



Java Foundations

7-2 Instanciando Objetos

ORACLE
Academy



Copyright © 2020, Oracle e/ou suas empresas afiliadas. Todos os direitos reservados.

Objetivos

- Esta lição abrange os seguintes objetivos:
 - Entender as consequências na memória da instanciação de objetos
 - Entender referências a objetos
 - Entender a diferença entre a memória stack e a memória heap
 - Entender como as Strings são objetos especiais





Obrigado por desenvolver o software para o meu banco! Seria um imenso prazer poder cumprimentá-lo pessoalmente

Graças a você, nossos clientes estão abrindo mais contas do que nunca



E as crianças nunca estiveram tão felizes!



Mais tarde nesta noite...

CRASH!

BANG!

BANG!

ORACLE
Academy

JFo 7-2
Instanciando Objetos

Copyright © 2020, Oracle e/ou suas empresas afiliadas. Todos os direitos reservados.

4



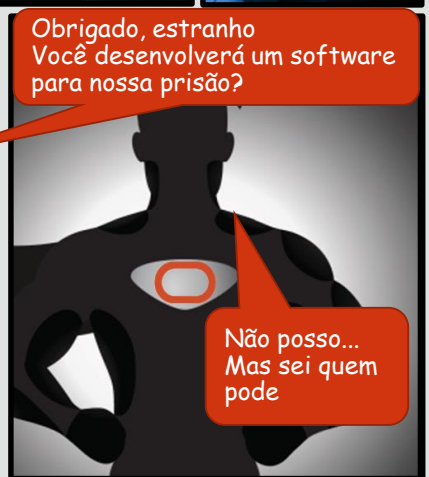
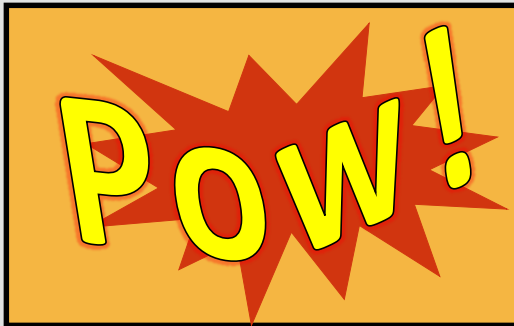
Ha! Ha! Ha! Roubar é divertido!

ORACLE
Academy

JFo 7-2
Instanciando Objetos

Copyright © 2020, Oracle e/ou suas empresas afiliadas. Todos os direitos reservados.

5



ORACLE
Academy

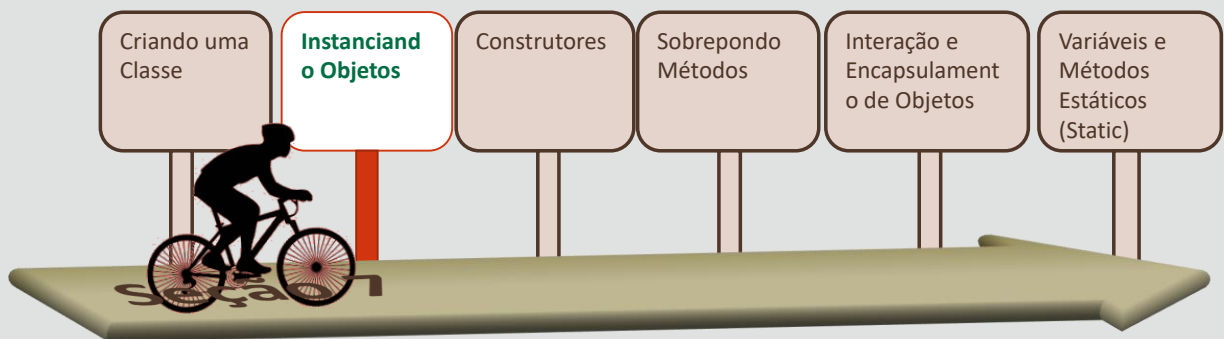
JFo 7-2
Instanciando Objetos

Copyright © 2020, Oracle e/ou suas empresas afiliadas. Todos os direitos reservados.

6

Tópicos

- **Objetos na Memória**
- Referências a Objetos e Gerenciamento de Memória
- Instanciando Strings



ORACLE
Academy

JFo 7-2
Instanciando Objetos

Copyright © 2020, Oracle e/ou suas empresas afiliadas. Todos os direitos reservados.

