01

Composição de Objetos

Transcrição

Demos pouca atenção para o atributo `titular` do tipo `String`. Podemos utilizar esse atributo dentro da classe `TestaMetodo` sem dificuldades.

```
public class TestaMetodo {

    System.out.println(contaDaMarcela.saldo);
    System.out.println(contaDoPaulo.saldo);

    contaDoPaulo.titular = "paulo silveira";
    System.out.println(contaDoPaulo.titular);
}
```

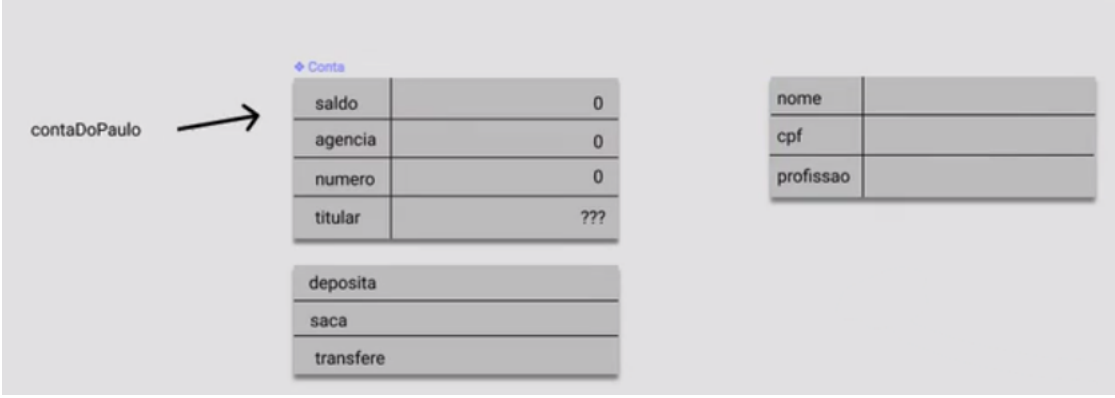
Foi dito que Java zera o valor dos atributos quando acionamos a palavra-chave `new`. Agora compreenderemos melhor o que acontece no caso dos tipos não numéricos como `String`.

Suponhamos que a conta bancária do **ByteBank** além de guardar as informações de saldo, agência, número e o nome do titular, também guardará o *CPF* do titular e sua *profissão*. Uma possível solução seria incluir esses novos atributos à classe `Conta`.

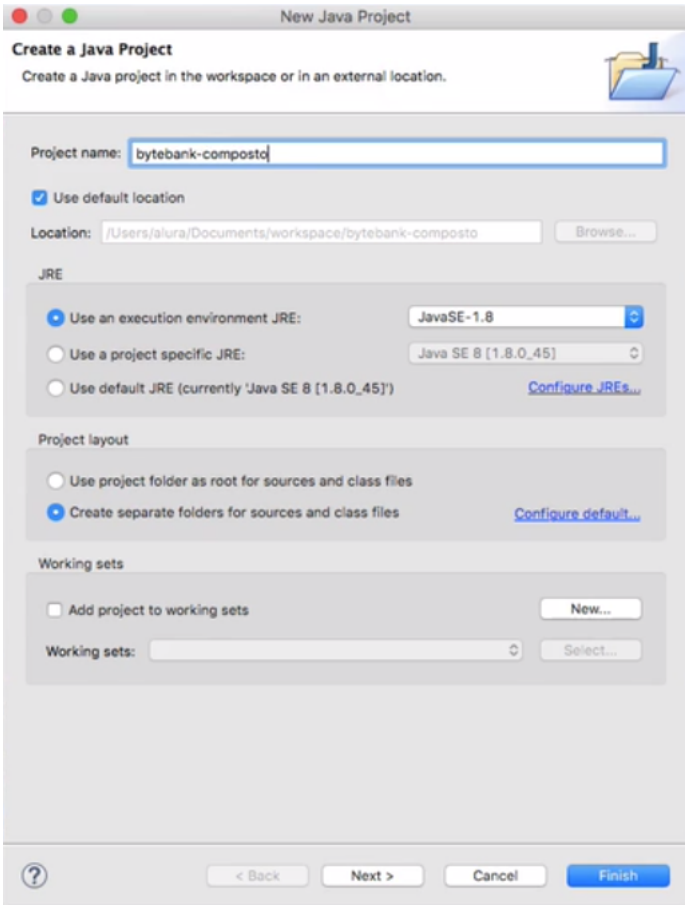
```
public class Conta {
    double saldo;
    int agencia;
    int numero;
    String titular;
    String cpf;
    String profissão;

    <!-- ... -->
```

Percebam que a classe começa a ficar "inchada" e com muitas informações que não dizem respeito exatamente a uma conta bancária, como a profissão do titular e seu CPF.

Para resolver essa questão, podemos criar um tipo novo chamado `Cliente`, que terá os atributos de `nome`, `cpf` e `profissão`.

