DEPARTAMENTO DE COMPUTAÇÃO

UFRPE
Universidade
Federal Rural
de Pernambuco

Herança

DC – UFRPE

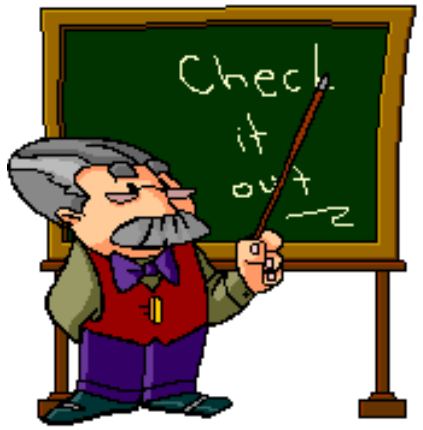
Programação II

Prof. Gustavo Callou

gustavo.callou@ufrpe.br

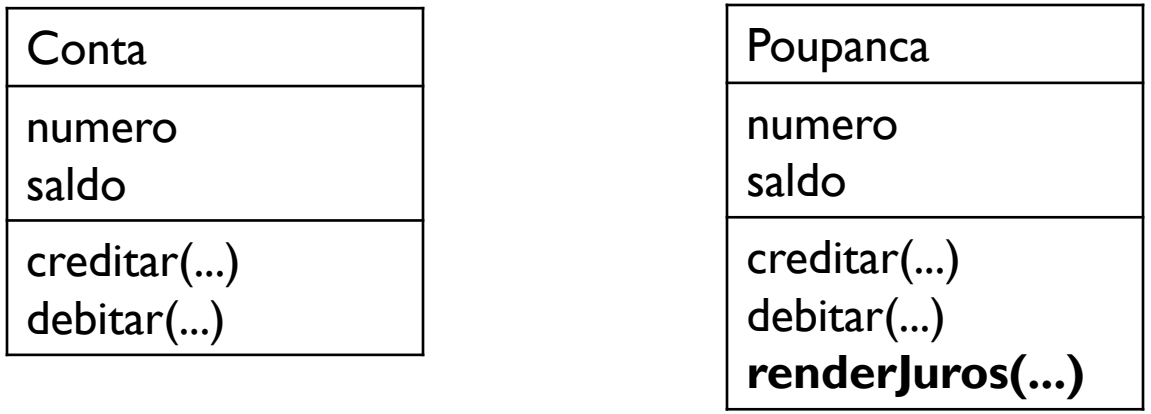
Roteiro

- Conceitos de herança em Java
- Modificador protected
- Classe Object



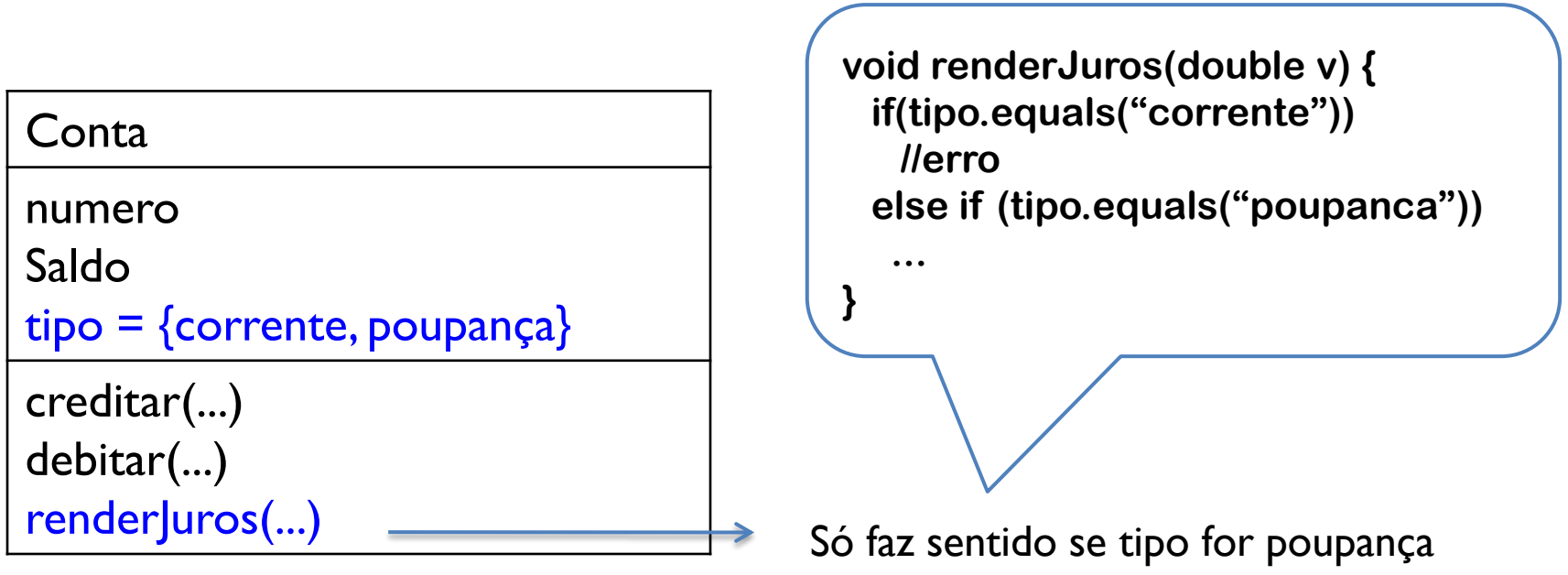
Como NÃO estender uma classse

- Exemplo: código de poupança é uma cópia do código de Conta com o acréscimo de novos atributos e métodos