7

Descrevendo um Prisioneiro

- Propriedades:

- Nome
- Altura
- Anos de Condenação



- Comportamentos:

- Pense no que eles fizeram

Exercício 1, Parte 1

- Crie um novo projeto Java
- Crie uma classe PrisonTest com um método principal
- Crie uma classe Prisoner com base na descrição do slide anterior
- Instancie dois prisioneiros e atribua a eles as seguintes propriedades:



Variável: bubba
Nome: Bubba
Altura: 2,08 m
(6'10")
Sentença: 4 anos



Variável: twitch
Nome: Twitch
Altura: 1,73 m
(5'8")
Sentença: 3 anos

ORACLE
Academy

JFo 7-2
Instanciando Objetos

Copyright © 2020, Oracle e/ou suas empresas afiliadas. Todos os direitos reservados.

9

É mais fácil programar a altura em metros.



Exercício 1, Parte 2

- É possível os prisioneiros enganarem a segurança fingindo ser outro prisioneiro?
 - Escreva uma instrução de impressão com uma expressão booliana que teste `bubba == twitch`
 - Altere as propriedades de `twitch` para que elas correspondam às de `bubba`
 - Em seguida, teste a igualdade desses objetos novamente



Variável: `bubba`
Nome: `Bubba`
Altura: `2,08 m`
`(6'10")`
Sentença: `4 anos`



Variável: `twitch`
Nome: `Bubba`
Altura: `2,08 m`
`(6'10")`
Sentença: `4 anos`

Programando a Classe Prisoner

- Sua classe pode ser parecida com esta:

```
public class Prisoner {  
    public String name;  
    public double height;  
    public int sentence;  
  
    public void think(){  
        System.out.println("Terei minha vingança.");  
    }//fim do método think  
}//fim da classe Prisoner
```

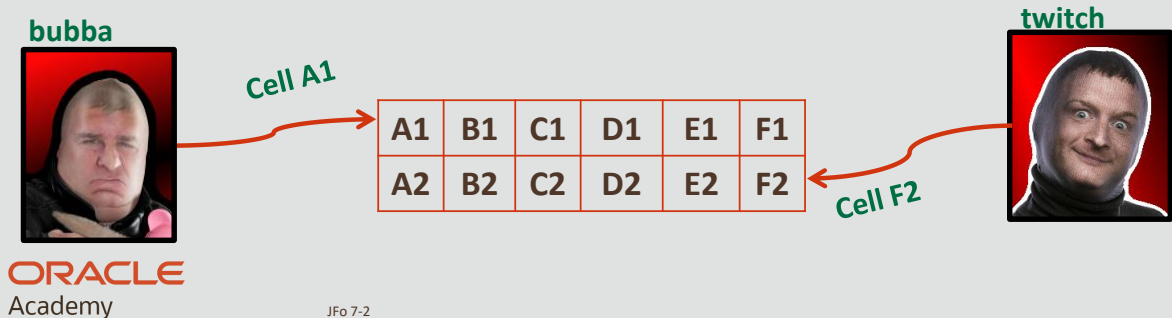
Representação do Prisioneiro

- A expressão booleana `bubba == twitch` é `false`
 - A segurança não foi enganada pelos prisioneiros que compartilham as mesmas propriedades
 - A segurança percebeu que cada prisioneiro era um objeto único
- Como isso é possível?

```
public class PrisonTest {  
    public static void main(String[] args){  
        Prisoner bubba = new Prisoner();  
        Prisoner twitch = new Prisoner();  
  
        ...  
        System.out.println(bubba == twitch); //falso  
    } //fim do método main  
} //fim da classe PrisonTest
```

Localização dos prisioneiros

- Os prisioneiros vivem em celas
- Uma cela disponível é atribuída a novos prisioneiros
- Se um prisioneiro viver em uma única cela, ele será um objeto exclusivo



JFo 7-2
Instanciando Objetos

Copyright © 2020, Oracle e/ou suas empresas afiliadas. Todos os direitos reservados.

