**Programação Orientada a Objetos - POO**

**Objetivo:** aproximar o mundo digital do mundo real

Programação baixo nível -> programação linear -> programação estruturada -> programação modular -> POO

**Classe conta =>** geral, com especificação de atributos: agencia, numero, saldo e titular

**Objeto conta =>** baseado na classe conta, onde eu posso alterar um atributo. Atribuições: agência: 0000, numero: 0000, saldo: 100.00 e titular: Julia Silva Prado

Obs: os atributos das classes são inicializados (zerados) automaticamente. Se algum atributo tiver que inicializar com outro número, deverá ser atribuído na classe principal.

**Conta.java – Classe**

**public** **class** Conta {

**double** saldo;

**int** agencia;

**int** numero;

String titular;

}

**CriaConta.java – Criação dos objetos**

**public** **class** CriaConta {

**public** **static** **void** main(String[] args) {

Conta primeiraConta = **new** Conta(); //criação do objeto primeiraConta baseado na classe Conta

primeiraConta.saldo = 200; //atribuindo o saldo do objeto primeiraConta

System.***out***.println("Primeira conta tem "+primeiraConta.saldo);

Conta segundaConta = **new** Conta(); //criação do objeto segundaConta

segundaConta.saldo = 50;

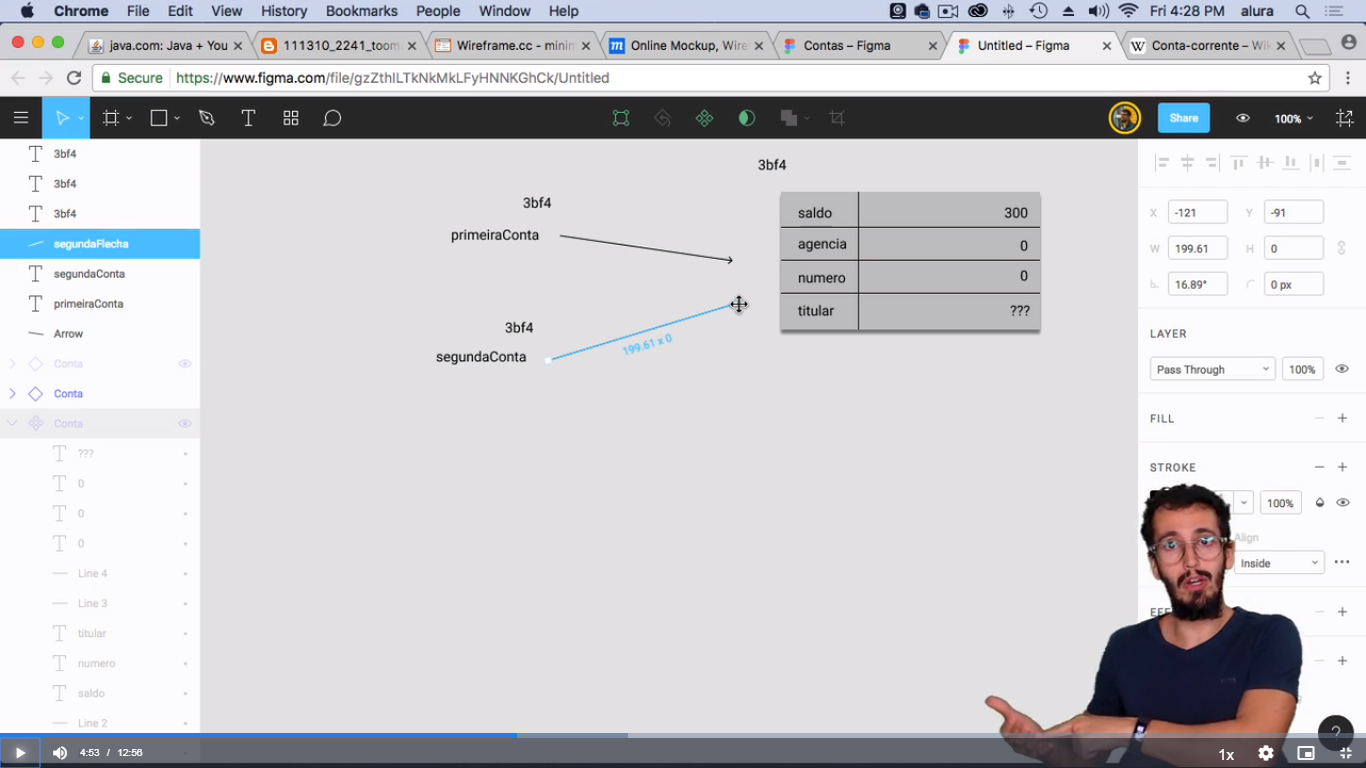
System.***out***.println("Segunda conta tem "+segundaConta.saldo);

}

}

**Referências**

“primeiraConta” e “segundaConta” apesar de parecerem, não são objetos, são referências para o objeto em si que tem seus dados inseridos. Observe:



As referências “primeiraConta” e “segundaConta” estão se **referindo** ao mesmo objeto que é na verdade a conta e recebe as informações.

**Métodos**

Um método em uma classe é literalmente o método de se fazer algo. Ele é composto por um bloco de código que você chama em outro momento e ele executa o que está dentro do bloco de chaves.

Um método pode retornar alguma coisa ou não, quando ele não precisa retornar nada, usamos a palavra reservada **void** como no exemplo a seguir:

**Conta.java - classe**

**public** **class** Conta {

**double** saldo;

**int** agencia;

**int** numero;

String titular;

**public** **void** deposita (**double** valor) {

**this**.saldo += valor;

}

}

**TesteMetodo.java**

**public** **class** TestaMetodo {

**public** **static** **void** main(String[] args) {

Conta conta = **new** Conta();

conta.saldo = 100;

conta.deposita(50);

System.***out***.println(conta.saldo);

}

}

**Herança**

Exemplo: todo mamífero é um animal, e todo cachorro é mamífero. Há uma hierarquia entre esses conceitos, que vai do mais geral (animal) ao mais específico (cachorro).

Por exemplo, poderíamos ter uma classe Animal que representasse animais em geral e as atividades que todos eles executam. Uma dessas atividades é comer, e nossa classe poderia ser como a abaixo.

**Animal.java - Classe**

**public** **class** Animal {

**public** **void** comer (String alimento) {

System.***out***.println("Estou comendo "+alimento);

}

}

Uma classe de mamíferos estenderia a classe de animais, porque mamíferos podem fazer coisas que outros animais não podem – por exemplo, gerar leite. Uma classe Mamifero estenderia a classe Animal usando a palavra reservada extends em sua declaração:

**Mamifero.java – Classe**

**public** **class** Mamifero **extends** Animal {

**public** String lactar() {

**return** "leite";

}

}

Como mamíferos são animais, também precisam do método comer(), e aí entra uma das vantagens da herança: como a classe Mamifero estende a classe Animal, ela herda todos os métodos de Animal, então não é preciso reescrevê-los.