  - Duplicação causa problemas de consistência e manutenção



Como NÃO estender uma classse

- Exemplo: usar uma mesma classe para representar dois conceitos.
  - Uso consistente depende do cliente da classe

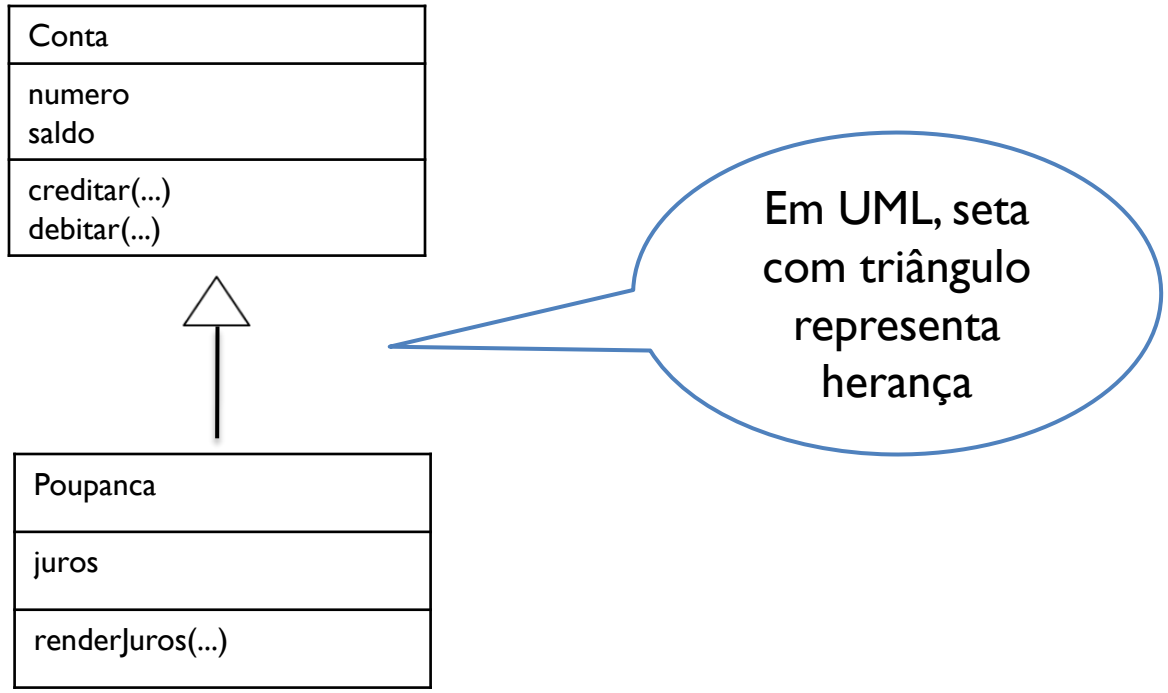


Herança

- A herança define um **hierarquia entre classes com atributos e métodos comuns**
- Superclasse: classe mais geral na hierarquia
- Subclasses: classes mais especializadas
- Permite:
  - Reuso
  - Separação de conceitos
  - Facilita manutenção do código