13

Localizações do Objeto Prisoner

- As celas são como localizações na memória
- O instanciamento de um Prisoner preenche um local disponível na memória com o novo objeto Prisoner

```
public class PrisonTest {  
    public static void main(String[] args){  
        Prisoner bubba = new Prisoner();  
        Prisoner twitch = new Prisoner();  
    } //fim do método main  
} //fim da classe PrisonTest
```

bubba



Cell A1

A1	B1	C1	D1	E1	F1
A2	B2	C2	D2	E2	F2

twitch



Cell F2

A palavra-chave new


- A palavra-chave new aloca memória disponível para armazenar um objeto recém-criado
- Os desenvolvedores Java não precisam saber a localização de um objeto na memória
 - Só precisamos saber a variável do objeto
 - Mas ainda podemos imprimir endereços da memória

```
public class PrisonTest {  
    public static void main(String[] args){  
        Prisoner bubba = new Prisoner();  
        Prisoner twitch = new Prisoner();  
        System.out.println(bubba);    //prisonertest.Prisoner@15db9742  
        System.out.println(twitch);    //prisonertest.Prisoner@6d06d69c  
    }//fim do método main  
}//fim da classe PrisonTest
```


Endereços na memória

Objetos com as Mesmas Propriedades

- Os objetos podem compartilhar as mesmas propriedades
- Mas isso não significa que eles sejam iguais
- Desde que você use a palavra-chave `new` durante a instanciação...
 - Você terá objetos exclusivos
 - Cada objeto terá uma localização diferente na memória



Variável: **bubba**
Nome: Bubba
Altura: 2,08 m
(6'10")
Sentença: 4 anos
Endereço na memória:
@15db9742



Variável: **twitch**
Nome: Bubba
Altura: 2,08 m
(6'10")
Sentença: 4 anos
Endereço na memória:
@6d06d69c

ORACLE
Academy

JFo 7-2
Instanciando Objetos

Copyright © 2020, Oracle e/ou suas empresas afiliadas. Todos os direitos reservados.

16

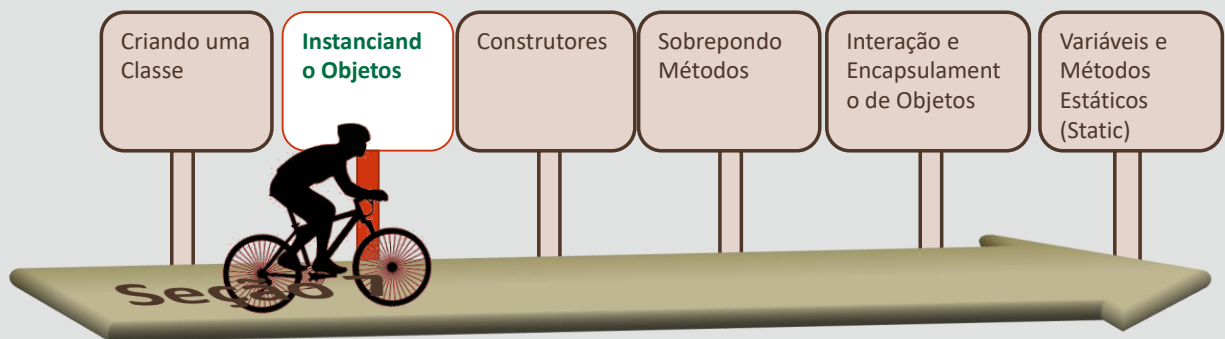
Comparando Objetos

- Se você comparar dois objetos usando o operador ==...
 - Você está verificando se os respectivos endereços na memória são iguais
 - Você não está verificando se os respectivos campos são iguais
- A expressão booleana `bubba == twitch` é `false` porque...
 - Os endereços na memória `@15db9742` e `@6d06d69c` são diferentes
 - Não importa se `bubba` e `twitch` compartilham as mesmas propriedades

```
public class PrisonTest {  
    public static void main(String[] args){  
        Prisoner bubba = new Prisoner();  
        Prisoner twitch = new Prisoner();  
        ...  
        System.out.println(bubba == twitch); //falso  
    } //fim do método main  
} //fim da classe PrisonTest
```

Tópicos

- Objetos na Memória
- **Referências a Objetos e Gerenciamento de Memória**
- Instanciando Strings



ORACLE
Academy

JFo 7-2
Instanciando Objetos

Copyright © 2020, Oracle e/ou suas empresas afiliadas. Todos os direitos reservados.

18

Acessando Objetos Usando uma Referência



Os objetos são acessados usando variáveis de referência. Uma boa analogia seria usar um controle remoto (a referência) para operar uma câmera (o objeto). Os botões no controle remoto são usados para acionar um comportamento específico da câmera. Por exemplo, você pode usá-lo para chamar as funções parar, reproduzir ou gravar da câmera.