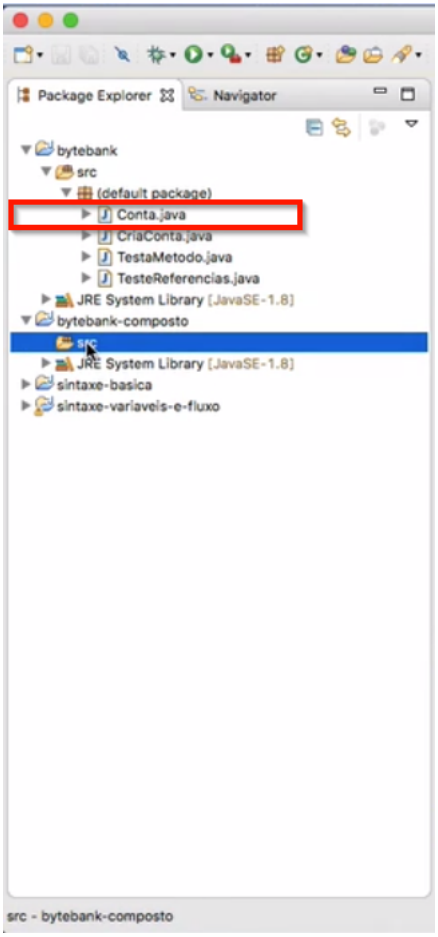


Para separar bem a organização das contas bancárias e dos titulares, criaremos um novo projeto Java intitulado `bytebank-composto`.



Na área do *Package Explorer*, selecionaremos a classe `Conta.Java` e a copiaremos utilizando o atalho "Ctrl + C", e a colaremos na pasta `src` via atalho "Ctrl + V".

Atenção! Quando temos duas classes com o mesmo nome, ainda que em projetos diferentes, podemos nos confundir e fazer edições na classe errada. O Eclipse disponibiliza um recurso para que se feche o projeto que não está sendo trabalhado. Na área *Package Navigator*, basta clicar com o botão direito sobre o nome do projeto e selecionar a opção "Close Project".



No novo projeto, criaremos uma nova classe intitulada `Cliente`. Tal classe conterá os atributos de `nome`, `cpf` e `profissao` como já foi dito.

```
public class Cliente {
    String nome;
    String cpf;
    String profissao;
}
```

Iremos estabelecer uma relação entre `Conta` e `Cliente`, ou seja, toda `Conta` faz uma referência a um `Cliente`.

Não é mais interessante para o nosso projeto que o atributo `titular` seja uma `String`, e sim que faça referência a um cliente específico.

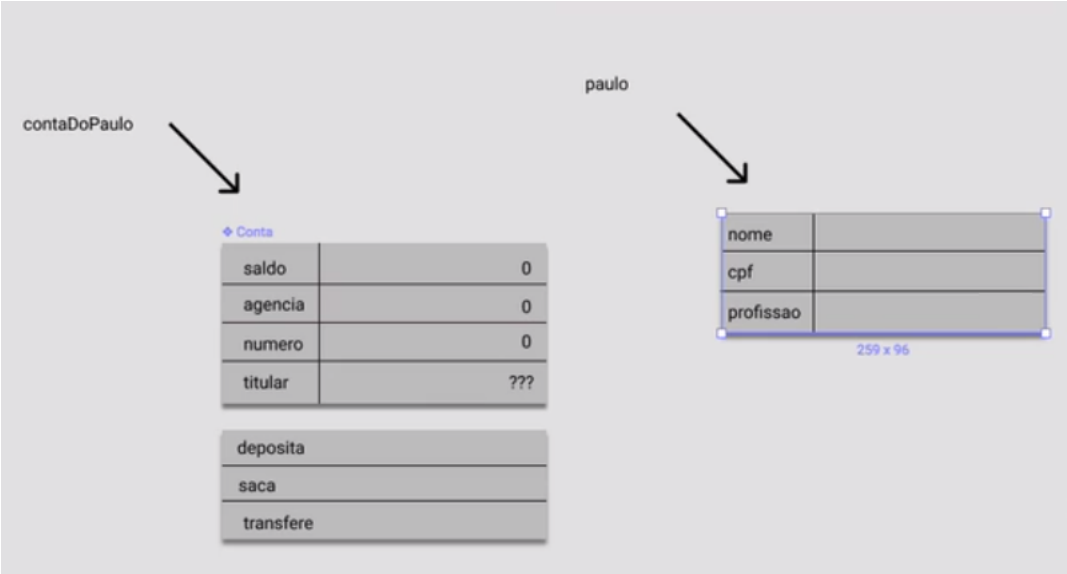
Criaremos uma nova classe chamada `TestaBanco`. Faremos um `main` e criaremos uma referência para um cliente que chamaremos de `paulo`.

```
public class TestaBanco {
    public static void main(String[] args) {
        Cliente paulo = new Cliente();
    }
}
```

Vamos popular este objeto, criando seus atributos.

```
public class TestaBanco {
    public static void main(String[] args) {
        Cliente paulo = new Cliente();
        paulo.nome = "Paulo Silveira";
        paulo.cpf = "222.222.222-22";
        paulo.profissao = "programador";
    }
}
```

A referência para este cliente está populada com os dados estipulados.



