pacotes devem ser escritos em letras minúsculas, usando nomenclatura em estilo camelCase, na qual os nomes devem ser descritivos, representando o propósito ou domínio do código contido no pacote, e a estrutura hierárquica deve refletir a organização dos componentes do projeto.

Geralmente, conforme imagem a seguir, adotamos a convenção de nomenclatura em estilo de domínio reverso, começando com o domínio da empresa invertido, seguido por subníveis representando diferentes partes do projeto.



PACOTE JAVA.LANG

Usamos adoidado as classes String e System (usada para o System.out.println) nos códigos e não fizemos um único import. Isto se deve ao fato que, as classes que estão incluídas no pacote java.lang (que inclui as duas supracitadas e demais classes essenciais do Java), são importadas automaticamente.

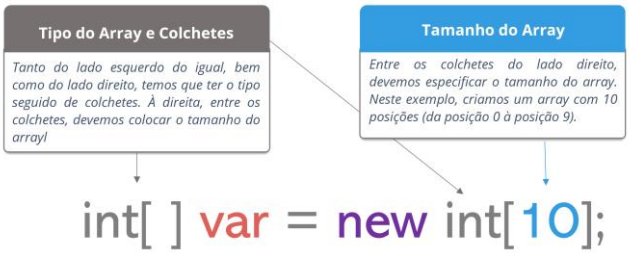
PACOTES (PACKAGES) EM JAVA

02 Imports

- Como vimos na seção anterior, para usarmos classes com modificador public de outros pacotes, precisamos usar import.

03 Arrays

- Arrays são usados em programação porque oferecem uma maneira eficiente de armazenar e acessar conjuntos de dados, permitindo uma manipulação rápida e organizada dos elementos. Eles fornecem estruturas de dados compactas e acessíveis que facilitam a implementação de algoritmos de classificação, pesquisa e outras operações em massa, tornando o código mais eficiente e escalável.



int[] array1 = new int[3];

double[] array2 = new double[3];

String[] array3 = new String[3];

- Abaixo as maneiras de criar um array em Java.

```
1  class TesteArray {
2
3      public static void main(String[] args) {
4
5          int[] numeros1 = new int[]{1, 8, 12};
6
7          int[] numeros2 = new int[3];
8          numeros2[0] = 1;
9          numeros2[1] = 8;
10         numeros2[2] = 12;
11
12         int[] numeros3 = {1, 8, 12};
13
14         String[] nomes1 = new String[]{"João", "Maria", "Pedro"};
15
16         String[] nomes2 = new String[3];
17         nomes2[0] = "João";
18         nomes2[1] = "Maria";
19         nomes2[2] = "Pedro";
20
21         String[] nomes3 = {"João", "Maria", "Pedro"};
22     }
23 }
```

OPÇÃO UM

OPÇÃO DOIS

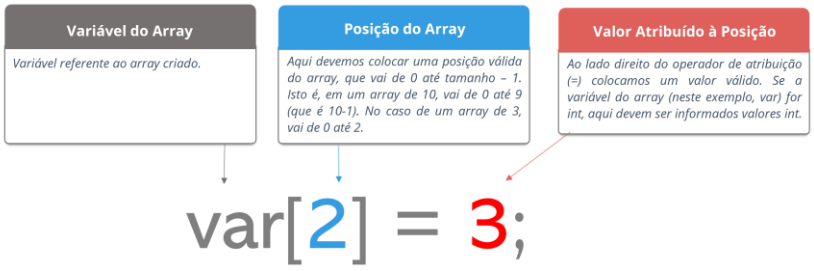
OPÇÃO TRÊS

OPÇÃO UM

OPÇÃO DOIS

OPÇÃO TRÊS

- A seguir, como preencher uma posição do Array. Agora que sabemos como declarar arrays, vamos entender como acessar as caixinhas, preenche-las e colocarmos coisas lá. Na linha 5, declaramos o array de inteiros com 3 posições, ou seja, de 0 até 2.




```
1  class TesteArray {
2
3      public static void main(String[] args) {
4
5          int[] numeros = new int[3];
6          System.out.println(numeros[2]);
7          numeros[2] = 3;
8          System.out.println(numeros[2]);
9      }
10 }
```