Relação de herança

- O que as superclasse tem/faz são automaticamente herdado para as subclasses
- Exemplo: Classe Poupanca herda (ou estende) a Classe Conta

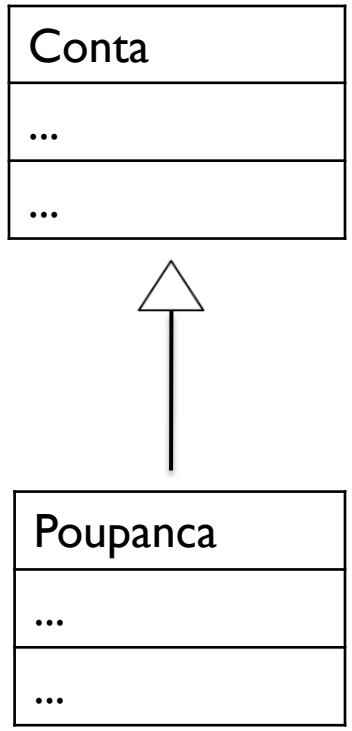


Herança vs Composição

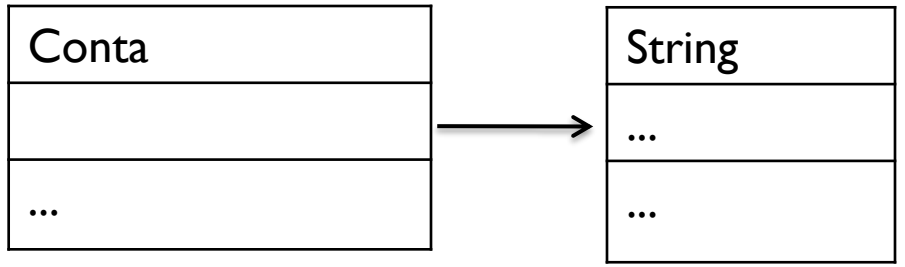
- Herança é uma relação → **é um tipo de**
  - Exemplos:
    - um cliente **é um tipo de** pessoa
    - uma carro **é um tipo de** veículo
    - Uma poupanca **é um tipo** de conta

- Composição é uma relação **tem um(a)**
  - Exemplo:
    - um carro **tem uma** placa
    - um endereço **tem um** cep

Herança vs Composição



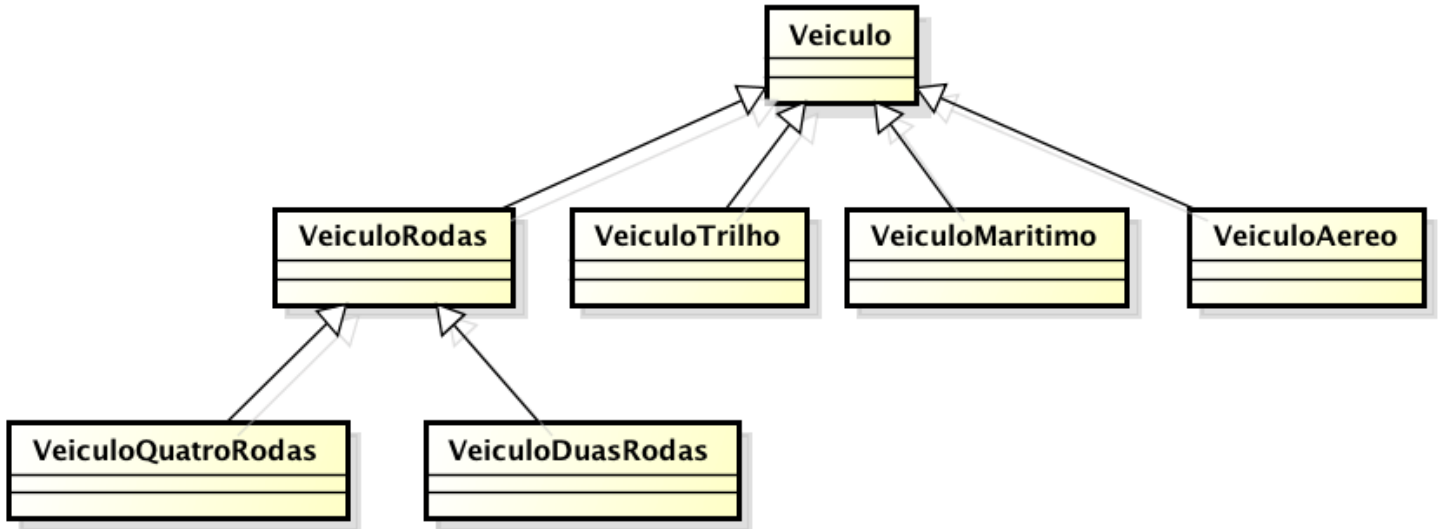
Um objeto poupança **é** um objeto conta



