Polimorfismo

DC – UFRPE

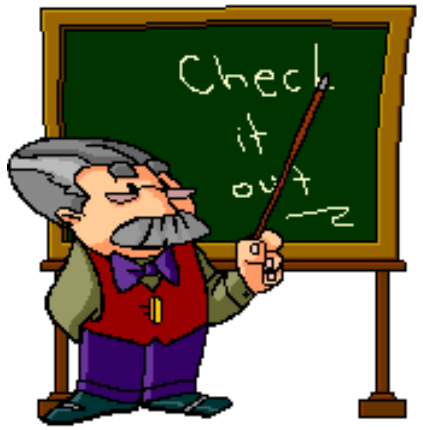
Programação II

Prof. Gustavo Callou

gustavo.callou@ufrpe.br

Roteiro

- Conhecer os mecanismos de coerção entre referências de Java
- Introduzir o conceito de polimorfismo

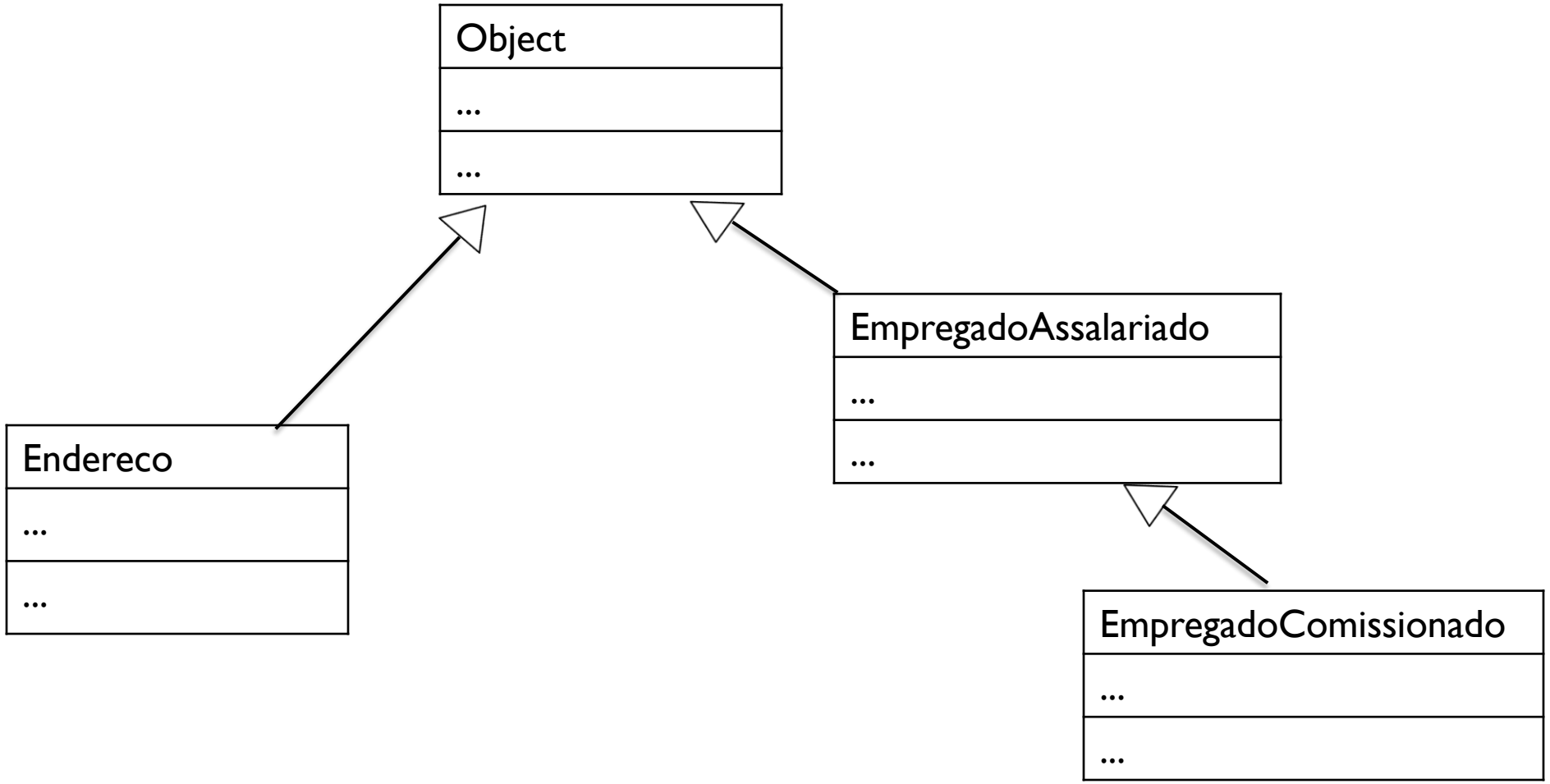


Coerção de referência (casting)