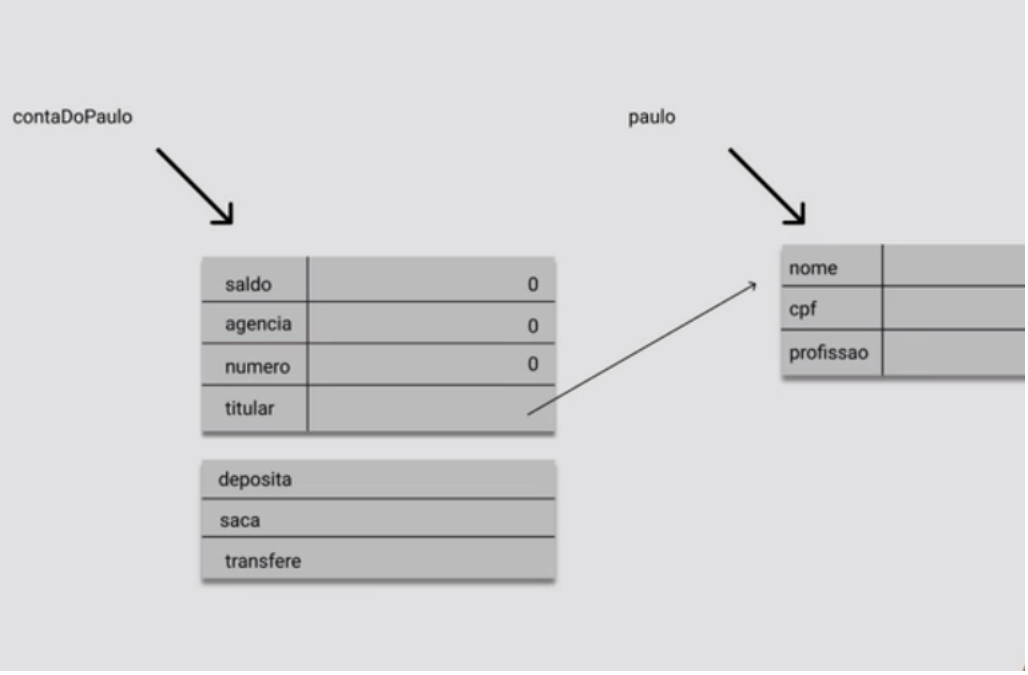
Criaremos a conta do cliente referido, e depositaremos um valor de `100` reais.

```
public class TestaBanco {
    public static void main(String[] args) {
        Cliente paulo = new Cliente();
        paulo.nome = "Paulo Silveira";
        paulo.cpf = "222.222.222-22";
        paulo.profissao = "programador";

        Conta contaDoPaulo = new Conta();
        contaDoPaulo.deposita(100);
    }
}
```

Agora temos uma classe `Conta` e outra `Cliente`. Queremos que o atributo `titular` não seja uma `String`, mas sim, uma referência para um objeto do tipo `Cliente`. Em nossa classe `Conta`, alteraremos o tipo do atributo `titular` para ser do tipo `Cliente`.

```
public class Conta {
    double saldo;
    int agencia;
    int numero;
    Cliente titular;

    <!-- ... -->
```

A nossa ideia pode ser ilustrada pelo diagrama. Queremos que o atributo `titular` faça uma referência a um cliente específico, ou seja, iremos fazer uma associação entre objetos.



Faremos essa associação na classe `TestaBanco`, montando, assim, a nossa composição de objetos.

```
public class TestaBanco {
    public static void main(String[] args) {
        Cliente paulo = new Cliente();
        paulo.nome = "Paulo Silveira";
        paulo.cpf = "222.222.222-22";
        paulo.profissao = "programador";

        Conta contaDoPaulo = new Conta();
        contaDoPaulo.deposita(100);

        contaDoPaulo.titular = paulo;
        System.out.println(contaDoPaulo.titular.nome);
    }
}
```

Ao executarmos a aplicação, veremos que será impresso o resultado `Paulo Silveira`.

Para testarmos o comportamento no programa, tentaremos imprimir apenas o `titular`.

```
public class TestaBanco {
    //...
    contaDoPaulo.titular = paulo;
      System.out.println(contaDoPaulo.titular.nome);
    System.out.println(contaDoPaulo.titular);
    }
}
```

Ao executarmos a aplicação veremos que o resultado será uma espécie de Id (`Cliente@15db9742`), que possui o mesmo valor que a variável `paulo`, afinal, trata-se da referência para um mesmo objeto.