DIN Departamento de Informática

UEM Universidade Estadual de Maringá

Métodos e Atributos de Classe

Lilian Passos Scatalon
lpscatalon2@uem.br

Programas em Java

São compostos por métodos e classes

- Predefinidos (Biblioteca de Classes Java ou Java API)
- Implementados no programa

Métodos

São as unidades de programa em Java (como as funções em C)

São declarados dentro de uma classe

Por exemplo, a classe abaixo possui um método, que é o método main:

```
public class MinhaClasse {

        public static void main(String[] args) {
                ...
        }

}
```

Métodos

Podem ser chamados:

- por instâncias da classe (objetos)
- diretamente pela classe

```
System.out.println("Raiz quadrada: " + Math.sqrt(49));
```

Métodos estáticos (de classe)

São declarados com a palavra-chave static antes do tipo de retorno

```
public static float calculaMedia(int[] vetor){
        ...
}
```

Métodos estáticos

Para qualquer classe importada pelo programa, é possível chamar os seus métodos estáticos

NomeDaClasse.nomeDoMétodo(argumentos)

O método é invocado sem a necessidade de criar uma instância da classe

Classe Math

Provê uma coleção de métodos para cálculos matemáticos

Por exemplo, a chamada de método estático Math.sqrt(900.0)

é avaliada como 30.0

Classe Math

Para imprimir o valor calculado pela chamada do método, pode-se escrever

System.out.println(Math.sqrt(900.0));

em que o retorno do método sqrt é usado como argumento para o método println

Method	Description
`abs(x)`	absolute value of x
`ceil(x)`	rounds x to the smallest integer not less than x
`cos(x)`	trigonometric cosine of x (x in radians)
`exp(x)`	exponential method e^x
`floor(x)`	rounds x to the largest integer not greater than x
`log(x)`	natural logarithm of x (base e)
`max(x, y)`	larger value of x and y
`min(x, y)`	smaller value of x and y
`pow(x, y)`	x raised to the power y (i.e., x^y)
`sin(x)`	trigonometric sine of x (x in radians)
`sqrt(x)`	square root of x
`tan(x)`	trigonometric tangent of x (x in radians)

Variáveis de classe

Uma classe também pode conter variáveis estáticas, também chamadas de variáveis de classe

São acessadas por:

NomeDaClasse.nomeDaVariávelEstática

Exemplos: as constantes Math.PI e Math.E

Variáveis de classe

São declaradas com o modificador static

Os métodos declarados na classe podem usar as variáveis declaradas na classe

```
public class MinhaClasse {

        static String place = "world";

        public static void main(String[] args) {
                System.out.println("Hello"+place+"!");
        }

}
```

```java
public class EncontraMaximo {

	public static void main(String[] args) {

		Scanner input = new Scanner(System.in);
		System.out.print("Insira três valores de ponto flutuante"+
				" separados por espaços: ");

		double numero1 = input.nextDouble();
		double numero2 = input.nextDouble();
		double numero3 = input.nextDouble();

		double resultado = maximo(numero1, numero2, numero3);

		System.out.println("O valor máximo é: " + resultado);

	}

	public static double maximo(double x, double y, double z) {
		double valorMaximo = x;

		if (y > valorMaximo) {
			valorMaximo = y;
		}

		if (z > valorMaximo) {
			valorMaximo = z;
		}

		return valorMaximo;

	}
}
```

Como o método maximo poderia fazer o reuso do método Math.max?

Chamando métodos

Usando apenas o nome do método quando a chamada está na mesma classe

```
double resultado = maximo(numero1, numero2, numero3);
```

Usando o nome de um objeto com o operador ponto

```
double numero1 = input.nextDouble();
```

Usando o nome da classe com o operador ponto

```
Math.sqrt(49)
```

Retornando de métodos

Quando o tipo de retorno é void

- Termina o corpo do método
- A instrução de retorno é executada

```
return;
```

Quando o método retorna um valor de determinado tipo

- A instrução de retorno contendo uma expressão é executada

```
return expressão;
```

Pilha de chamada de métodos

Também chamada de pilha de execução do programa

É a estrutura que fornece suporte para gerenciar a sequência de chamadas de métodos em um programa

Para cada chamada de método, um *stack frame* (ou registro de ativação) é colocado no topo da pilha

Pilha de chamada de métodos

O *stack frame* (ou registro de ativação) é composto por:

- variáveis locais do método
- endereço de retorno da chamada

Stack overflow é o erro que acontece quando o espaço de memória reservado para a pilha se esgota (tipicamente causado por recursão infinita)

Conversão de tipos dos argumentos

Os argumentos podem ser promovidos ao tipo esperado pelo método no parâmetro correspondente

A instrução (contendo a chamada com argumento inteiro)

```
System.out.println(Math.sqrt(4));
```

imprime 2.0 (valor em ponto flutuante)

Conversão de tipos dos argumentos

Type	Valid promotions
`double`	None
`float`	`double`
`long`	`float` or `double`
`int`	`long`, `float` or `double`
`char`	`int`, `long`, `float` or `double`
`short`	`int`, `long`, `float` or `double` (but not `char`)
`byte`	`short`, `int`, `long`, `float` or `double` (but not `char`)
`boolean`	None (`boolean` values are not considered to be numbers in Java)

Conversão de tipos dos argumentos

Para casos em que há perda de informação devido à conversão, é preciso usar um operador de cast

```
quadrado((int) valorDouble)
```

A chamada converte explicitamente o valor para um inteiro temporário, que é usado como argumento

Pacotes da Java API

Java *Application Programming Interface* (Java API)

Contém muitas classes predefinidas agrupadas em pacotes

São componentes reusáveis nos programas que escrevemos

Visão geral dos pacotes da API Java:

http://docs.oracle.com/javase/8/docs/api/overview-summary.html

Documentação da API Java:

http://docs.oracle.com/javase/8/docs/api

Package	Description
`java.awt.event`	The **Java Abstract Window Toolkit Event Package** contains classes and interfaces that enable event handling for GUI components in both the `java.awt` and `javax.swing` packages. (See Chapter 26, Swing GUI Components: Part 1, and Chapter 35, Swing GUI Components: Part 2.)
`java.awt.geom`	The **Java 2D Shapes Package** contains classes and interfaces for working with Java's advanced two-dimensional graphics capabilities. (See Chapter 27, Graphics and Java 2D.)
`java.io`	The **Java Input/Output Package** contains classes and interfaces that enable programs to input and output data. (See Chapter 15, Files, Input/Output Streams, NIO and XML Serialization.)
`java.lang`	The **Java Language Package** contains classes and interfaces (discussed throughout the book) that are required by many Java programs. This package is imported by the compiler into all programs.
`java.net`	The **Java Networking Package** contains classes and interfaces that enable programs to communicate via computer networks like the Internet. (See online Chapter 28, Networking.)
`java.security`	The **Java Security Package** contains classes and interfaces for enhancing application security.
`java.sql`	The **JDBC Package** contains classes and interfaces for working with databases. (See Chapter 24, Accessing Databases with JDBC.)
`java.util`	The **Java Utilities Package** contains utility classes and interfaces that enable storing and processing of large amounts of data. Many of these classes and interfaces have been updated to support Java SE 8's lambda capabilities. (See Chapter 16, Generic Collections.)

`java.util.concurrent`	The **Java Concurrency Package** contains utility classes and interfaces for implementing programs that can perform multiple tasks in parallel. (See Chapter 23, Concurrency.)
`javax.swing`	The **Java Swing GUI Components Package** contains classes and interfaces for Java's Swing GUI components. This package still uses some elements of the older `java.awt` package. (See Chapter 26, Swing GUI Components: Part 1, and Chapter 35, Swing GUI Components: Part 2.)
`javax.swing.event`	The **Java Swing Event Package** contains classes and interfaces that enable event handling (for example, responding to button clicks) for GUI components in package `javax.swing`. (See Chapter 26, Swing GUI Components: Part 1, and Chapter 35, Swing GUI Components: Part 2.)
`javax.xml.ws`	The **JAX-WS Package** contains classes and interfaces for working with web services in Java. (See online Chapter 32, REST Web Services.)
`javafx` packages	JavaFX is Java's preferred GUI, graphics and multimedia technology for the future. We cover JavaFX extensively throughout the book.
`java.time`	The Java SE 8 **Date/Time API Package** contains classes and interfaces for working with dates and times. (See Chapter 23, Concurrency.)
`java.util.function` and `java.util.stream`	These packages contain classes and interfaces for working with Java SE 8's functional programming capabilities. (See Chapter 17, Lambdas and Streams.)

Bibliografia

Deitel, P.J. e Deitel, H.M. **Java How to Program: Late Objects**. 11th edition. Pearson, 2020.