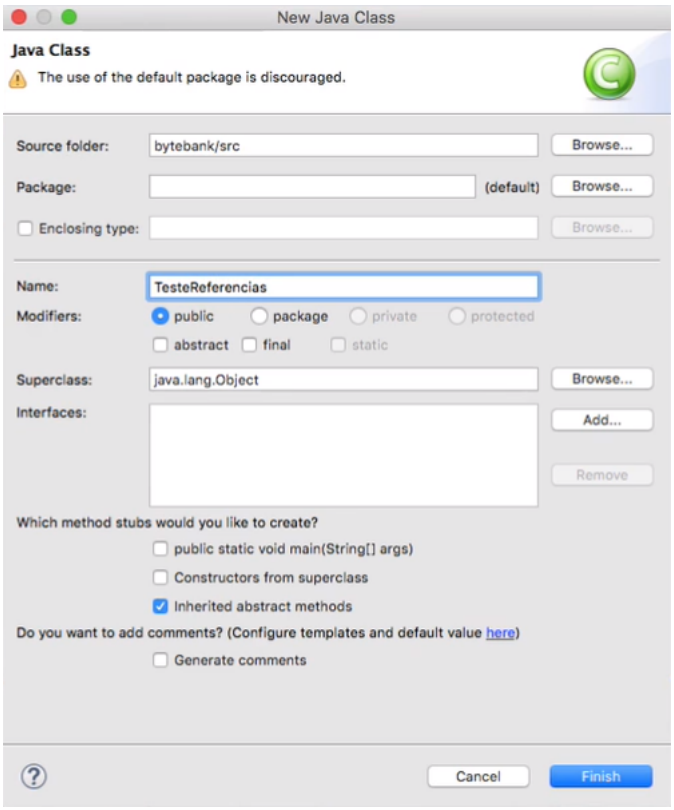
08

Referências vs Objetos

Transcrição

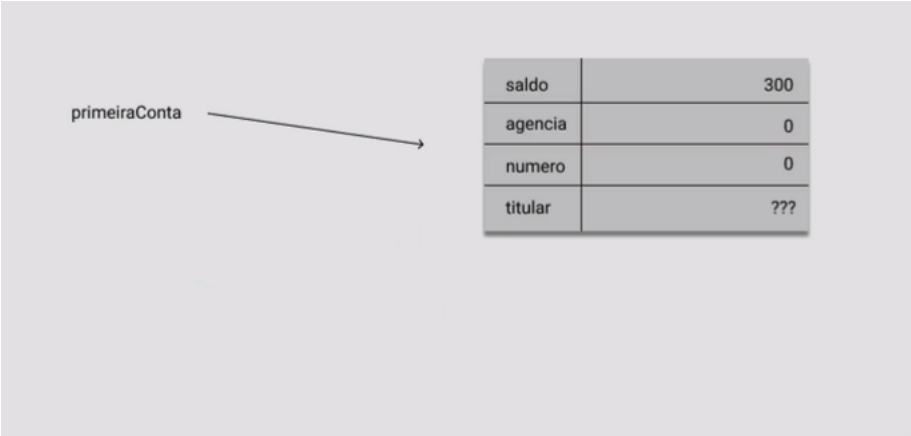
Nos atentaremos para um detalhe que causa confusão mesmo em programadores mais experientes. Criaremos uma nova classe chamada `TesteReferencias`.



Nesta nova classe, adicionaremos a `main`, e criaremos uma conta chamada `primeiraConta`. Estipularemos o valor de `300` para `saldo`.

