Métodos

● O que são métodos

Um método contém uma ou mais instruções. Em um código Java bem escrito, cada método executa apenas uma tarefa. Cada método tem um nome e é esse nome que é usado para chamá-lo. Em geral, podemos dar a um método o nome que quisermos. No entanto, lembre-se de que main() está reservado para o método que começa a execução do programa. (SCHILDT, 2015, p. 107).

Os métodos (conhecidos em C como procedimentos e funções) permitem modularizar programas separando suas tarefas em unidades autocontidas. Dentre os motivos que levam à utilização de métodos, podemos citar: simplificação do problema, maior gerenciabilidade e capacidade de reutilização de software. Métodos existentes podem – e devem – ser utilizados para criar novos programas. (SERSON, 2007, p. 123).

● O que é o método *main*

A palavra main tem um papel especial no Java. Em particular, porque nunca escrevemos um código que explicitamente chame um método main para agir. a palavra main é o nome do método que é chamado para entrar em ação automaticamente quando o programa começa a ser executado. (BURD, 2013, p. 48).

A classe contém o método main() para começar a execução do programa. O método main() é essencial para cada programa Java. Um programa Java pode conter mais de uma classe, mas apenas uma classe pode conter o método main() em um único programa. Uma única classe também pode conter mais de um método. (BUYYA; SELVI; CHU, 2009, p. 43, tradução nossa).

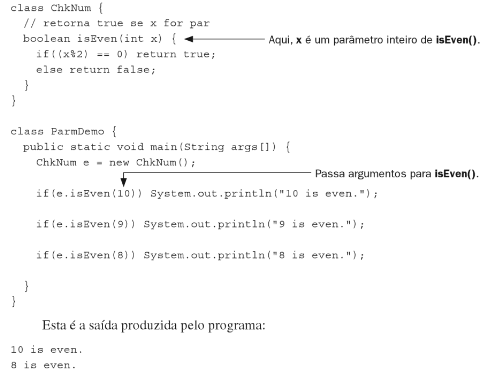
● Tipos de métodos

Os métodos podem ser classificados como métodos retornando um valor e métodos retornando void ou nenhum valor. O método retornando um valor produz um valor do método depois de realizar alguns cálculos na invocação. Métodos retornando um valor são normalmente chamados de funções. Por exemplo, um método pode ser definido para encontrar o quadrado de um número. Esse método terá que ser definido como um método retornando um valor que é o quadrado do número informado. Algumas vezes um método pode não retornar nenhum valor. Por exemplo, se um método é definido para limpar a tela, é necessário escrever o código para mostrar uma página cheia de linhas em branco. Esse método não retornará nenhum valor do método. Contudo, métodos de tipo void realizam algumas ações úteis necessárias. (BUYYA; SELVI; CHU, 2009, p. 182, tradução nossa).

Lembre-se, um método descrito como público significa que ele pode ser acessado de fora da classe. O tipo de retorno identifica o tipo de valor que o método irá retornar. Este pode ser um tipo primitivo ou uma classe. Se nenhuma informação é retornada pelo método, a palavra-chave void é usada para o tipo de retorno. (HOLMES; JOYCE, 2001, p. 94, tradução nossa).

Com retorno

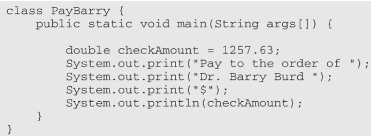
Figura 1 - Exemplo de método com retorno em Java



Fonte: Schildt (2015)

Sem retorno

Figura 2 - Exemplo de método sem retorno em Java



Fonte: Burd (2013)

● Padrão de nomes de métodos em Java

Nomes de métodos seguem as mesmas regras de nomes de campos: devem ser iniciados por uma letra (incluindo \_ e $), devem ser compostos de uma única palavra (sem espaços, vírgulas etc.), podem ser compostos de letras e números e conter acentos. Nomes de métodos não podem ser iguais a nenhuma das palavras reservadas. (SANTOS, 2003, p. 24).

“lowerCamelCase: A primeira letra é minúscula e as primeiras letras das palavras seguintes são maiúsculas”(tradução livre). (BAESENS; BACKIEL; BROUCKE, 2015).

● Parâmetros

Os parâmetros de um método são definidos entre parênteses, após o nome do método, separados por vírgulas. Um parâmetro consiste em duas partes: seu tipo e o seu nome. Se um método não tem parâmetros, então apenas um par de parênteses vazio é usado. Todos os parâmetros Java são passados por valor, ou seja, sempre que se passa um parâmetro para um método, uma cópia do parâmetro é feita para uso no contexto do corpo do método. Ao se passar uma variável int para um método, o valor daquela variável é copiado. O método pode alterar a cópia, mas não o original. Quando se passa uma referência do objeto como parâmetro para um método, então essa referência é copiada da mesma forma. É preciso lembrar que foi passada para ele. Por exemplo, ao passar uma referência g da classe Gnome para um método que chama este parâmetro de h, então o método pode alterar a referência h de maneira que ela aponte para outro objeto, porém g continuará a referenciar o mesmo objeto, alterando assim o estado do objeto apontado por g (desde que g e h referenciem o mesmo objeto). (GOODRICH; TAMASSIA, 2013, p. 17).

Alguns métodos requerem entradas que fornecem detalhes sobre o trabalho que precisam fazer. Por exemplo, o método println tem um entrada: a string que deve ser impressa. Cientistas da computação utilizam o termo técnico parâmetro para entradas de método. (HORSTMANN, 2009, p.65).

Referências

BAESENS, Bart; BACKIEL, Aimée; BROUCKE, Seppe Vancen. **Beginning Java Programming**: the object-oriented approach. Indianapolis: John Wiley & Sons, 2015.

BURD, Barry. **Java para leigos**. 5. ed. Rio de Janeiro: Alta Books, 2013.

BUYYA, Rajkumar; SELVI, S Thamarai; CHU, Xingchen. **Object-Oriented Programming with Java**: essentials and applications. New Delhi: Tata McGraw Hill, 2009.

GOODRICH, Michael T.; TAMASSIA, Roberto. **Estruturas de Dados & Algoritmos**. 5. ed. Porto Alegre: Bookman, 2013.

HOLMES, Barry; JOYCE, Daniel. **Object-Oriented Programming with Java**. 2. ed. Sudburry: Jones And Bartlett, 2001.

HORSTMANN, Cay. **Conceitos de Computação com Java**. Porto Alegre: Bookman, 2009.

SANTOS, Rafael. **Introdução à programação orientada a objetos usando Java**. Rio de Janeiro: Elsevier, 2003.

SCHILDT, Herbert. **Java para iniciantes**: crie, compile e execute programas Java rapidamente. 6. ed. Porto Alegre: Bookman, 2015.

SERSON, Roberto Rubistein. **Programação orientada a objetos com Java**. Rio de Janeiro: Brasport, 2007.