Uma conta **possui** um número

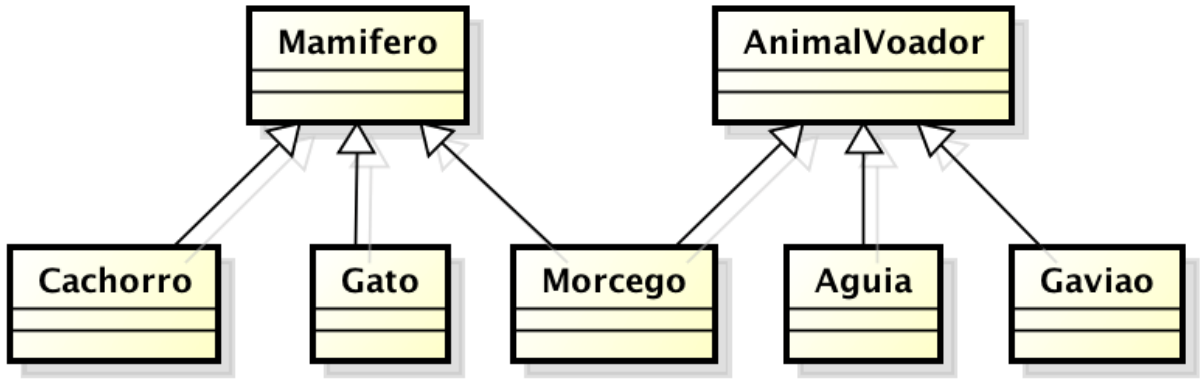
Árvore de Herança

- Relações de herança representadas através de árvores
- Nós pais (superclasses)
- Nós filhos (subclasses)



Herança simples vs múltipla

- Herança **simples**: classes herdam de um tipo
- Herança **múltipla**: classes herdam de vários tipos



Morcego é ao mesmo tempo um
mamífero e um animal voador (herança múltipla)

