CONVENÇÃO DE CÓDIGO JAVA

Eligiane Ceron - Abril de 2012 – Versão 1.0

Conteúdo

Considerações iniciais 2
Introdução 2
Extensão de arquivos 2
Arquivos de código Java 2
Comentários iniciais 2
Declarações de pacotes e importações 2
Declaração da classe/interface 3
Identation (recuo) de código 3
Quebras de linha 3
Sobre comentários 4
Comentários de implementação 5
Comentários de documentação 6
Declarações 6
Espaços em branco 6
Linhas em branco 6
Espaços em branco 6
Convenções de Nomenclatura 7
REFERÊNCIAS 7

Copyright 1995-1999 Sun Microsystems, Inc. All rights reserved. Used by permission.

Considerações iniciais

Este texto é uma tradução adaptada da convenção de código Java publicada pela Sun e mantida pela Oracle. Você pode ler o material original no link que se encontra no tópico "Referências".

Introdução

A padronização de código é conveniente devido a vários aspectos. Segundo estatísticas, cerca de 80% do tempo gasto com desenvolvimento de software é referente à manutenção do sistema. E dificilmente os softwares desenvolvidos são mantidos pelos seus autores. A padronização ajuda a deixar o código mais legível, facilitando assim o entendimento do mesmo por parte de novos integrantes da equipe de desenvolvimento.

Extensão de arquivos

Os softwares java usam dois tipos de arquivos: os arquivos de código, que recebem a extensão *.java* e os *bytecodes* que recebem a extensão *.class*. O *bytecode* é gerado pelo compilador Java.

Arquivos de código Java

Um arquivo Java é composto por seções que devem ser separadas por linhas em branco, podendo utilizar também comentários para identificar cada uma.

Cada arquivo de código java deve conter uma única classe ou interface pública. Arquivos com mais de duas mil linhas de código devem ser evitados, pois ficam muito pesados.

A organização básica de um arquivo de código Java deve obedecer à seguinte ordem:

1. Comentários iniciais
2. Declarações de pacotes e importações
3. Declaração da classe/interface

Comentários iniciais

Todo arquivo de código deve iniciar com comentários de início, que devem conter informações a respeito da classe/interface, como por exemplo, nome da classe, informações de versão, data e autor.

Exemplo:

```
/**
* Nome da classe
*
* @version v 2.0 Janeiro/2012
* @author CERON, Eligiane
*/
```

Declarações de pacotes e importações

A primeira linha de código após os comentários iniciais deve ser a declaração do pacote (se necessário), e em sequência as declarações de importações.

Exemplo:

```
Copyright 1995-1999 Sun Microsystems, Inc. All rights reserved. Used
by permission.
```

```
package pacote1;

import java.util.Scanner;
import java.util.ArrayList;
```

Declaração da classe/interface

A tabela abaixo mostra cada parte de uma classe/interface, na ordem em que elas devem aparecer.

1	Comentário de documentação (/** ..... */)	Observações
2	Declaração da classe/interface	
3	Comentários de implementação da classe (/* .... */)	Se necessário aqui pode ser inserido algum comentário a respeito da classe
4	Atributos estáticos	Inicialmente os públicos, depois os protegidos e por último os privados
5	Demais atributos	Inicialmente os públicos, depois os protegidos e por último os privados
6	Construtores	
7	Demais métodos	Agrupe os métodos por funcionalidade, exemplo: *getters*, *setters*, métodos de validação, métodos de cálculos, etc.

Identation (recuo) de código

Os recuos são utilizados para alinhar visualmente comandos pertencentes a blocos de código. Um bloco é o código envolvido pelos delimitadores { e }, como por exemplo o *if*. A regra geral é abrir o bloco na linha do comando e fechar alinhado a ele.

Exemplo:

```
if (condicao){
     //comandos
}else{
     //comandos
}
```

Normalmente se utiliza 4 espaços como medida padrão da tabulação, porém algumas vezes é necessário utilizar 8 espaços para obter uma visualização melhor do código. Devem ser evitadas linhas com mais de 80 caracteres.

Quebras de linha

Quando uma expressão não couber numa única linha, quebrá-la de acordo com as seguintes regras básicas:

- Após uma vírgula
- Antes de um operador
- Alinhar a nova linha com o início da expressão da linha anterior
- Se as regras acima gerarem código confuso, ou se a linha de baixo ficar colada na margem, use uma tabulação de 8 espaços.

Copyright 1995-1999 Sun Microsystems, Inc. All rights reserved. Used by permission.

Alguns exemplos de quebra de linha numa chamada de método:

```
executarMetodo(parametro1, parametro2, parametro3,
        parametro4, parametro5);//usando 8 espaços
int res = executarMetodo(parametro1, parametro2, parametro3,
          parametro4, paramentro5); //alinhando   com   o   inicio   da
expressão anterior
```

Exemplo de quebra de linha numa expressão aritmética:

```
double res = valor1 * (valor2 + valor3)
             + valor4 / valor5;
```

Para inserir quebras de linha em condições nos ifs, por exemplo, cuide para não alinhar as condições com os comandos, pois eles se misturariam visualmente e tornaria ruim a leitura do código.

Exemplo 1: EVITAR

```
if(condicao1 && condicao2 || condicao3
   && condicao4){
   executarAlgo();
}
```

Exemplo 2: PREFIRA

```
if(condicao1 && condicao2 || condicao3
        && condicao4){
   executarAlgo();
}
```

Sobre comentários

Programas desenvolvidos em Java podem ter dois tipos de comentários: comentários de implementação e comentários de documentação. Os comentários de implementação são os mesmo da linguagem C++, que são delimitados por /* ... */, e //. Os comentários de documentação (conhecido como "comentários doc") são Java-only, e são delimitados por / ** ... * /. Comentários de documentação podem ser extraídos para arquivos HTML usando a ferramenta *javadoc*.