Trabalhando com Referências a Objetos

1

Escolha o controle remoto para ter acesso à câmera

1

Crie um objeto Camera e faça uma referência a ele

```
Camera remotel = new Camera();
```

2

Pressione os controles remotos para que a câmera faça algo

2

Chame um método para que o objeto Camera faça algo

```
remotel.play();
```

ORACLE
Academy

JFo 7-2
Instanciando Objetos

Copyright © 2020, Oracle e/ou suas empresas afiliadas. Todos os direitos reservados.

20

Vamos examinar a analogia de usar um controle remoto para operar um dispositivo eletrônico. Para operar um dispositivo eletrônico com um controle remoto, você precisa:

1. Escolher o controle remoto (e ativá-lo).
2. Pressionar um botão no controle remoto para fazer algo na câmera.

Da mesma forma, para fazer algo com um objeto Java, você precisa:

1. Obter seu “controle remoto” (denominado referência).
2. Pressionar os respectivos “botões” (denominados métodos).

Trabalhando com Referências a Objetos: Exemplo 1



```
Camera remote1 = new Camera();  
Camera remote2 = new Camera();  
remote1.play();  
remote2.play();
```

Existem dois objetos Camera

Existem dois objetos camera neste exemplo. Cada câmera tem seu próprio controle remoto exclusivo. `remote2` não funcionará na câmera de `remote1`, e `remote1` não funcionará na câmera de `remote2`. Isso reflete como, no Java, dois objetos diferentes podem ser instanciados com suas próprias referências exclusivas. Essas referências podem ser usadas para chamar métodos em seus respectivos objetos.

Trabalhando com Referências a Objetos: Exemplo 2



```
Camera remote1 = new Camera();  
Camera remote2 = remote1;  
remote1.play();  
remote2.stop();
```

O diagrama mostra outro aspecto importante de como as referências funcionam. Neste exemplo, um objeto `Camera` é criado com a referência `remote1`. Essa referência é, então, atribuída, a outra referência de `Camera`, `remote2`. As duas referências `remote1` e `remote2` estão associadas ao mesmo objeto `Camera`. Chamar os métodos usando uma das duas referências afeta o mesmo objeto `Camera`. Chamar `remote1.play()` não é diferente de chamar `remote2.play()`.

Referências a Objetos Diferentes



Trabalhar com diferentes tipos de objetos (por exemplo, uma câmera e uma televisão) requer um controle remoto específico a esse tipo de objeto. No Java, você precisa de uma variável de referência do tipo correto do objeto a que está fazendo referência.

Referências a Objetos Diferentes: Exemplo

Tipo de referência Variável de referência Tipo de objeto

```
Camera remote1 = new Camera();  
remote1.menu();
```

```
TV remote2 = new TV();  
remote2.menu();
```

```
Prisoner bubba = new Prisoner();  
bubba.think();
```

Um prisioneiro não pode representar uma TV para enganar a segurança.

Referências a Objetos Diferentes: Exemplo

- O exemplo a seguir não é permitido porque...
 - O Tipo de Referência não corresponde ao Tipo de Objeto
 - Um prisioneiro e uma TV são coisas totalmente distintas



```
Prisoner twitch = new TV();
```

Um prisioneiro não pode representar uma TV para enganar a segurança.

Exercício 2

- Continue experimentando com a classe PrisonTest
- A segurança é enganada quando variáveis de referência mudam?
 - Instancie dois prisioneiros e atribua a eles as seguintes propriedades:
 - Teste a igualdade desses objetos
 - Em seguida, defina a variável de referência de bubba como igual a twitch
 - Teste a igualdade desses objetos novamente