Herança em Java

- Apenas herança simples

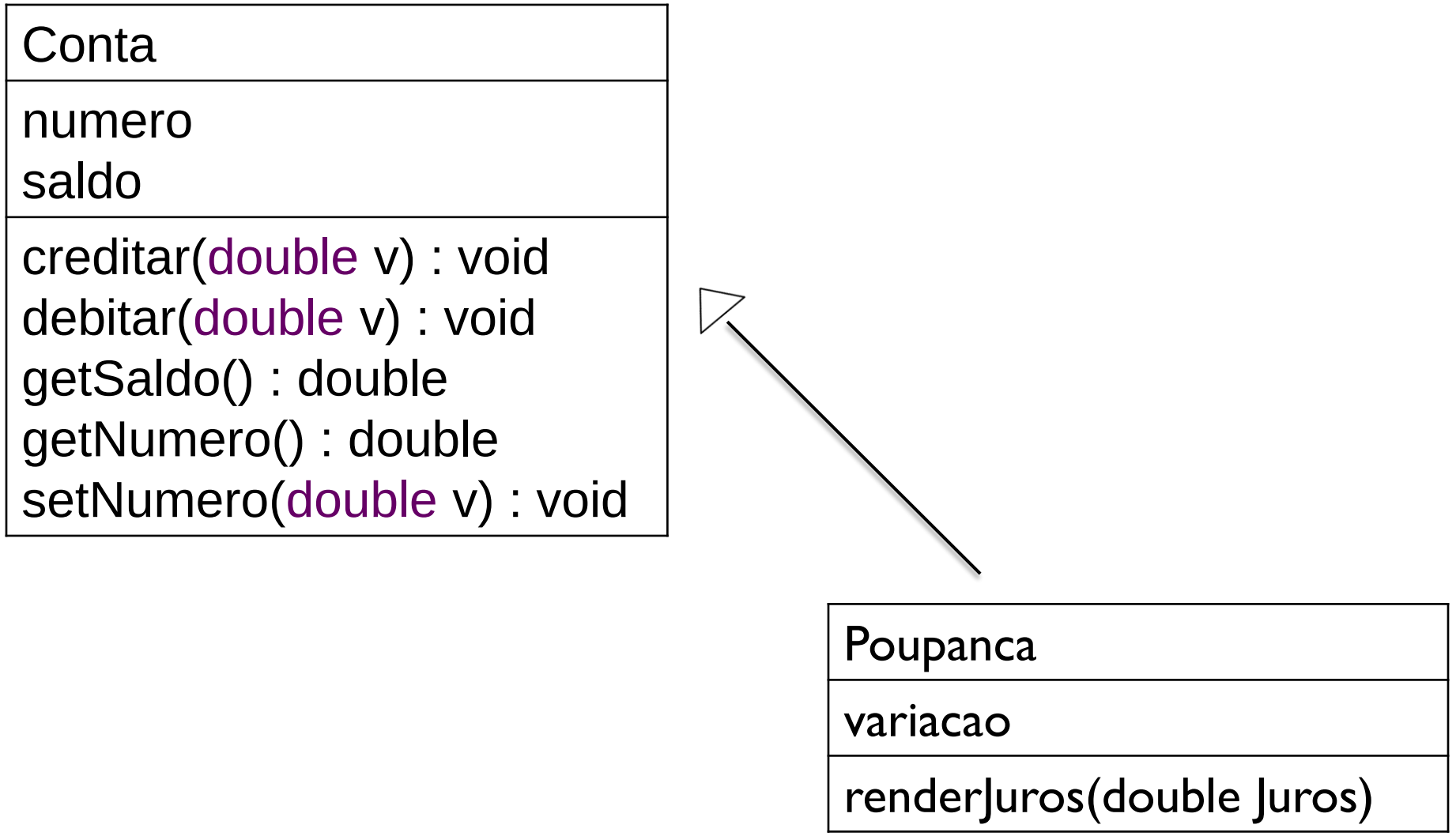
```
public class NomeSubclasse extends
  NomeSuperclasseDireta {

  ...

}
```

- Herança múltipla simulada: (1) usando interfaces (2) usando composição

Tipo de Contas



Classe Poupanca herda da Classe Conta

```
public class Poupanca extends Conta {
    public Poupanca(String n, double s) {
        ...
    }
    public void renderJuros(double juros){
                creditar(getSaldo()*juros);
    }
}
```

Classe Poupanca

```
Poupanca p = new Poupanca("1", 100);

p.debitar(10);
p.creditar(20);

double saldo = p.getSaldo();
p.renderJuros(0.01);
```

Atributos e métodos herdados

- O que é público na superclasse
  - é público na subclasse
  - Exemplo: métodos de Conta

- O que é privativo na superclasse
  - é escondido na subclasse
  - Exemplo: atributos de Conta