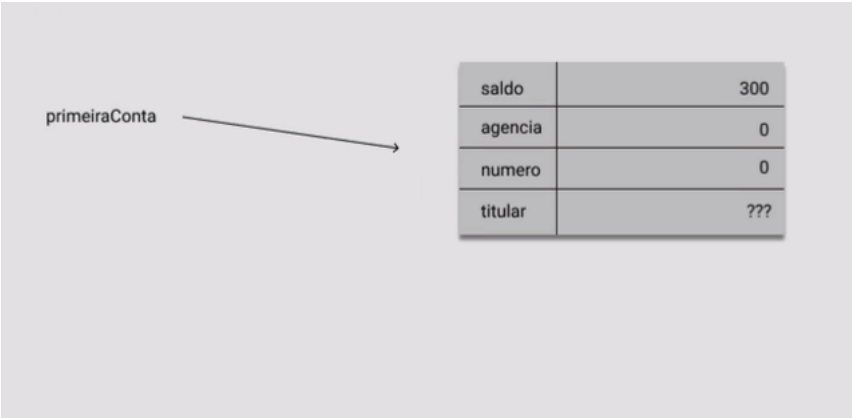
```
public class TesteReferencias {
    public static void main(String [] args) {
        Conta primeiraConta = new Conta();
        primeiraConta.saldo = 300;

        System.out.println("saldo da primeira: " + primeiraConta.saldo);
```

Temos a seguinte representação no cartão cinza:



Executaremos o programa e verificaremos que o código está funcional.

Nos atentaremos para uma questão importante: a diferença entre o tipo (`Conta`) e a variável desse tipo (`primeiraConta`). A variável não é um objeto `Conta` , e sim, uma indicação a um objeto específico, uma *referência* de um objeto. Sua representação gráfica seria a flecha que referencia o objeto.



Observem a seguinte declaração:

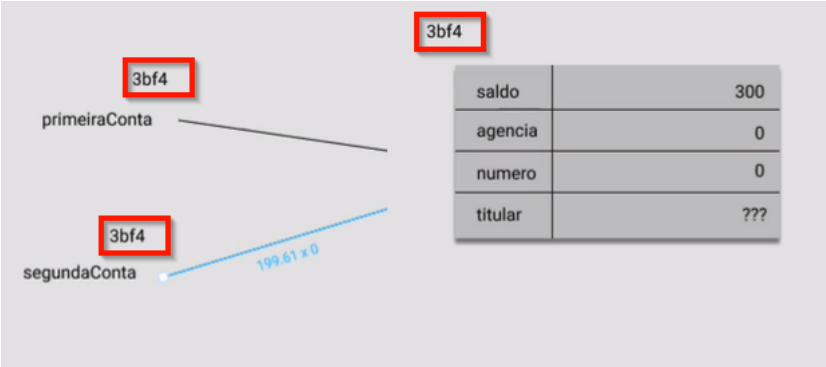
```
public class TesteReferencias {
    public static void main(String[] args) {
        Conta primeiraConta = new Conta();
        primeiraConta.saldo = 300;

        System.out.println("saldo da primeira: " + primeiraConta.saldo);

        Conta segundaConta = primeiraConta;
    }
}
```

A princípio, podemos pensar que esta declaração gera uma cópia da `primeiraConta` para a `segundaConta` e teríamos uma espécie de "clone de objeto".

Lembrem-se `=` no Java copia o que está na direita e cola na esquerda.

A questão é que não há um objeto `Conta` à direita, e sim uma **referência**. O que copiamos é a referência para um mesmo objeto.

Pense da seguinte forma: existe uma espécie de Id dos objetos, que chamaremos de `3bf4`. A variável `primeiraConta` possui o valor `3bf4`, fazendo referência ao `Id` do objeto. Quando declaramos que `primeiraConta = segundaConta`, na verdade estamos copiando esse Id `3bf4` que é a referência, e não o objeto em si.



O que temos são duas referências para o mesmo objeto. É como se duas cartas fossem endereçadas ao mesmo local. Embora sejam cartas diferentes, possuem o mesmo destino.

Veremos qual é o resultado dessa dupla referenciação no nosso código. Faremos o `sysout` no `saldo` de `segundaConta`.

```
public class TesteReferencias {
    public static void main(String[] args) {
        Conta primeiraConta = new Conta();
        primeiraConta.saldo = 300;

        System.out.println("saldo da primeira: " + primeiraConta.saldo);

        Conta segundaConta = primeiraConta;
    }
}
```

Ao executarmos o código, veremos que o programa irá imprimir o valor `300`, pois temos duas variáveis, e não dois objetos.