Variável: bubba
Nome: Bubba
Altura: 2,08 m
(6'10")
Sentença: 4 anos



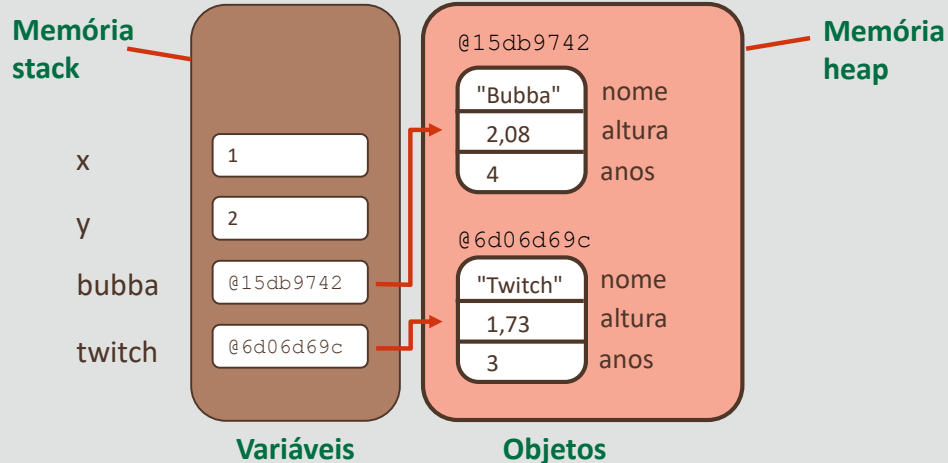
Variável: twitch
Nome: Twitch
Altura: 1,73 m
(5'8")
Sentença: 3 anos

Memória Stack e Memória Heap

- Para entender os resultados do Exercício 2, é preciso compreender os tipos de memória que o Java usa
- A Memória stack é usada para armazenar...
 - Variáveis de local
 - Primitivas
 - Referências a locais na memória heap
- A Memória heap é usada para armazenar...
 - Objetos

Referências e Objetos na Memória

```
int x = 1;  
int y = 2;  
Prisoner bubba = new Prisoner();  
Prisoner twitch = new Prisoner();  
...
```



ORACLE
Academy

JFo 7-2
Instanciando Objetos

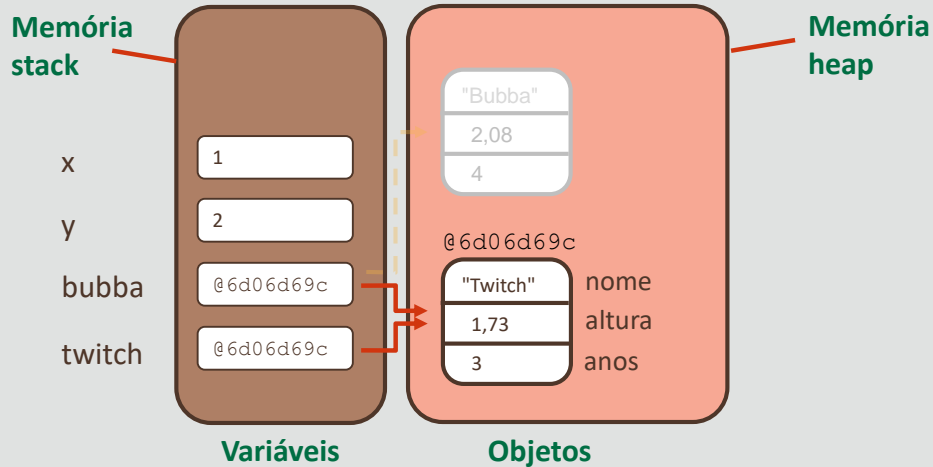
Copyright © 2020, Oracle e/ou suas empresas afiliadas. Todos os direitos reservados.

28

Este diagrama mostra como variáveis de referência apontam para determinado objeto na memória. Existem duas referências a objetos `Prisoner` apontando para dois objetos `Prisoner`. A memória stack contém variáveis locais, sejam variáveis de referência ou primitivas, e a memória heap contém objetos.

Atribuindo uma Referência a Outra Referência

```
bubba = twitch;
```



ORACLE
Academy

JFo 7-2
Instanciando Objetos

Copyright © 2020, Oracle e/ou suas empresas afiliadas. Todos os direitos reservados.

29

As variáveis de referência `bubba` e `twitch` agora apontam para o mesmo objeto.

Duas Referências, Um Objeto

- A partir da linha 14, bubba e twitch fazem referência ao mesmo objeto
- Qualquer uma dessas variáveis de referência poderia ser usada para acessar os mesmos dados

```
11 Prisoner bubba = new Prisoner();
12 Prisoner twitch = new Prisoner();
13
14 bubba = twitch;
15
16 bubba.name = "Bubba";
17 twitch.name = "Twitch";
19
20 System.out.println(bubba.name);      //Twitch
21 System.out.println(bubba == twitch); //verdade
```

A impressão de `bubba.name` faz "Twitch" ser impresso porque `bubba.name` e `twitch.name` fazem referência ao mesmo campo no mesmo objeto.


