FAESA CENTRO UNIVERSITÁRIO

Sobrecarga em JAVA

Apresentação

Java é uma linguagem de programação orientada a objetos, o que significa que ela trabalha com objetos que são atributos (dados) e objetos que são métodos (procedimentos sobre os dados).

Em orientação a objetos, sobrecarga de método é quando existem dois métodos com o mesmo nome, mas com assinaturas diferentes, isto é, dois métodos podem ter o mesmo nome se tiverem diferentes números e tipos de parâmetros. Já polimorfismo significa executar de diversas formas uma chamada de método. Sobrecarga e polimorfismo facilitam o reúso de código, reduzindo o trabalho do desenvolvedor.

Nesta Unidade de Aprendizagem, você vai aprender os conceitos de sobrecarga e polimorfismo e terá acesso a exemplos práticos sobre como implementá-los em seus programas.

Bons estudos.

Ao final desta Unidade de Aprendizagem, você deve apresentar os seguintes aprendizados:

- Definir sobrecarga.
- Identificar polimorfismo.
- Construir uma aplicação de sobrecarga.

Desafio

Trabalhar com sobrecarga de métodos em orientação a objetos pode trazer muitas vantagens, como melhor organização do código. Entretanto, é importante ter bom conhecimento sobre o assunto para aplicá-lo da melhor forma em seus projetos.

Você trabalha como analista/programador em uma fábrica de *software* e foi encarregado de atender uma empresa de construção civil. O cliente precisa realizar o cálculo para obter a área exata de cômodos do projeto arquitetônico para elaborar o orçamento. Seus projetos são baseados em três figuras geométricas: retângulo, quadrado e trapézio. Sabe-se que as fórmulas para o cálculo das áreas são as seguintes: quadrado {lado × lado}, retângulo {base × altura} e trapézio {(base maior + base menor) × altura)/2}.

Assim, utilize uma linguagem de programação orientada a objetos para executar a tarefa. Crie uma classe chamada "Area", que contenha os métodos, e outra de controle; na classe de controle, instancie a classe "Area". Chame os métodos das respectivas figuras por meio de uma variável de referência e insira alguns valores para teste. Após, exporte o projeto em um arquivo zip e o envie como anexo.

Infográfico

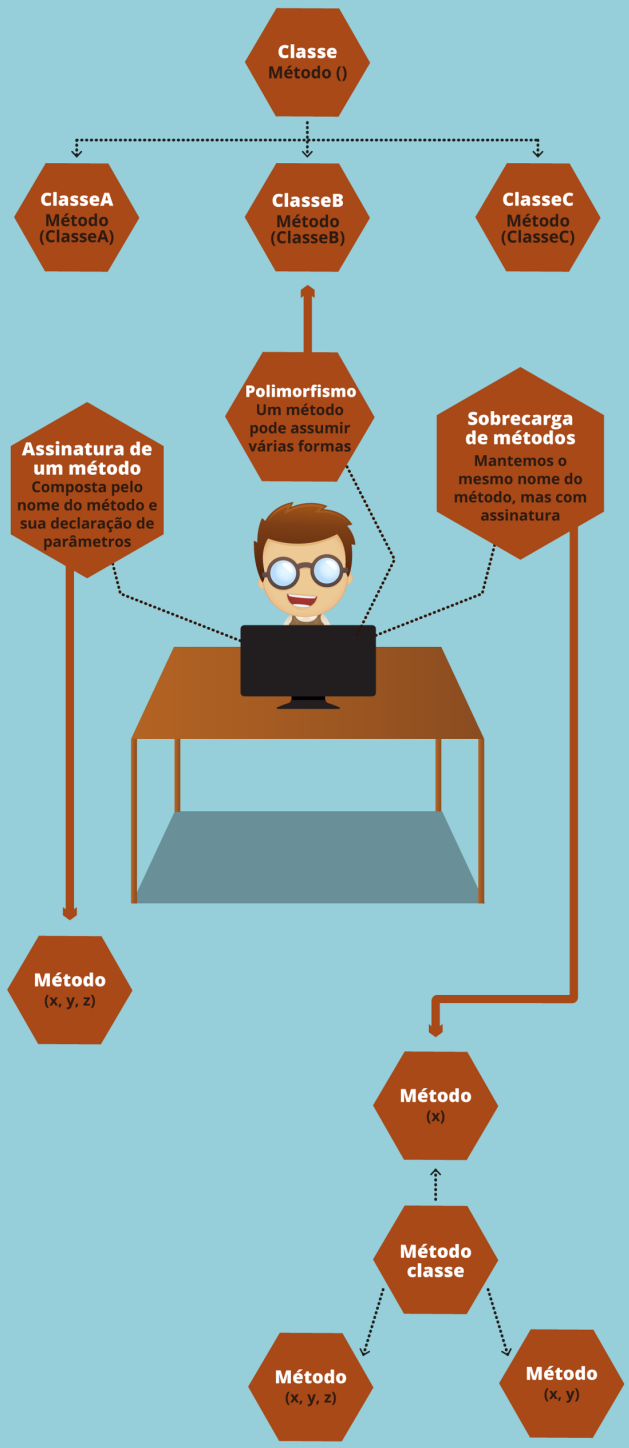
A assinatura do método é um conceito muito relevante no entendimento de sobrecarga e polimorfismo. No entanto, para entender a fundo os conceitos, é interessante usar algum tipo de visualização que faça a correlação entre as expressões.

