

The logo for Oracle Academy. The word "ORACLE" is in a bold, orange, sans-serif font. Below it, the word "Academy" is in a smaller, dark gray, sans-serif font. The entire logo is centered on a light gray background, which is framed by dark gray horizontal bars at the top and bottom.

ORACLE

Academy

Java Fundamentals

7-1

Classes, Objetos e Métodos

ORACLE
Academy



Copyright © 2022, Oracle e/ou suas empresas afiliadas. Oracle, Java e MySQL são marcas comerciais registradas da Oracle Corporation e/ou de suas empresas afiliadas. Outros nomes podem ser marcas comerciais de seus respectivos proprietários.

Objetivos

- Esta aula abrange os seguintes tópicos:
 - Reconhecer o formato geral correto de uma classe
 - Criar um objeto de uma classe
 - Criar métodos que sejam compilados sem erros
 - Retornar um valor de um método
 - Usar parâmetros em um método
 - Criar uma classe de driver e adicionar instâncias de classes de Objeto
 - Adicionar um construtor a uma classe
 - Aplicar o novo operador
 - Descrever coleta de lixo e finalizadores
 - Aplicar a referência this
 - Adicionar um construtor para inicializar um valor



Criando um Modelo de Classe

- Os programadores podem criar suas próprias classes
- As classes são essencialmente um modelo ou um esquema de todas as instâncias da classe
- O código de classe também comunica ao compilador como definir, criar e interagir com objetos da classe
- O código no slide a seguir começa a criar a classe Vehicle que representará a descrição básica dos objetos Vehicle

Criando um Exemplo de Modelo de Classe

```
public class Vehicle {  
    //The Vehicle class has two fields  
    private String make;  
    private int milesPerGallon;  
    //construtor  
    public Vehicle() {  
    } //end construtor  
    //mutator/setter methods  
    public void setMake(String m) {  
        make = m;  
    } //fim do método setMake  
    public void setMilesPerGallon(int mpg) {  
        milesPerGallon = mpg;  
    } //end method setMilesPerGallon  
    public String getMake() {  
        return make;  
    } //fim do método getMake  
    public int getMilesPerGallon() {  
        return milesPerGallon;  
    } //fim do método getMilesPerGallon  
} //fim da classe Vehicle
```

Criando uma Instância de uma Classe

- Após criar uma classe, você pode criar instâncias da classe (objetos) em uma classe Driver ou dentro de outras classes de objetos
- Instâncias:
 - Herdam todos os atributos e métodos definidos no modelo de classes
 - Interagem independentemente umas das outras
 - São objetos de referência
 - São criadas usando o operador new

Como ocorre com as Strings, as referências armazenam o endereço do objeto. (Naturalmente, as Strings são objetos.)

Instanciar uma Instância

- Para instanciar uma instância de um Veículo chamado myCar, escreva:

```
public class VehicleTester{  
    public static void main(String[] args){  
        Vehicle myCar = new Vehicle();  
    } //fim do método main  
} //fim da classe VehicleTester
```

Em Java, a instanciação é a criação de objetos de uma classe.



Construtores

- Construtores são métodos que permitem ao usuário criar instâncias de (instanciar) uma classe
- A boa prática de programação indica que as classes devem ter um construtor padrão
- Os construtores que contêm parâmetros normalmente inicializam as variáveis privadas da classe para valores passados pelo usuário
- Os construtores não têm um tipo de retorno (void ou outro)



Os construtores não têm um tipo de retorno porque eles retornam um novo objeto dessa classe.

Construtor Padrão

- A boa prática de programação indica que as classes devem ter um construtor padrão
- Um construtor padrão:
 - Não aceita parâmetros
 - Normalmente inicializa todas as variáveis privadas para valores de base

```
public Vehicle() {  
    make = "";  
    milesPerGallon = 0;  
} //fim construtor
```

Construtor com Parâmetros

- Um construtor com parâmetros é usado quando você deseja inicializar as variáveis privadas para valores diferentes dos valores padrão

```
public Vehicle(String m, int mpg){  
    make=m;  
    milesPerGallon=mpg;  
} //fim construtor
```

Parâmetros

Os parâmetros são variáveis listadas como parte de uma declaração de um método (ou construtor). No exemplo acima, String m e int mpg são parâmetros. Os valores são fornecidos aos parâmetros quando é feita uma chamada ao método ou construtor.