- Para saber o tamanho de um array, devemos usar o atributo length.
- Entretanto, o uso mais comum do length é associado ao for tradicional. No código abaixo, a variável i é usada como o índice do array e é inicializada com 0. Usamos, na linha 7, o atributo length (em tradução, comprimento), existente em Arrays, que informa a quantidade de elementos, neste caso, 3. Então, considerando a varível i = 0 e a condição i < numeros.length, o i percorrerá de 0 até 2.

```
1  class TesteArray {
2
3      public static void main(String[] args) {
4
5          int[] numeros = { 1, 8, 12 };
6
7          for (int i = 0; i < numeros.length; i++) {
8              System.out.println("O elemento " + (i+1) + " da lista é " + numeros[i]);
9          }
10     }
11 }
```

- O enhanced for loop, também conhecido como for each, é uma forma simplificada de iterar sobre os elementos de um array, permitindo permite percorrer todos os elementos de um array sem a necessidade de controlar explicitamente um índice, conforme podemos verificar no exemplo abaixo.

```
1  class TesteArray {
2
3      public static void main(String[] args) {
4
5          int[] numeros = { 1, 8, 12 };
6
7          for (int num : numeros) {
8              System.out.println(num);
9          }
10     }
11 }
```

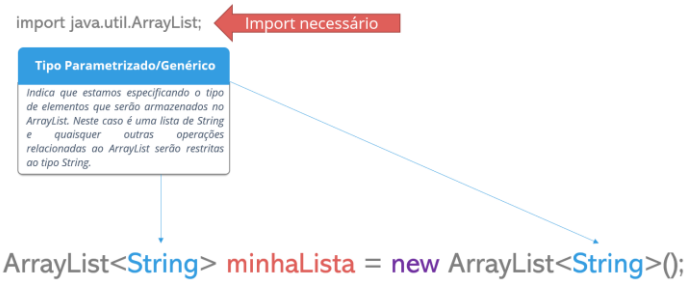
- A seguir a sua estrutura de funcionamento.

LOOP ENHANCED FOR

```
for(tipo var; elemento) {
    //linhas de código
}
```

04 ArrayList

- Ao contrário dos arrays, que têm tamanho fixo, ArrayList são flexíveis, pois permite adicionar, remover e modificar elementos facilmente durante a execução do programa. Além disso, ArrayList oferece métodos extremamente simples para buscar, ordenar e percorrer os elementos.
- A seguir, temos como criar um ArrayList. Inicialmente temos a necessidade de importar a classe para posterior uso. E, para criarmos um ArrayList, fazemos uma instanciação normal. Entretanto, você pode ter percebido os sinais de menor e maior "<>", conhecido como operador diamante, que entre os sinais, especificamos o tipo de elementos que serão armazenados no ArrayList.



- A seguir, temos um bom exemplo envolvendo as principais funcionalidades de um ArrayList. No passo 1, criamos um ArrayList seguindo as indicações supracitadas, enquanto que no passo 1, chamamos o método add para adicionar Strings ao ArrayList. Note que, como especificamos que o ArrayList é de String, no método add, apenas podemos adicionar Strings.

```java
import java.util.ArrayList;

class TesteArrayList {

    public static void main(String[] args) {

        ArrayList<String> nomes = new ArrayList<String>();

        nomes.add("Daniel");
        nomes.add("Nathaly");
        nomes.add("Lucas");
        nomes.add("Arthur");

        String primeiroNome = nomes.get(0);
        System.out.println(primeiroNome);

        for (String nome : nomes) {
            System.out.println(nome);
        }

        for(int i = 0; i < nomes.size(); i++) {
            System.out.println(nomes.get(i));
        }

        boolean contemDaniel = nomes.contains("Daniel");
        System.out.println(contemDaniel);

        nomes.remove("Nathaly");
        nomes.remove(0); //removendo "Daniel" (posição 0)

        int tamanho = nomes.size();
        System.out.println(tamanho);
    }
}
```