O Infográfico a seguir apresenta os conceitos de sobrecarga, com os respectivos exemplos.

Exemplificando sobrecarga e polimorfismo

A sobrecarga de métodos permite solucionar diferentes problemas. Ao desenvolver uma aplicação para cadastro de clientes, por exemplo, será necessário criar um método para receber parâmetros iguais, mas que representem o cliente como único: nome, endereço, cidade. Em casos como este, a sobrecarga pode ser utilizada.

Observe o esquema no Infográfico.



Em Java, diferentes métodos são utilizados. Enquanto a sobrescrita cria métodos em classes-filhas, a sobrecarga permite que os métodos existentes utilizem o mesmo nome, mas implementem assinaturas diferentes para dar origem a novos argumentos, representados por meio dos métodos.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Conteúdo do Livro

Em Java, dois ou mais métodos da mesma classe podem compartilhar o mesmo nome, contanto que suas declarações de parâmetros sejam diferentes. Quando é este o caso, diz-se que os métodos são sobrecarregados, e o processo é chamado de sobrecarga de método. A sobrecarga de métodos é uma das maneiras como a linguagem Java implementa o polimorfismo.

No capítulo **Sobrecarga em Java**, base teórica desta Unidade de Aprendizagem, você vai compreender o que é sobrecarga e polimorfismo e aprender a aplicá-los em seus projetos na prática.

Boa leitura.

LINGUAGEM DE PROGRAMAÇÃO

Priscila de Fátima Gonçalves

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS

Sobrecarga

Objetivos de aprendizagem

Ao final deste texto, você deve apresentar os seguintes aprendizados:

- Definir o que é sobrecarga.
- Identificar o polimorfismo.
- Construir uma aplicação de sobrecarga.

Introdução

Em orientação a objetos, a sobrecarga de método ocorre quando temos dois métodos com mesmo nome, mas com assinaturas diferentes, isto é, diferentes números e tipos de parâmetros. Dois métodos podem ter o mesmo nome se tiverem diferentes números e tipos de parâmetros, portanto, diferentes assinaturas.

Neste capítulo, você estudará a sobrecarga e sua aplicação, identificará o polimorfismo e aprenderá como construir as aplicações de sobrecarga.

Conceito

Quando falamos sobre orientação a objetos, vemos que, geralmente, a sobrecarga de método ocorre quando temos dois métodos que possuem o mesmo nome, mas com diferentes assinaturas, ou seja, diferentes tipos e quantidades de parâmetros. Para que você entenda melhor, suponha que dois métodos tenham o mesmo nome, mas possuam diferentes tipos de parâmetros e diferentes números, obtendo, assim, assinaturas diferentes. Em Java, por exemplo, dois ou mais métodos em uma classe podem ter o mesmo nome, desde que os parâmetros declarados sejam diferentes. O polimorfismo é implementado por meio da sobrecarga.