Instanciar a Instância Vehicle

- Para instanciar uma instância Vehicle usando o construtor com parâmetros, escreva: argumentos

```
Vehicle myCar = new Vehicle("Toyota", 30);
```

- Para instanciar uma instância Vehicle usando o construtor padrão, escreva:

```
Vehicle myCar = new Vehicle();
```



Definindo Métodos

- Um método é um bloco de código chamado pelo nome e pode ser chamado a qualquer momento em um programa utilizando simplesmente o nome o método
- Existem quatro partes principais para definir seu próprio método:
 - Modificador de Acesso (público, privado, protegido, padrão)
 - Tipo de retorno
 - Nome do método
 - Parâmetro(s)

```
public returnType methodName(Parameter p, ...)  
{  
    /*code that will execute with each call to the method goes here*/  
}
```

O modificador de acesso padrão é indicado com a ausência de público, privado ou protegido. Não use a palavra padrão.

Os métodos (e as classes), geralmente, terão o modificador de acesso público. Campos não constantes, geralmente, têm o modificador de acesso privado.

Componentes de um Método

- Os componentes do método incluem:
 - Tipo de retorno:
 - identifica o tipo de objeto, se houver, que será retornado quando o método for chamado
 - Se nada for retornado, o tipo de retorno será declarado como void (nulo)
 - Nome do método:
 - usado para fazer uma chamada ao método

Componentes de um Método

- Parâmetro(s):
 - o programador pode optar por incluir parâmetros, dependendo da finalidade e função do método
 - Os parâmetros podem ser de qualquer tipo de objeto, mas o tipo e o parâmetro usados ao chamar o método devem coincidir com o tipo de parâmetro especificado na definição do método

Exemplo de Componentes do Método

Tipo de retorno

Nome do método

Parâmetros

```
public String getName(String firstName, String lastName)
{
    return( firstName + " " + lastName );
} //fim do método getName
```

Métodos de Classe

- Toda classe terá um conjunto de métodos associado a ela, o que permite a funcionalidade da classe
- Método de acesso
 - “getter”
 - Retorna o valor de uma variável privada específica
- Método modificador
 - “setter”
 - Altera ou define o valor de uma variável privada específica
- Método funcional
 - Retorna ou realiza algum tipo de funcionalidade da classe

Métodos de Acesso

- Os métodos de acesso acessam e retornam o valor de uma variável privada específica da classe
- O tipo de retorno não void (não nulo) corresponde ao tipo de dados da variável que você está acessando
- Incluem uma instrução de retorno
- Normalmente não têm parâmetros

```
public String getMake() {  
    return make;  
} //fim do método getMake  
  
public int getMilesPerGallon() {  
    return milesPerGallon;  
} //fim do método getMilesPerGallon
```



Métodos Modificadores

- Os métodos modificadores definem ou modificam o valor de uma variável privada específica da classe
- Tipo de retorno void (nulo)
- Parâmetro com um tipo que corresponde ao tipo da variável que está sendo definida

```
public void setMake(String m) {\n    make = m;\n} //fim do método setMake\n\npublic void setMilesPerGallon(int mpg) {\n    milesPerGallon = mpg;\n} //fim do método setMilesPerGallon
```



Os métodos modificadores também podem ter código adicional para testar ou modificar os parâmetros. Por exemplo, pode haver outro método `setMilesPerGallon` que tenha um parâmetro duplo. O corpo do método pode adicionar 0.5 e, em seguida, converter o valor em um inteiro e atribuí-lo ao campo.

Métodos Funcionais

- Os métodos funcionais executam uma funcionalidade para a classe
- Tipo de retorno nulo e não nulo
- Os parâmetros são opcionais e usados de acordo com o que é necessário para a função do método



Métodos Funcionais

- Veja abaixo um método funcional da classe Vehicle que compara dois veículos e retorna um valor int para a comparação

```
/*Compares the miles per gallon of each vehicle passed in,
 returns 0 if they are the same, 1 if the first vehicle is
 larger than the second and -1 if the second vehicle is
 larger than the first*/

public int compareTo(Vehicle v1, Vehicle v2){
    if(v1.getMilesPerGallon() == v2.getMilesPerGallon())
        return 0;
    if(v1.getMilesPerGallon() > v2.getMilesPerGallon())
        return 1;
    return -1;
} //fim do método compareTo
```

Usando Construtores e Métodos em um exemplo de método principal da classe Driver

- No seguinte:
 - Qual funcionalidade cada linha tem?
 - O que a instrução final de impressão exibirá na tela?

```
public class VehicleTester{  
    public static void main(String[] args){  
        Vehicle v;  
        v=new Vehicle() ;  
        v.setMake("Ford") ;  
        v.setMilesPerGallon(35) ;  
  
        System.out.print("My "+v.getMake()  
                        + " gets " + v.getMilesPerGallon()  
                        + " mpg.");  
    }//fim do método main  
}//fim da classe VehicleTester
```