- Objetos não são convertidos!
- Tipo de um referência pode ser convertido
  - Desde que um tipo seja subtipo do outro (compatíveis)
- Upcasting: converção de referência de subtipo para supertipo
- Downcasting: inverso do upcasting

Coerção de referência

- Referências convertidas entre tipos compatíveis (subtipo ou supertipo)



Coerção de referência (casting)

```
Poupanca p1 = new Poupanca(...);

Conta c2 = p1;

Conta c3 = new Poupanca(...);
```

Coerção de referência (casting)

- Referência alvo da conversão define a interface que pode ser utilizada
- Ex: Conta c = new Poupanca(...)

  Referência c contém métodos e atributos de Conta,

  Não pode chamar c.renderJuros() !! Dá erro de compilação!

Coerção de referências

- Também ocorrem em passagem de parâmetro
- Exemplo: suponha o método transferir da Classe Conta

```
public void transferir(valor double,
                       Conta destino) {...}

conta.transferir(200.0, new Poupanca("1245-9", 10));
```

Ligação dinâmica

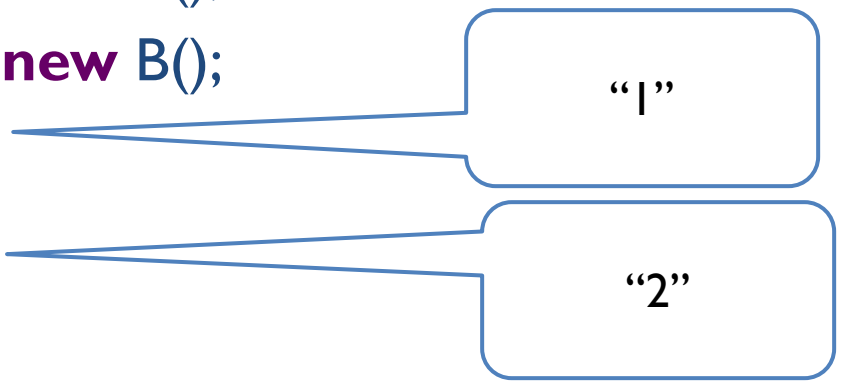
```
public class A {

public void m() {
     System.out.println("1");
  }
}
```

```
public class B extends A {
 public B(){
  super();
 }
 @Override
 public void m() {
    System.out.println("2");
 }
}
```

Ligação dinâmica

- Qual o resultado da chamada ao método m()?

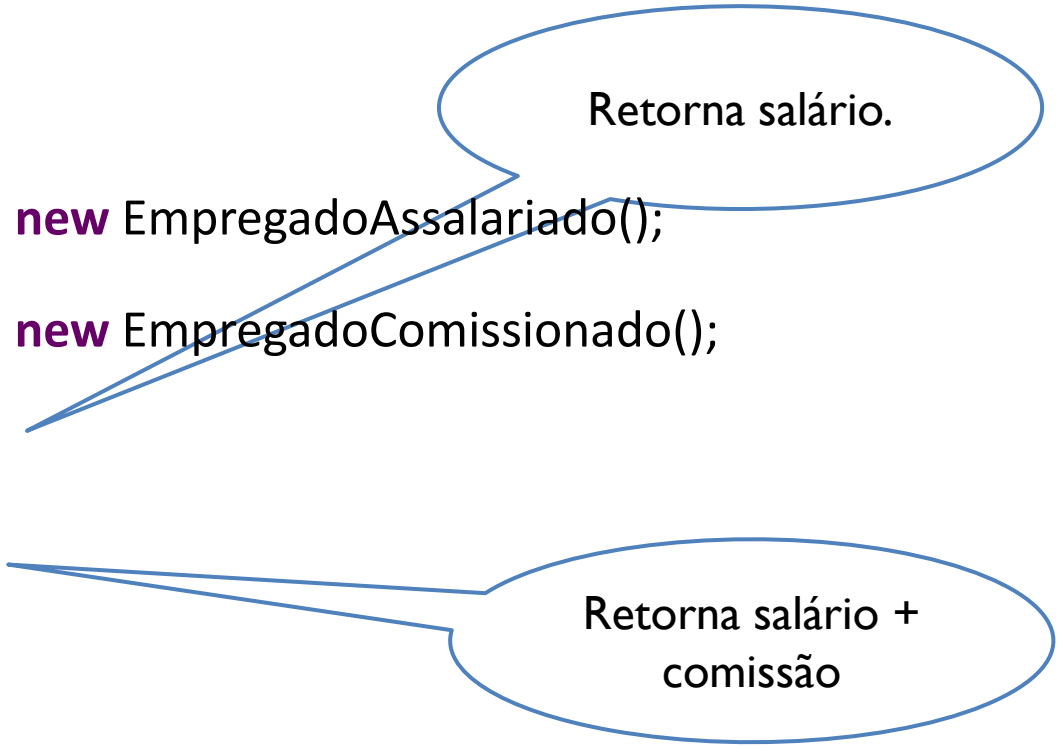
```
A ref1 = new A();
A ref2 = new B();
ref1.m();
ref2.m();
```