Em C++ você pode fazer uso de sobrecarga em funções e operadores, como apresentado na Figura 1. Dessa forma, fornecerá mais de uma definição para um determinado nome de função dentro do mesmo programa. Em geral, o compilador é quem faz a seleção sobre qual a versão de função é mais apropriada, ou qual é o melhor operador, baseando nos argumentos com os quais ele é chamado. Para esse procedimento, o compilador deve verificar os seguintes pontos:

- objeto que está sendo inicializado;
- o lado esquerdo de uma instrução de atribuição;
- o argumento formal a uma função;
- o argumento formal a um operador definido pelo usuário;
- o tipo de retorno de função.

```
class ponto
{
    int x,y;
public:

    ponto(int a, int b)
    {
       x = a;
       y = b;
    }
    ponto operator+(ponto p);
};

ponto ponto::operator+(ponto p)
{
    int a, b;
    a = x + p.x;
    b = y + p.y;

    return ponto(a, b);
}
```

Figura 1. Sobrecarga de operadores.
Fonte: WikiLivros (2018, documento on-line).

No exemplo da Figura 1, é mostrado um trecho de um código em que x e y não tem referência sobre a qual objeto pertencem. Contudo, a operação ocorre dentro de um dos objetos, o que está antes do operador. Logo, você pode notar que o operador pertence a um dos objetos, obrigatoriamente àquele que o antecede. Dessa forma, neste exemplo foi declarado o segundo dado a operar.

Fique atento

Se for selecionada mais de uma função, a sobrecarga será ambígua e apresentará erro, ou seja, a função que deverá ser selecionada será melhor que todas as outras do grupo para, no mínimo, um argumento. Se não ocorrer dessa maneira, a chamada da função irá gerar um erro.

Identificação de polimorfismo

O polimorfismo é um dos conceitos mais utilizados em programação orientada ao objeto, pois promove a reutilização contínua dos códigos, apresentando a maneira como um método pode assumir formas diferentes das quais foram implementadas inicialmente e agir de modo que possa ser reutilizado, inclusive em outra classe. O polimorfismo cria variações de métodos com nomes totalmente iguais em uma classe, contendo listas de argumentos diferentes para que seja feita a separação deles. Em C++ isso ocorre por meio da conversão de ponteiros ou referências, utilizando objetos em hierarquia de classes.

Segundo Horstmann (2005), polimorfismo descreve um conjunto de objetos de diferentes classes com comportamento similar. A herança é usada para expressar atributos em comum entre as classes, e as funções virtuais permitem variações no comportamento.

Saiba mais

Uma das vantagens da sobrecarga é permitir que os métodos relacionados sejam acessados com o uso de um mesmo nome, assim, quando um método for sobrecarregado, poderá executar qualquer atividade em cada versão existente.

Construção de aplicação de sobrecarga

Para construir uma aplicação que contenha sobrecarga, você, como programador, terá que declarar dois ou mais métodos da mesma classe que compartilhem o mesmo nome, porém trazendo declarações de parâmetros de entradas diferentes, ou seja, declarar versões diferentes do mesmo método. Por meio da sobrecarga de operadores, você poderá redefinir o significado da grande maioria dos operadores em C++, quando ao menos um operando for um objeto da classe.

Uma informação importante é que não se pode criar operadores em C++, apenas podem ocorrer novas definições para operadores já existentes; e nem todos podem ser sobrecarregados.

Veja no Quadro 1 um exemplo de operadores que podem receber a sobrecarga.