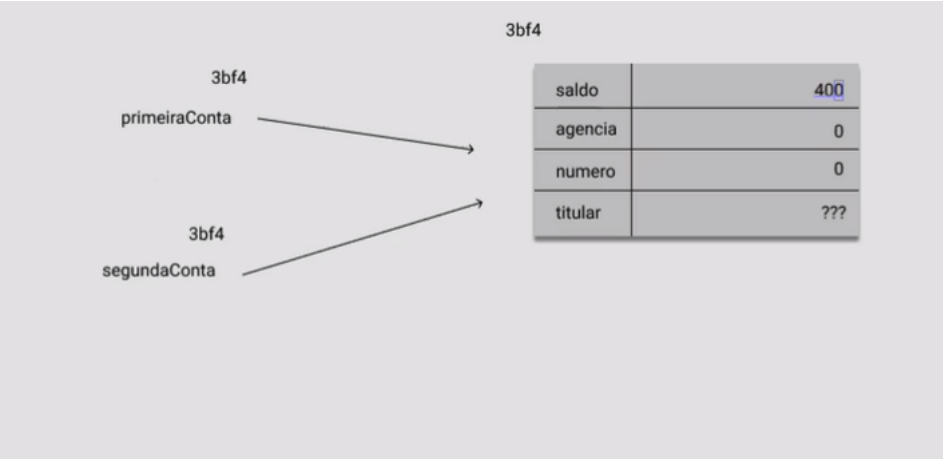
Prosseguiremos com um desafio mais complicado. Primeiramente, vamos incluir mais `100` reais em `sengundaConta`.

```
public class TesteReferencias {
    public static void main(String[] args) {
        Conta primeiraConta = new Conta();
        primeiraConta.saldo = 300;

        System.out.println("saldo da primeira: " + primeiraConta.saldo);

        Conta segundaConta = primeiraConta;

        System.out.println("saldo da segunda conta: " + segundaConta.saldo);
    }
}
```

Ao executarmos a aplicação teremos um saldo de `400` reais para `segundaConta`. Feito isso, iremos acionar o `sysout` no saldo de `primeiraConta`.

```
public class TesteReferencias {
    public static void main(String[] args) {
        Conta primeiraConta = new Conta();
        primeiraConta.saldo = 300;

        System.out.println("saldo da primeira: " + primeiraConta.saldo);

        Conta segundaConta = primeiraConta;

        System.out.println("saldo da segunda conta: " + segundaConta.saldo);

        segundaConta.saldo += 100;
        System.out.println("saldo da segunda conta " + segundaConta.saldo);

        System.out.println(primeiraConta.saldo);
    }
}
```

Quando executarmos o programa teremos um saldo de `300` ou `400`? Continuaremos tendo duas referências para apenas um objeto, portanto, o saldo será de `400`.