Duas Referências, Duas Primitivas

- As primitivas sempre são variáveis separadas
- Os valores das primitivas sempre ocupam locais diferentes na memória stack
- De maneira resumida, a linha 14 torna iguais os valores x e y das primitivas


```
11 int x;  
12 int y;  
13  
14 x = y;  
15  
16 x = 1;  
17 y = 2;  
19  
20 System.out.println(x);           //1  
21 System.out.println(x == y);     //falso
```

O que Aconteceu com Bubba?

- Se nenhuma outra variável de referência apontar para um objeto...
- O Java limpará automaticamente a memória que era ocupada por esse objeto
 - Esse processo denomina-se Coleta de Lixo
 - Os dados associados a esse objeto serão perdidos para sempre



Variável:
Nome: Bubba
Altura: 2,08 m
(6'10")
Sentença: 4 anos
Endereço na memória:
@15db9742



Variável: **twitch, bubba**
Nome: Bubba
Altura: 2,08 m
(6'10")
Sentença: 3 anos
Endereço na memória:
@6d06d69c

ORACLE
Academy

JFo 7-2
Instanciando Objetos

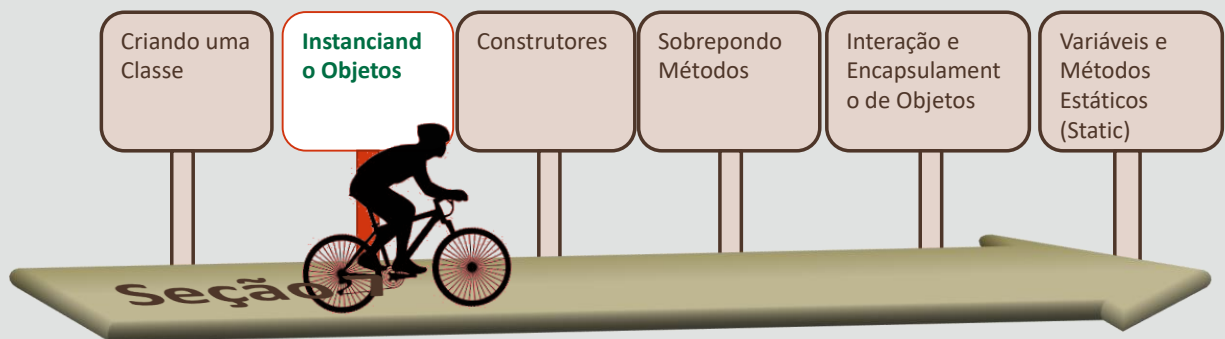
Copyright © 2020, Oracle e/ou suas empresas afiliadas. Todos os direitos reservados.

32

Em linguagens como C++, é preciso que você limpe a memória manualmente.

Tópicos

- Objetos na Memória
- Referências a Objetos e Gerenciamento de Memória
- **Instanciando Strings**



ORACLE
Academy

JFo 7-2
Instanciando Objetos

Copyright © 2020, Oracle e/ou suas empresas afiliadas. Todos os direitos reservados.

33

Strings São Objetos Especiais

- A impressão de uma referência de String imprime a String propriamente dita, em vez de imprimir o endereço do objeto na memória
- As strings podem ser instanciadas com a palavra-chave new
 - Mas você não deve fazer isso
- As strings devem ser instanciadas sem a palavra-chave new
 - Em termos de memória, isso é mais eficiente
 - Explicaremos por que nos próximos slides

```
String s1 = new String("Teste");
```

```
String s2 = "Teste";
```



Exercício 3

- Continue experimentando com a classe `PrisonTest`
- Veja você mesmo as consequências de Strings na memória
 - Instancie dois prisioneiros com os nomes mostrados abaixo
 - Defina seus nomes com a palavra-chave `new` e teste a igualdade dessas Strings usando `==`
 - Defina seus nomes com a palavra-chave `new` e teste a igualdade dessas Strings usando `==`