Quadro 1. Linguagem de programação C++

Operadores unários	
!	NOT lógico
&	Address-of
( )	Operador cast
*	Desreferência de ponteiro
+	Mais unário
++	Incremento
-	Negação unária
--	Decremento
~	Complemento de um
conversion operators	Operadores de conversão
Operadores binários	
,	Vírgula
!=	Desigualdade
%	Módulo
%=	Atribuição de módulo
&	AND bit a bit
&&	AND lógico
&=	Atribuição AND de bit a bit
*	Multiplicação
*=	Atribuição de multiplicação

(Continua)

(Continuação)

Quadro 1. Linguagem de programação C++

Operadores binários	
+	Adição
+=	Atribuição de adição
-	Subtração
-=	Atribuição de subtração
->	Seleção de membro
->*	Seleção de ponteiro para membro
/	Divisão
/=	Atribuição de divisão
<	Menor que
<<	Shift esquerda
<<=	Atrib. de deslocamento para a esquerda
<=	Menor ou igual
=	Atribuição
==	Igualdade
>	Maior que
>=	Maior ou igual
>>	Shift direita
>>=	Atrib. de deslocamento para a direita
^	OR exclusivo
^=	Atribuição de OR exclusivo
Operadores unários ou binários	
( )	Chamada de função
[ ]	Subscrito de matriz
delete	Delete
new	New

Exemplo

Para mostrar o conceito de sobrecarga, veja o código a seguir, em que um salário deverá ser calculado de duas formas: uma somente passando o número de horas trabalhadas no mês; e outra com o salário básico mais o número de horas extras na classe Instrutor (Java ou C#):

```
...
public float calculaSalario (int horas){
    this.salario = 20*horas;
}
public float calculaSalario (float básico, int horas){
    this.salario = básico + 20*horas;
}
.....
```

Observe como a sobrecarga pode ser utilizada:

```
// function_overloading.cpp
// compile with: /EHsc
#include <iostream>
#include <math.h>

// Prototype three print functions.
int print( char *s );                   // Print a string.
int print( double dvalue );             // Print a double.
int print( double dvalue, int prec );  // Print a double with a
//  given precision.
using namespace std;
int main( int argc, char *argv[] )
{
const double d = 893094.2987;
if( argc < 2 )
    {

// These calls to print invoke print( char *s ).
print( "This program requires one argument." );
print( "The argument specifies the number of" );
print( "digits precision for the second number" );
print( "printed." );
exit(0);
    }

// Invoke print( double dvalue ).
print( d );

// Invoke print( double dvalue, int prec ).
print( d, atoi( argv[1] ) );
}

// Print a string.
int print( char *s )
{
cout << s << endl;
```

```
return cout.good();
}

// Print a double in default precision.
int print( double dvalue )
{
cout << dvalue << endl;
return cout.good();
}

// Print a double in specified precision.
//  Positive numbers for precision indicate how many digits
//  precision after the decimal point to show. Negative
//  numbers for precision indicate where to round the number
//  to the left of the decimal point.
int print( double dvalue, int prec )
{
// Use table-lookup for rounding/truncation.
static const double rgPow10[] = {
10E-7, 10E-6, 10E-5, 10E-4, 10E-3, 10E-2, 10E-1, 10E0,
10E1,  10E2,  10E3,  10E4, 10E5,  10E6
    };
const int iPowZero = 6;
// If precision out of range, just print the number.
if( prec < -6 || prec > 7 )
return print( dvalue );
// Scale, truncate, then rescale.
dvalue = floor( dvalue / rgPow10[iPowZero - prec] ) *
rgPow10[iPowZero - prec];
cout << dvalue << endl;
return cout.good();
}
```

Referências

HORSTMANN, C. *Conceitos de computação com o essencial de C++*. 3. ed. Porto Alegre: Bookman, 2005.

WIKILIVROS. *Programar em C++/Sobrecarga de operadores*. 2018. Disponível em: <https://pt.wikibooks.org/wiki/Programar_em_C%2B%2B/Sobrecarga_de_operadores>. Acesso em: 1 jul. 2018.

Leituras recomendadas

ARNOLD, K.; GOSLING, J.; HOLMES, D. *A linguagem de programação Java*. 4. ed. Porto Alegre: Bookman, 2007.

MEYERS, S. *C++ eficaz*: 55 maneiras de aprimorar seus programas e projetos. 3. ed. Porto Alegre: Bookman, 2011.