Comentários de implementação são aqueles destinados a comentar o código ou a aplicação em si. Comentários de documentação são destinadas a descrever a especificação do código, a partir de uma perspectiva de implementação livre, para ser lido por desenvolvedores que não necessariamente possuem o código fonte em mãos.

Os comentários devem ser usados para dar uma visão geral do código e fornecer informações adicionais que não está prontamente claro no próprio código. Eles devem conter apenas informações relevantes para a leitura e compreensão do programa. Por exemplo, informações sobre como o pacote correspondente é construído ou em qual diretório ele reside não deve ser incluído como um comentário.

Copyright 1995-1999 Sun Microsystems, Inc. All rights reserved. Used by permission.

Comentários de implementação

Há quatro possíveis formatos de comentário de implementação: bloco, linha, à direita e no fim da linha.

Os comentários de bloco são utilizados para descrever arquivos de código, métodos ou algum algoritmo dentro de um método. Procure sempre inserir uma linha em branco antes do bloco de comentários.

Exemplo de comentário de bloco:

```
/ *
  * Aqui está um comentário de bloco.
  * /
```

Os comentários de linha são para comentários curtos, e devem ser precedidos com uma linha em branco, assim como o comentário de bloco.

```
if (condicao){

     /* comentários */
     executarAlgo();
}
```

Comentários muito curtos podem aparecer na mesma linha do código que descrevem, mas devem ser deslocados o suficiente para separá-los das declarações. Se mais de um breve comentário aparecer em um pedaço de código, todos eles deverão ser recuados de modo a se alinharem.

Exemplo:

```
if (condicao) {
    return valor1;            /* comentário 1*/
} else {
    return outroValor;        /* comentário 2 */
}
```

O delimitador de comentário // pode servir para comentar toda uma linha ou apenas parte dela. Evite utilizá-los quando muitas linhas de comentário são necessárias (prefira o delimitador de bloco). No entando, ele pode ser utilizado em várias linhas para comentar ou anular um trecho de código.

```
if (condicao) {

    // comentário.
    ...
} else {
    return false;          // Explicação aqui.
}
//if (condicao2) {
```

Copyright 1995-1999 Sun Microsystems, Inc. All rights reserved. Used by permission.

```
//
//    return ...
//}
//else {
//    return false;
//}
```

Comentários de documentação

Os comentários de documentação possuem regras específicas e serão abordados em um próximo artigo.

Declarações

Ao declarar variáveis, observe as seguintes regras:

- Faça apenas uma declaração por linha, de modo a incentivar o uso de comentários
- Inicialize apenas uma variável por linha
- Use letras minúsculas e evite caracteres especiais.

Espaços em branco

Linhas em branco

A função das linhas em branco é prover maior legibilidade de código e organização. Utilize duas linhas em branco para separar seções em um arquivo fonte e entre definições de classe e interfaces.

Use uma linha em branco entre métodos, após declarar variáveis, antes de um bloco ou linha de comentário ou mesmo dentro de um bloco de código para separar comandos em comum.

Espaços em branco

Sempre que uma palavra reservada da linguagem for seguida de parênteses, deve conter um espaço separando-os.

```
while (condição){
      //comandos
}
```

Não utilize espaço entre um método e seus parênteses:

```
public void executarAlgo(){
....
```

Use um espaço em branco após a vírgula numa lista de argumentos e nas expressões aritméticas utilize um espaço em branco para separar operandos de seus operadores.

```
a = b + c; //ok!
a=b+c; //evitar!
```

Separe também operadores relacionais das condições

```
if(condicao1 && condicao2){
```

Copyright 1995-1999 Sun Microsystems, Inc. All rights reserved. Used by permission.

Convenções de Nomenclatura

Pacotes	Letras minúsculas sem caracteres especiais	gui
Classes e interfaces	Os nomes de classe devem ser substantivos, com a primeira letra de cada palavra interna em maiúscula. Tente manter seus nomes de classe simples e descritivos. Evite siglas e abreviações (a menos que a sigla seja muito mais usada do que a forma longa, como a URL ou HTML) .	class Pessoa class ContaCorrente
Métodos	Métodos devem ser verbos, em maiúsculas e minúsculas com a primeira letra minúscula, com a primeira letra de cada palavra interna em maiúscula.	calcularAlgo()
Variáveis	Devem iniciar com minúscula. Palavras internas começam com letras maiúsculas. Não deve começar com underline ou $, mesmo que ambos sejam permitidos. Use nomes curtos, mas significativos. O nome deve indicar a intenção da utilização da variável. Os nomes comuns para variáveis temporárias são i, j, k, m, n e para inteiros, c, d, e e para caracteres.	idade nomeCompleto
Constantes	Constantes devem ter todas as letras maiúsculas separadas por underline	MAX_SIZE VALOR_PADRAO

REFERÊNCIAS

ORACLE**.** *CodeConventions for the Java ™ ProgrammingLanguage*. 1999. Disponível em: http://www.oracle.com/technetwork/java/codeconvtoc-136057.html

ORACLE. Java SE Specifications. 2012. Disponível em <http://docs.oracle.com/javase/specs/>

Copyright 1995-1999 Sun Microsystems, Inc. All rights reserved. Used by permission.