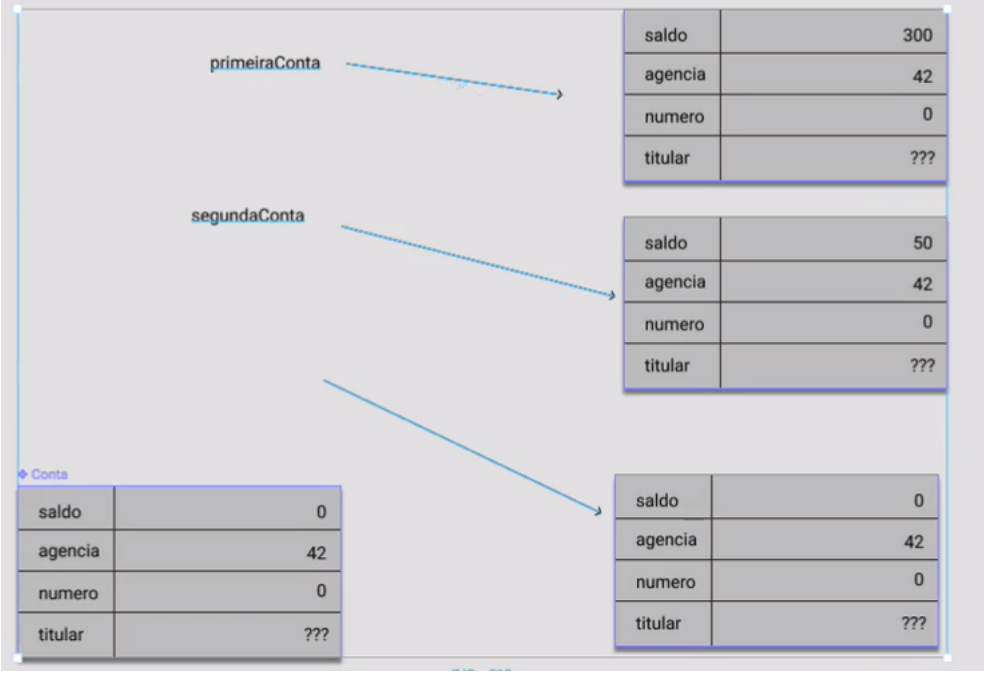
Podemos verificar se `primeiraConta` possui as mesmas informações de `segundaConta`, fazendo um `if`.

```
public class TesteReferencias {
    public static void main(String[] args) {
        Conta primeiraConta = new Conta();
        primeiraConta.saldo = 300;

        System.out.println("saldo da primeira: " + primeiraConta.saldo);

        Conta segundaConta = primeiraConta;

        System.out.println("saldo da segunda conta: " + segundaConta.saldo);

        segundaConta.saldo += 100;
        System.out.println("saldo da segunda conta " + segundaConta.saldo);
```

```
        System.out.println(primeiraConta.saldo);

        if(primeiraConta == segundaConta) {
            System.out.println("é a mesma conta");
        }
    }
}
```

Os números de referência são iguais, portanto, são a mesma conta, fazem referência ao mesmo objeto neste código.

Se formos na classe `CriaConta` que fizemos nas aulas passadas, teremos as variáveis `primeiraConta` e `segundaConta` referenciando objetos diferentes. Há dois `new` s no nosso código, um para cada referência. Podemos perceber a individualidade das referências na representação gráfica do código anterior.



Na classe `CriaConta` , iremos fazer um `if` e um `else` para vermos como o nosso código se comporta.

```
public class CriaConta {

    public static void main(String[] args) {
        Conta primeiraConta = new Conta();
        primeiraConta.saldo = 200;
        System.out.println(primeiraConta.saldo);

        primeiraConta.saldo += 100;
        System.out.println(primeiraConta.saldo);

        Conta segundaConta = new Conta();
        segundaConta.saldo = 50;

        System.out.println("primeira conta tem " + primeiraConta.saldo);
        System.out.println("segunda conta tem " + segundaConta.saldo);

        System.out.println(primeiraConta.agencia);
        System.out.println(primeiraConta.numero);
```

```
        System.out.println(segundaConta.agencia);

        segundaConta.agencia = 146;
        System.out.println("agora a segunda conta está na agencia " + segundaConta.agencia);

        if(primeiraConta == segundaConta) {
            System.out.println("mesma conta");
        } else {
            System.out.println("contas diferentes");
        }
    }
}
```

O resultado da execução dessa aplicação será `contas diferentes`. O sinal `==` irá **comparar referências**, e não objetos.

De volta à classe `TesteReferencias`, acionaremos o `sysout` para `primeiraConta`.

```
public class TesteReferencias {
    public static void main(String[] args) {
        Conta primeiraConta = new Conta();
        primeiraConta.saldo = 300;

        System.out.println("saldo da primeira: " + primeiraConta.saldo);

        Conta segundaConta = primeiraConta;

        System.out.println("saldo da segunda conta: " + segundaConta.saldo);

        segundaConta.saldo += 100;
        System.out.println("saldo da segunda conta " + segundaConta.saldo);

        System.out.println(primeiraConta.saldo);

        if(primeiraConta == segundaConta) {
            System.out.println("é a mesma conta");
        }
        System.out.println(primeiraConta);
    }
}
```

