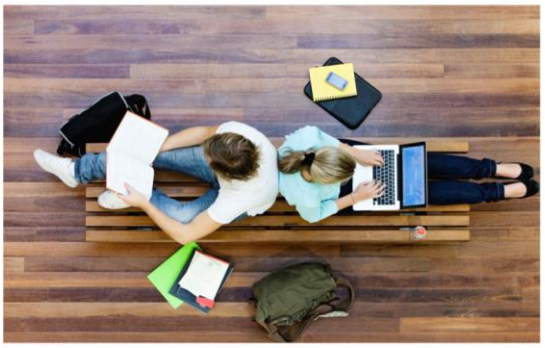

ORACLE
Academy

Java Fundamentals

7-2

Parâmetros e métodos sobrecarregados



ORACLE
Academy

Objetivos

- Esta aula abrange os seguintes tópicos:
  - Usar modificadores de acesso
  - Passar objetos para métodos
  - Retornar objetos de métodos
  - Usar métodos de argumento variáveis
  - Sobrecarregar construtores
  - Sobrecarregar métodos
  - Escrever uma classe com arrays, construtores e métodos específicos

Modificadores de acesso

- Os modificadores de acesso especificam a acessibilidade a variáveis, métodos e classes que se modificam
- Há quatro modificadores de acesso em Java:

Modificador de Acesso	Descrição
public (público)	Permite o acesso de qualquer lugar
protected (protegido)	Permite o acesso somente de dentro da mesma classe, de uma subclasse ou de outras classes do mesmo pacote que o modificador
private (privado)	Permite o acesso somente de dentro da mesma classe do modificador
"default" (“padrão” não especificado/em branco)	Permite o acesso somente de dentro da mesma classe ou de outras classes do mesmo pacote que o modificador

Modificador de Acesso Público

- Os modificadores de acesso público permitem o acesso de qualquer lugar
- Em Java, adicionar a palavra-chave público da forma como a variável, o método ou a classe é declarada torna a variável, o método ou a classe acessível de qualquer lugar

Classes e métodos geralmente são públicos.

Declarando como Público

- O código abaixo mostra como declarar uma variável, um método ou uma classe como pública
  - Variável:

```
public int milesRan = 2;//public access
int timePassed = 17;//access not specified
```

  - Método:

```
public int addMiles(int a, int b){
   return a+b;
}//fim do método addMiles
```

  - Classe:

```
public class Jogging{
   //class code here
}//fim da classe Jogging
```


Variáveis de local nunca têm um modificador de acesso. Somente variáveis de instância usam modificadores de acesso.

Modificadores de Acesso Protegidos e Padrão

- Um modificador de acesso protegido permite o acesso de dentro da classe, subclasse ou outras classes do mesmo pacote do modificador
- Para declarar uma variável, um método ou uma classe como protegida, escreva a palavra-chave protegido em vez de público
- Um modificador de acesso padrão permite o acesso de dentro do mesmo pacote
- Para declarar uma variável, um método ou uma classe como padrão, você não deve incluir um modificador de acesso

Um erro comum que os programadores iniciantes em Java cometem é não especificar um modificador de acesso e, dessa forma, eles obtêm o padrão.

Modificador de Acesso Privado

- Um modificador de acesso privado:
  - Permite o acesso somente de dentro da mesma classe
  - É o modificador de acesso mais restritivo
  - É o oposto do modificador de acesso público

```
private int bankAccountNumber;
```

Em geral, os campos são privados.

Quando Usar Público ou Privado

Tipo	Definição	Quando Usar
`public` (público)	Permite o acesso de qualquer lugar	Quando não importa que todos possam acessar seu código ou quando você deseja compartilhar seu código com os outros
`private` (privado)	Permite acesso somente de dentro da mesma classe	Quando é importante que seu código esteja seguro e não possa ser acessado de qualquer lugar, exceto de dentro da própria classe

Objetos como Parâmetros

- Um parâmetro é uma variável em uma declaração de método que é repassada ao método

```
public int method(int parameter1, int parameter2)
```

- Tipos de parâmetro são os tipos de parâmetros que podem ser repassados a um método
- Isso inclui:
  - Tipos primitivos (como int, double, char)
  - Objetos
    - String
    - Array

```
public int method(int anInt, double aDouble,
            String aString, MyClassName anObjectOfMyClass)
```


Não faz diferença se o objeto é originário de uma classe de API ou se é de uma classe criada pelo usuário.