Variável: bubba
Nome: Bubba
Altura: 2,08 m
(6'10")
Sentença: 4 anos

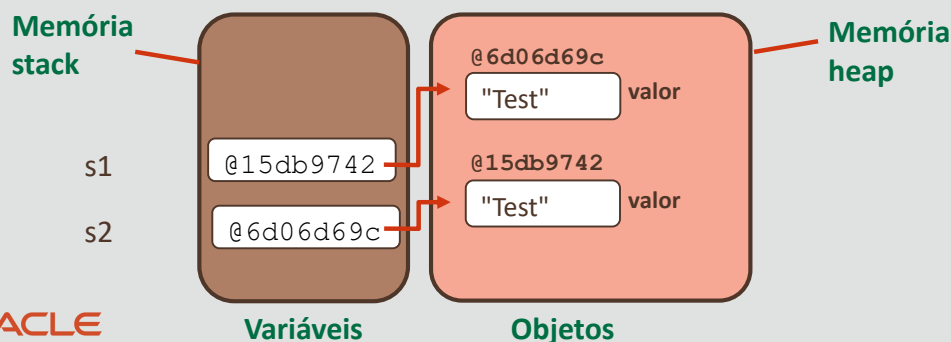


Variável: twitch
Nome: Bubba
Altura: 2,08 m
(6'10")
Sentença: 4 anos

Instanciando Strings com a Palavra-chave new

- O uso da palavra-chave new cria duas referências diferentes a dois objetos diferentes

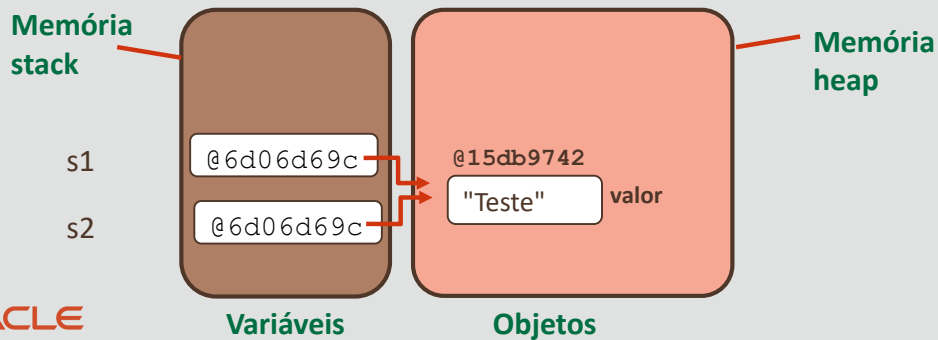
```
String s1 = new String("Test");  
String s2 = new String("Test");
```



Instanciando Strings sem a Palavra-chave new

- O Java reconhece automaticamente Strings idênticas e economiza espaço na memória classificando o objeto uma única vez
- Isso cria duas referências diferentes a um objeto

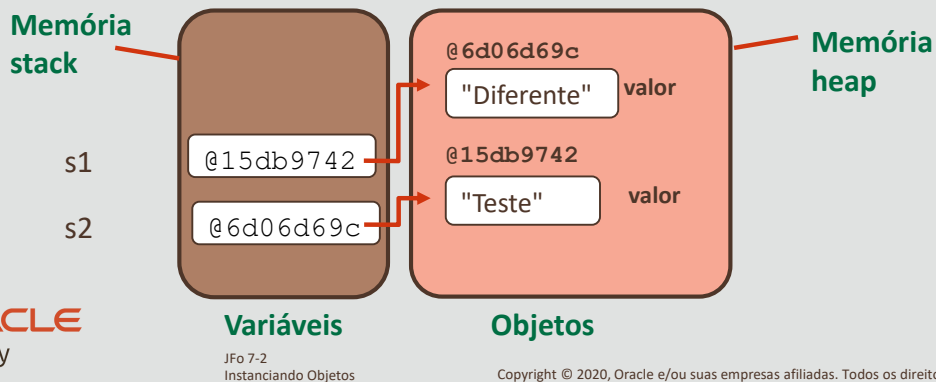
```
String s1 = "Teste";  
String s2 = "Teste";
```



Referências de String

- A alteração de uma String usando uma referência não afeta outras referências
- O Java aloca nova memória de outra String

```
String s1 = "Teste";  
String s2 = "Teste";  
s1 = "Diferente";
```



Resumo

- Nesta lição, você deverá ter aprendido a:
 - Entender as consequências na memória da instanciação de objetos
 - Entender referências a objetos
 - Entender a diferença entre a memória stack e a memória heap
 - Entender como as Strings são objetos especiais



The logo for Oracle Academy is centered on a light gray background. It features the word "ORACLE" in a bold, orange, sans-serif font. Below it, the word "Academy" is written in a smaller, 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