Ao executarmos o programa, veremos que o Java irá imprimir `Conta@15db9742`. Estamos fazendo uma referência a um objeto do tipo `Conta`, e que este objeto está dentro de uma "gaveta de memória" identificada por essa sequência numérica, estamos falando da mesma ideia do "Id" que vimos anteriormente com o `3bf4`. Faremos o mesmo procedimento com `segundaConta`.

```
public class TesteReferencias {
    public static void main(String[] args) {
        Conta primeiraConta = new Conta();
        primeiraConta.saldo = 300;

        System.out.println("saldo da primeira: " + primeiraConta.saldo);

        Conta segundaConta = primeiraConta;
```

```
        System.out.println("saldo da segunda conta: " + segundaConta.saldo);

        segundaConta.saldo += 100;
        System.out.println("saldo da segunda conta " + segundaConta.saldo);

        System.out.println(primeiraConta.saldo);

        if(primeiraConta == segundaConta) {
            System.out.println("é a mesma conta");
        }

        System.out.println(primeiraConta);
        System.out.println(segundaConta);
    }
}
```

O resultado impresso será o mesmo, pois são duas referências apontando para o mesmo objeto.

Faremos o mesmo procedimento na classe `CriaConta` para que vejamos a diferença de comportamento de um código para o outro.

```
public class CriaConta {
    public static void main(String[] args) {
        Conta primeiraConta = new Conta();
        primeiraConta.Saldo = 200;
        System.out.println(primeiraConta.saldo);

        primeiraConta.saldo += 100;
        System.out.println(primeiraConta.saldo);

        Conta segundaConta = new Conta();
        segundaConta.saldo = 50;

        System.out.println("primeira conta tem " + primeiraConta.saldo);
        System.out.println("segunda conta tem " + segundaConta.saldo);

        System.out.println(primeiraConta.agencia);
        System.out.println(primeiraConta.numero);

        System.out.println(segundaConta.agencia);

        segundaConta.agencia = 146;
        System.out.println("agora a segunda conta está na agencia " + segundaConta.agencia);

        if(primeiraConta == SegundaConta) {
            System.out.println("mesma conta");
        } else {
```

```
            System.out.println("contas diferentes");
        }

        System.out.println(primeiraConta);
        System.out.println(segundaConta);
    }
}
```

Quando iniciamos a aplicação, veremos que o Java imprimiu os valores `Conta@15db972` e `Conta@6d06d69c` diferentes , afinal, estamos trabalhando com referências orientadas para dois objetos diferentes.

Façamos um teste: colocaremos a mesma quantidade de `saldo` e a mesmo número em `agencia` nas duas contas, e verificaremos se a numeração `Id` continua diferente.

```
public class CriaConta {
    public static void main(String[] args) {
        Conta primeiraConta = new Conta();
        primeiraConta.saldo = 200;
        System.out.println(primeiraConta.saldo);

        primeiraConta.saldo += 100;
        System.out.println(primeiraConta.saldo);

        Conta segundaConta = new Conta();
        segundaConta.saldo = 300;

        System.out.println("primeira conta tem " + primeiraConta.saldo);
        System.out.println("segunda conta tem " + segundaConta.saldo);

        segundaConta.agencia = 146;
        System.out.println(primeiraConta.agencia);
        System.out.println(primeiraConta.numero);

            System.out.println(segundaConta.agencia);

        segundaConta.agencia = 146;
        System.out.println("agora a segunda conta está na agencia " + segundaConta.agencia);

        if(primeiraConta == segundaConta) {
            System.out.println("mesma conta");
        } else {
            System.out.println("contas diferentes");
        }

        System.out.println(primeiraConta);
        System.out.println(segundaConta);
    }
}
```

Temos o saldo de `300` para ambas as contas, bem como a agencia de número `146` , e a numeração das contas continua diferente, sendo `Conta@15db9742` e `Conta@6d06d69c` . O Java não irá comparar objetos, e sim **referências**.