MICROSOFT. *Sobrecarga de função*. c2018. Disponível em: <https://msdn.microsoft.com/pt-br/library/5dhe1hce.aspx>. Acesso em: 4 maio 2018.

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

WIKILIVROS. *Programar em C++*. 2011. Disponível em: <https://pt.wikibooks.org/wiki/Categoria:Livro/Programar_em_C%2B%2B>. Acesso em: 4 maio 2018.

Encerra aqui o trecho do livro disponibilizado para esta Unidade de Aprendizagem. Na Biblioteca Virtual da Instituição, você encontra a obra na íntegra.

Conteúdo:

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS

Dica do Professor

Cada linguagem de programação tem sua própria sintaxe e pode reaproveitar alguns métodos durante o desenvolvimento de aplicações. Por exemplo, o Java permite que seja feita a sobrecarga do método construtor. Ele precisa ter o mesmo nome da classe e não pode ter indicação do tipo de retorno. Pode-se invocar o construtor ao se criar um objeto usando o operador *new*. O retorno do operador *new* será uma referência para o objeto recém-criado.

Nesta Dica do Professor, veja um exemplo prático de como implementar a sobrecarga em construtores.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Exercícios

1) **Um dos principais paradigmas de programação determina que os programadores decidam o melhor caminho e sejam capazes de solucionar problemas de alta complexidade. Para isso, diferentes métodos, funções e recursos são aplicados ao código. Sobrecarga e polimorfismo são conceitos úteis na programação orientada a objetos.**

Quando ocorre a sobrecarga de métodos?

A) Quando um método é declarado com *void*.

B) Quando não se declara *void*, mas um tipo de dado nos métodos.

C) Quando o método é declarado com a palavra *abstract*.

D) Quando um ou mais métodos têm a mesma assinatura, mas implementações específicas de acordo com a classe a que pertencem.

E) Quando um ou mais métodos têm o mesmo nome, mas conjuntos de parâmetros diferentes.

2) **Na programação orientada a objetos, assinatura é a identificação única de um método. É comum, ao trabalhar com métodos, utilizar a expressão "assinatura do método".**

Marque a alternativa que melhor explica do que se trata essa expressão.

A) Assinatura do método é quando a palavra-chave *public* é colocada no início da declaração do método.

B) Assinatura do método é quando se declara variáveis e tipos no argumento do método.

C) Assinatura do método consiste em toda a declaração do método, menos o seu corpo.

D) A assinatura do método é composta por seu nome e parâmetros.

E) Assinatura do método refere-se ao conjunto de parâmetros desse método.

3) **No contexto de programação orientada a objetos, um programador pode definir o que está sendo implementado, ao analisar o código, identificando de que forma os objetos trocam**

mensagens entre si; o mesmo ocorre com a sobreposição e a sobrecarga. Para saber se um programa em Java apresenta sobrecarga, é preciso analisar o código-fonte.

Analise os códigos a seguir e informe em qual deles existe sobrecarga de métodos.

A) public abstract class Area { public abstract void calcular(); }

B) public class Pessoa { private String nome; public String getNome() { return nome; } public void setNome(String nome) { this.nome = nome; } }

C) public class Colaborador { String nome; String endereco; public Colaborador(String nome, String endereco){ this.nome = nome; this.endereco = endereco; } }

D) public class Colaborador { String nome; String endereco; public Colaborador(){ } public Colaborador(String nome){ this.nome = nome; } public Colaborador(String nome, String endereco){ this.nome = nome; this.endereco = endereco; } }

E) public class Colaborador { private String nome; private String endereco; public Colaborador(){ } public String getNome() { return nome; } public void setNome(String nome) { this.nome = nome; } public String getEndereco() { return endereco; } public void setEndereco(String endereco) { this.endereco = endereco; } }

4) Sobrecarga de método (sobrecarga) é um conceito de polimorfismo, que basicamente envolve a criação de variantes do mesmo método, ou seja, a criação de dois ou mais métodos com o mesmo nome em uma classe. Por esse motivo, é possível tratar a sobrecarga de métodos como um tipo de polimorfismo.

Marque a alternativa que melhor explica o polimorfismo.