Referência this

- Dentro de um método ou construtor de uma instância, `this` é uma referência ao objeto atual
- A referência ao objeto cujo método ou construtor está sendo chamado
- Consulte qualquer membro do objeto atual usando `this`
- Mais comumente usada quando um campo é seguido por um parâmetro de método ou construtor com o mesmo nome

Exemplo de Referência this

- Quando um argumento de método “segue” um campo do objeto, a referência this é usada para diferenciar o escopo local do escopo da classe

```
public class Point {  
    private int x;  
    Private int y;  
  
    //construtor  
    public Point(int x, int y) {  
        this.x = x;  
        this.y = y;  
    }//fim construtor  
}//fim da classe Point
```

Exemplo da Classe Card

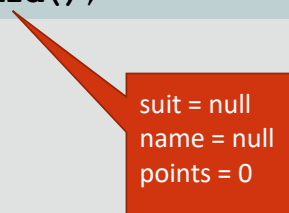
- Considere um baralho de cartas padrão
- Para representar cada carta como uma instância de uma classe Card, que atributos a classe deveria ter?
 - Suit
 - Name
 - Points

```
public class Card {  
    private String suit;  
    private String name;  
    private int points;  
} //fim da classe Card
```


Representação do Objeto de Referência

- Ao criar uma nova instância de um objeto, uma referência é feita ao objeto da memória
- A referência aponta para o objeto
- Todas as variáveis do atributo são criadas e inicializadas com base no construtor usado

```
Card c = new Card();
```



suit = null
name = null
points = 0

Noções Básicas sobre o Exemplo de Coleta de Lixo

- Considerando o código abaixo, o que acontecerá na memória após a linha `c2 = c;` ?
- Quando for executada, a linha `c2 = c;` pegará a referência `c2` e fará com que ela referencie o mesmo objeto que `c`
- Isso classifica o objeto original `c2` como inútil, e a coleta de lixo cuida dele, removendo-o da memória

```
Card c  = new Card("Diamonds", "Four", 4);  
Card c2 = new Card("Spades", "Ace", 1);  
c2 = c;
```



Finalizadores

- Um finalizador é um código chamado pelo coletor de lixo quando ele determina que não há mais referências ao objeto
- Todos os objetos herdam um método `finalize()` do `java.lang.Object`
- Esse método não aceita parâmetros e é escrito para não executar nenhuma ação quando chamado

Finalizadores

- Substituir o método `finalize()` nas classes permite que você modifique o que acontece antes da coleta de lixo, como:
 - Notificar o usuário sobre a coleta de lixo que está prestes a ocorrer
 - Remover recursos não-Java, como fechar um arquivo

Exemplo do Método Finalize

- Este é um exemplo do método finalize() substituído em uma classe
- Ele fecha todos os arquivos associados e notifica o usuário da ocorrência da finalização

```
protected void finalize(){
    try{
        close(); //close all files
    }//fim try
    finally{
        System.out.println("Finalization has occurred");
    }//fim finally
} //fim do método finalize
```

Terminologia

- Os principais termos usados nesta aula foram:
 - Método de acesso
 - Classe
 - Construtor
 - Finalizadores
 - Coleta de lixo
 - Inicialização
 - Instanciar
 - Método

Terminologia

- Os principais termos usados nesta aula foram:
 - Método modificador
 - Palavra-chave new
 - Nulo
 - Objeto
 - Referência
 - Referência this

Resumo

- Nesta aula, você deverá ter aprendido a:
 - Reconhecer o formato geral correto de uma classe
 - Criar um objeto de uma classe
 - Criar métodos que sejam compilados sem erros
 - Retornar um valor de um método
 - Usar parâmetros em um método
 - Criar uma classe de driver e adicionar instâncias de classes de Objeto
 - Adicionar um construtor a uma classe
 - Aplicar o novo operador
 - Descrever coleta de lixo e finalizadores
 - Aplicar a referência this
 - Adicionar um construtor para inicializar um valor



The Oracle Academy logo is centered on a light gray background. It features the word "ORACLE" in a bold, orange, sans-serif font, with the letters "R", "A", and "C" having a unique, slightly slanted design. Below "ORACLE" is the word "Academy" in a dark gray, sans-serif font. The entire logo is framed by two horizontal dark gray bars, one at the top and one at the bottom.

ORACLE

Academy