Exemplo de Objetos como Parâmetros

- Um empregador tem uma vaga para a promoção de um de seus funcionários
- Ele deseja criar um método que aceite um funcionário como um parâmetro e calcule e retorne a classificação de funcionários com base em suas qualificações para o novo cargo

```
public int promotion(Employee E){
   int timeEmployed = E.getLengthOfEmployment();
   //do some calculations to set a rating for E
   return rating;
}//fim do método promotion
```

Repassando Objetos como Parâmetros

- Repassar objetos como parâmetros permite um acesso bem mais fácil às informações contidas no objeto
- Permite também fazer alterações nos objetos dentro do método e até comparar dois objetos que não podem usar métodos de comparação primitivos

Retornando Objetos

- Escrever um método que retorne um objeto é muito semelhante a escrever um método que retorne um tipo primitivo
- Por exemplo, o empregador do exemplo anterior acaba de aprender que os métodos podem retornar um objeto
- Para facilitar a busca de um funcionário a ser promovido, ele pode escrever um método que aceite dois funcionários
- O método retorna o que tiver a melhor classificação
- Isso é mais fácil do que passar por cada funcionário, obter cada uma de suas classificações e compará-los

Exemplo de Retorno de Objetos

- O funcionário identifica o que está sendo retornado

Para retornar um objeto, basta escrever o tipo do objeto aqui.

```
public Employee promotion(Employee A, Employee B){
   //calculate to compare which employee is better

   //if employee A is better
   return A;
   //if employee B is better
   return B;
}//fim do método promotion
```

Métodos de Argumento Variável

- Um método de argumento variável:
  - É um método escrito para tratar de um número variável de argumentos
  - Funciona somente se você chamar o método com o mesmo tipo de argumento que o método exige
- Um método de argumento variável parece com isso:

```
public int total(int ... nums){
   int sum = 0;
   for(int i = 0; i < nums.length; i++)
      sum += nums[i];
   return sum;
}//fim do método total
```

Exemplo de Métodos de Argumento Variável

- Por exemplo, um método inicializado com um argumento variável de inteiros não pode ser chamado com qualquer número de Strings, e sim com qualquer número de inteiros do argumento
- Se outro método for declarado com um argumento variável de Strings, elas devem chamar esse método com String(s) para atender os argumentos

Por Que os Arrays Não São Usados em Métodos de Argumento Variável?

- Por que não simplesmente usar um array?
  - Em um programa, você deve saber o número de elementos em um array para criar um
  - Se o número de elementos mudar, você precisará de um array diferente para cada tamanho diferente
  - O uso de um método de argumento variável permite o uso do método sem nunca precisar inicializar um array
  - Permite também vários usos com um número variável de elementos

Métodos de Argumento Variável e Inteiros

- Um método de argumento variável funciona somente com números inteiros?
  - Não, o argumento variável funciona com qualquer tipo primitivo, objeto e até arrays
  - Você pode ter um argumento variável de arrays

Exemplo de Funcionário

- Para determinar as promoções de funcionários, o empregador estava codificando um método que comparava dois funcionários e retornava o melhor deles
  - Agora que o empregador tem o método para comparar os funcionários, ele precisa de uma forma de comparar todos os funcionários de uma vez, em vez de comparar somente dois de cada vez
  - É aí que os argumentos variáveis podem ajudar

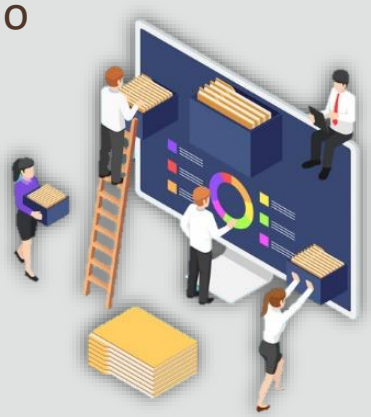
Código de Exemplo do Funcionário do Argumento Variável

- Código para comparar todos os funcionários:

```
public Employee promotion(Employee ... employees){
   Employee bestCandidate = employees[0];
   /*go through the list of employees and calculate
     which one is the best candidate*/
   for(int i = 1; i < employees.length; i++){
     //if there is a candidate better than the current best
     if(employees[i].getRating > bestCandidate.getRating){
        //update the bestCandidate to the better one
        bestCandidate = employees[i];
     }//fim if
   }//fim for
   //return the best candidate found for the promotion
   return bestCandidate;
}//fim do método promotion
```

Chamando um Método com Argumentos Variáveis

- Chamar um método com argumentos variáveis é como chamar qualquer outro método
- No entanto, ele pode ser chamado com um número diferente de argumentos cada vez que ele é chamado



Chamando um Método com Argumentos Variáveis

- O código abaixo demonstra essa ideia
- Sam, Erica, Dominic, Sandy e Jake são funcionários
- O empregador está querendo promover Sam, Erica ou Dominic a gerente, e Sandy ou Jake a auxiliar de gerente

```
/*This compares Sam, Erica, and Dominic and assigns
  the best candidate of the 3 to newManager.*/
Employee newManager = promotion(sam, erica, dominic);

/*This compares Sandy and Jake and assigns the better
  of the 2 to newAssistantManager*/
Employee newAssistantManager = promotion(sandy, jake);
```

Sobrecarregando Construtores

- Os construtores atribuem valores iniciais a variáveis da instância de uma classe
- Os construtores dentro de uma classe são declarados como métodos
- Sobrecarregar um construtor significa ter mais de um construtor com o mesmo nome, porém tipos e/ou números diferentes de argumentos