A) Trata-se de uma técnica na qual subclasses herdam membros da superclasse.

B) É quando uma classe implementa uma interface.

C) É quando o mesmo método é utilizado de maneiras diferentes de acordo com a necessidade de cada classe.

D) É quando se utiliza o método com o mesmo nome, mas alterando seu conjunto de parâmetros, sejam números, ordem dos parâmetros ou tipos diferentes.

E) Polimorfismo é quando uma classe tem métodos *set* e *get*.

5) Sobrecarga e polimorfismo são muito úteis em Java para facilitar o reúso de código já criado.

Em relação à sobrecarga e ao polimorfismo, é correto afirmar que:

A) quando os métodos de uma mesma classe têm a mesma assinatura, tem-se a ocorrência de sobrecarga.

B) há sobrecarga de métodos quando, em classes diferentes, há métodos com mesmo nome e conjunto de parâmetros diferentes.

C) na sobrecarga, o tipo e/ou a quantidade dos parâmetros de cada método sobrecarregado devem diferir.

D) quando uma interface é implementada e seus métodos são usados de forma específica em cada classe, caracteriza-se a sobrecarga.

E) na sobrecarga, os métodos têm implementações sempre idênticas.

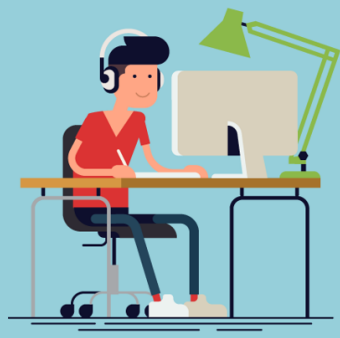
Na prática

A sobrecarga de métodos pode ser utilizada em diferentes situações; ela consiste na reutilização do nome em mais de um método. No entanto, para haver sobrecarga, a assinatura do método deve mudar, ou seja, precisa haver diferença nos parâmetros.

Veja, Na Prática, como usar a sobrecarga em dois métodos de soma, em que um método recebe dois valores como *input* e outro recebe três valores.

Aplicando sobrecarga em Java para operações matemáticas

A sobrecarga de método pode ser utilizada para somar em uma classe. Acompanhe o caso a seguir.



João precisa criar uma função no sistema de estoque que permita a soma de dois números inteiros. O método será implementado para atualizar a quantidade de itens, sempre que houver novas entradas, considerando a quantidade atual, declarada no campo que contabiliza os produtos e a quantidade entrante.

Para isso, ele declara o método da seguinte forma:

```
public void somar(int n1, int n2){
int soma = n1 + n2;
}
```

O método **somar** recebe, por parâmetro, dois números inteiros e executa a soma de seus valores atribuindo o resultado à variável soma.

Agora, imagine que, no mesmo sistema, ocorram devoluções de produtos e que, ao final do dia, elas devem ser contabilizadas no sistema, pois retornam ao estoque.
O sistema, então, precisa somar um terceiro campo, “devolvidos”, junto à quantidade de produtos atual e aos novos produtos. Para isso, João precisa somar três números, adaptando o método criado.

```
public void somar(int n1, int n2, int n3){
int soma = n1 + n2 +n3;
}
```

Perceba que o nome continua o mesmo, porém a assinatura do método muda com o acréscimo de mais uma variável e tipo.

A sobrecarga de métodos (*overload*) é um conceito vinculado ao polimorfismo que permite criar variações de um mesmo método, agregando dois ou mais métodos com o mesmo nome em uma classe.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Saiba mais

Para ampliar o seu conhecimento a respeito desse assunto, veja abaixo as sugestões do professor:

Sobrecarga em Java

O conteúdo a seguir aborda a aplicação de sobrecarga em Java, apresentando sua utilização em conjunto com o polimorfismo.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Polimorfismo em Java

Neste vídeo, aprofunde seu conhecimento prático sobre como funciona o polimorfismo em Java.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Padrões e projetos orientados a objetos

Nesta obra, estude a sobrecarga de operadores.

Conteúdo interativo disponível na plataforma de ensino!