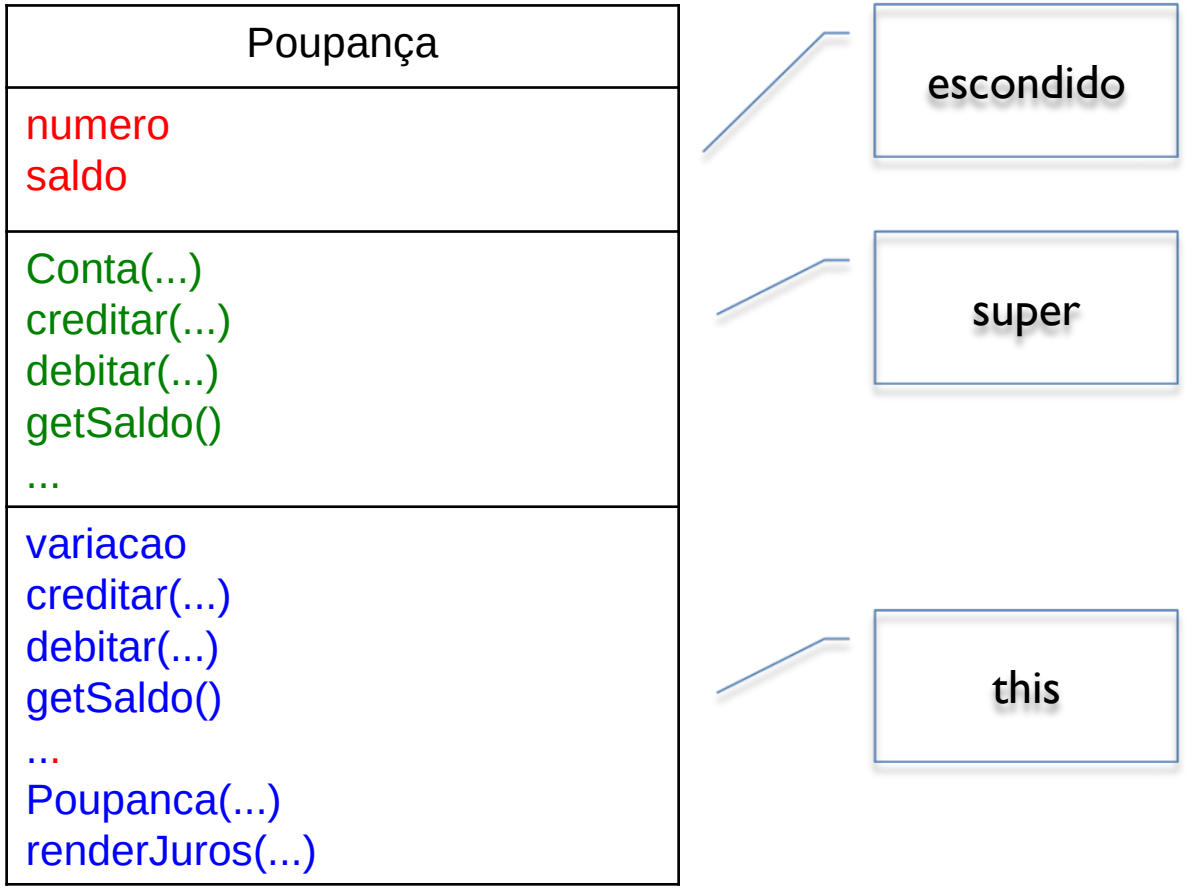
super vs this

- Toda classe possui duas referências:
  - this : referência para a classe
    - acessa elementos da própria classe
  - super : referência para a superclasse
    - Acessa elementos herdados da superclasse que são visíveis (ex: **public, protected**)

- Atributos/métodos escondidos : são herdados mas não podem ser acessados pela subclasse

super vs this

- Exemplo: parte escondida, super e this da classe Poupanca



Atributos e métodos herdados

```
public class Poupanca extends Conta {
        private int variacao;
        public Poupanca(String numero, double saldo, int variacao) {
                super.numero = numero;
                super.saldo = saldo;
            this.variacao = variacao;
        }

        public void renderJurosMensal(double jurosMensal){
                creditar(getSaldo()*jurosMensal);
        }
}
```

Erro de compilação

Ok!

Construtor da subclasse

- Deve chamar o construtor da superclasse
- Exemplo: Poupanca chama o construtor de Conta

```
public class Poupanca extends Conta {
    public Poupanca(String numero, double saldo) {
                super(numero, saldo);
     }
    ...
}
```

Atributos e métodos herdados

```
public class Poupanca extends Conta {

        public Poupanca(String numero, double saldo) {
                super(numero, saldo);
        }

        public void renderJurosMensal(double jurosMensal){
                this.creditar(this.getSaldo()*jurosMensal);
        }
}
```