Exemplo 1 de Sobrecarga de Construtores

- Este exemplo sobrecarrega o construtor público de uma classe Dog

```
public class Dog{
   public Dog(){
      ...implementation...
   }//fim construtor
   public Dog(int weight){
      ...implementation...
   }//fim construtor
   public Dog(String barkNoise){
      ...implementation...
   }//fim construtor
   public Dog(int weight, int loudness, String barkNoise){
      ...implementation...
   }//fim construtor
}//fim da classe Dog
```

Como Funciona a Sobrecarga de Construtores

- A sobrecarga de construtores funciona da seguinte forma:
  - O Java lê o construtor com base nos argumentos que são repassados a ele
  - Após identificar o nome do construtor, ele compara os tipos de argumento
  - Se os tipos de argumento não coincidirem com o primeiro construtor com esse nome, ele continuará no segundo, terceiro, e assim por diante, até identificar uma correspondência do nome do construtor e do tipo de argumento
  - Se ele não encontrar uma correspondência, o programa não fará a compilação

Exemplo 2 de Sobrecarga de Construtores

```
public class Dog{
   private int weight;
   private int age;
   private String barkNoise;

   public Dog(){
     weight = 12;
     loudness = 4;
     barkNoise = "Woof";
   }//fim construtor

   public Dog(int w, int l){
     weight = w;
     loudness = l;
     barkNoise = "ARF!";
   }//fim construtor

   public Dog(int w, int l, String bark){
     weight = w;
     loudness = l;
     barkNoise = bark;
   }//fim construtor
}//fim da classe Dog
```

Este é um construtor que especifica o peso do cachorro e a altura do latido nos argumentos

Este é um construtor que especifica o peso do cachorro, a altura do latido e o som do latido nos argumentos.

Explicação do Exemplo 2 de Sobrecarga de Construtores

- Dog() é o construtor padrão
- Um construtor padrão não tem argumentos
- Se você inicializou um objeto Dog usando este construtor, ele teria um peso de 12, uma altura de latido de 4 e um som de latido “woof”
- Os últimos dois construtores da classe Dog permitem a atribuição de variáveis de instâncias para diferenciar de acordo com as especificações durante a inicialização

Explicação do Exemplo 2 de Sobrecarga de Construtores

- Embora o construtor padrão Dog tenha código para inicializar as variáveis da classe, isso é opcional
- Se o construtor padrão não tiver código, as variáveis da classe serão inicializadas com:
  - null para objetos
  - 0 (zero) para tipos numéricos primitivos
  - false para booleano

Explicação do Exemplo 2 de Sobrecarga de Construtores

- Se um construtor não for escrito para uma classe, o construtor padrão (sem código) será fornecido pela JVM
- Se não houver um construtor padrão escrito e houver um ou mais de outros construtores, a JVM não fornecerá um construtor padrão

Esteja atento ao segundo marcador porque ele geralmente é esquecido

Sobrecarregando Métodos

- Assim como sobrecarregar construtores, a sobrecarga de um método ocorre quando o tipo e/ou número de parâmetros são diferentes
  - Veja abaixo um exemplo de uma situação na qual um método precisa ser sobrecarregado
    - Crie a classe Dog e crie uma instância de Dog em uma classe Driver
    - Chame (use) ambos os métodos bark()

```
public class Dog{
   private int weight;
   private int loudness;
   private String barkNoise;

   public void bark(String b){
         System.out.println(b);
   }//fim do método bark
   public void bark(){
         System.out.println("Woof");
   }//fim do método bark
}//fim da classe Dog
```

ORACLE Academy

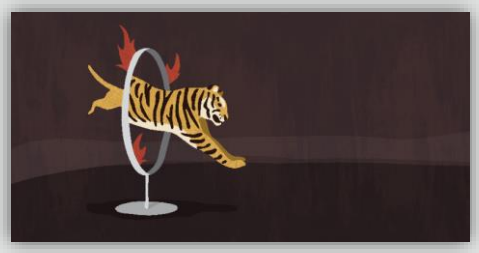
Um erro comum é tentar sobrecarregar um método alterando apenas o tipo de retorno. Sobrecarregar métodos envolve apenas a alteração dos tipos e do número de parâmetros.

Terminologia

- Os principais termos usados nesta aula foram:
  - Modificador de acesso
  - Construtor
  - Construtor padrão
  - Sobrecarga
  - Modificador de acesso privado
  - Modificador de acesso público
  - Método de argumento variável

Resumo

- Nesta aula, você deverá ter aprendido a:
  - Usar modificadores de acesso
  - Passar objetos para métodos
  - Retornar objetos de métodos
  - Usar métodos de argumento variáveis
  - Sobrecarregar construtores
  - Sobrecarregar métodos
  - Escrever uma classe com arrays, construtores e métodos específicos




ORACLE
Academy