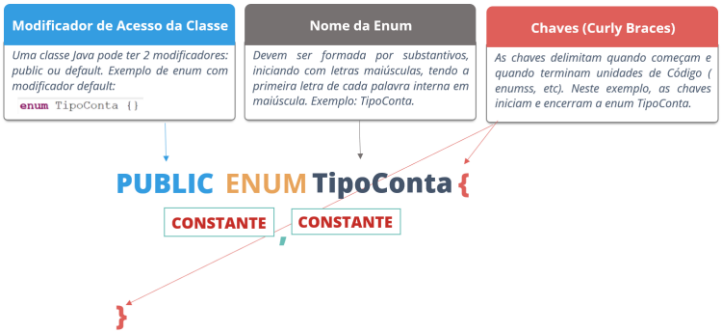
- No passo 3, chamamos o método get, que obtém um elemento de uma data posição, na qual obtemos o primeiro elemento do ArrayList, que como os Arrays, a primeira posição inicia-se com 0. O passo seguinte (4), usamos o enhanced for (for each) para iterar as Strings, nada diferente do que já vimos.
- Entretanto, no passo 5, usamos o for tradicional e verificamos grandes diferenças com relação ao array. Primeira delas é que, ao invés de length, temos um método chamado size() que botém o tamanho da lista. E, ao invés de usarmos os colchetes (como nomes[i]) como em Arrays, em ArrayList usamos o método get(posição).
- Agora, no passo 6, usamos o método contains para saber se um dado elemento ("Daniel") está contido na lista nomes. Este método, como esperado, retorna um boolean, podendo ser true ou false.
- No passo 7, apresentamos duas maneiras de remover o elemento de um ArrayList. A primeira delas, na linha 28, é remover pelo valor, ou seja, informamos qual valor queremos remover da lista. Por outro lado, a outra opção é a remoção pela posição, isto é, informamos a posição (índice) do elemento que queremos remover. Não existe melhor ou pior opção, mas a opção que atende a sua necessidade em um dado momento. Por fim, no passo 8, verificamos que, com o uso do método size, obtemos o tamanho da lista.

05 Wrappers

Tipo Primitivo	Wrapper
boolean	Boolean
byte	Byte
char	Character
float	Float
int	Integer
long	Long
short	Short
double	Double

06 Enumerations

- Enumerations (enum) são um tipo de dado especial que permitem definir um conjunto de constantes relacionadas, sendo usadas geralmente para representar valores fixos e bem-definidos, sendo úteis em situações onde se deseja restringir as opções de escolha a um conjunto limitado de valores relacionados. Por exemplo, podemos criar uma enum TipoConta, que pode ter como valores CONTA_CORRENTE e CONTA_POUPANCA, representando os possíveis valores referentes a um tipo de uma conta. A seguir temos a estrutura para criação de uma enum.



- O exemplo referente a TipoConta é apresentado a seguir. Na classe Conta, a variável tipoConta, ao invés de int, tem como tipo o TipoConta. Por fim, no método main da classe TesteComEnum, nas linhas 5 e 7, informamos no construtor os tipos válidos. A forma de informar um valor é sempre da seguinte forma TipoEnum.CONSTANTE, tal como fizemos em TipoConta.CONTA_CORRENTE.
- Complementarmente, as linhas 6 e 8 que estão comentadas, pois agora, com uso da enum, não podemos passar outros valores que não os 2 disponíveis (TipoConta.CONTA_CORRENTE e TipoConta.CONTA_POUPANCA).

```java
public enum TipoConta {

    CONTA_CORRENTE, CONTA_POUPANCA
}
```

```java
public class Conta {

    private int numeroAgencia;
    private int numeroConta;
    private String titular;
    private double saldo;
    private TipoConta tipoConta;

    //getters e setters omitidos

    public Conta(int numeroAgencia, int numeroConta,
            String titular, double saldo, TipoConta tipoConta) {
        this.numeroAgencia = numeroAgencia;
        this.numeroConta = numeroConta;
        this.titular = titular;
        this.saldo = saldo;
        this.tipoConta = tipoConta;
    }
}
```

```java
class TesteComEnum {

    public static void main(String[] args) {

        Conta conta1 = new Conta(1,123,"Daniel",1000,TipoConta.CONTA_CORRENTE);
        //Conta conta2 = new Conta(1,124,"Abella",1000,1);
        Conta conta3 = new Conta(1,124,"Abella",1000,TipoConta.CONTA_POUPANCA);
        //Conta conta4 = new Conta(1,124,"Abella",1000,3);
    }
}
```