Ligação dinâmica

- A versão do método é escolhida em tempo de execução de acordo com o objeto apontado
- Exemplo:

```
EmpregadoAssalariado e1 = new EmpregadoAssalariado();

EmpregadoAssalariado e2 = new EmpregadoComissionado();

e1.getPagamento();

e2.getPagamento();
```



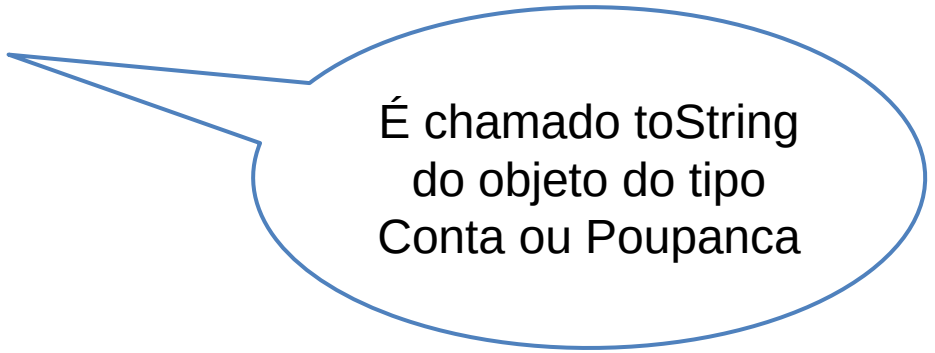
Polimorfismo

- Quando um método de um tipo é sobrescrito (*override*) nas subclasses este método passa a ter várias formas (ou implementações)

  Ex: toString tem versões diferentes em Conta e Poupanca

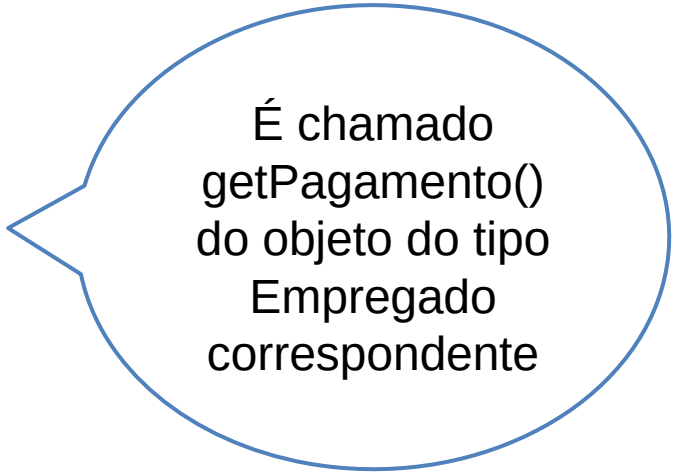
Polimorfismo

```
ArrayList<Conta> listaContas = new ArrayList<Conta>();
Conta c1 = new Conta(...);
Conta c2 = new Poupanca(...);
listaContas.add(c1);
listaContas.add(c2);
for(Conta conta : listaContas) {
  System.out.println(conta.toString());
}
```



Polimorfismo

```
ArrayList<EmpregadoAssalariado> listaEmpregados =
                                        new ArrayList ();

listaEmpregados.add(new EmpregadoAssalariado(...));
listaEmpregados.add(new EmpregadoComissionado(...));

for(Empregado emp : listaEmpregados) {
  System.out.println(emp.getNome());
  System.out.println(emp.getPagamento());
}
```



Coerção de referência

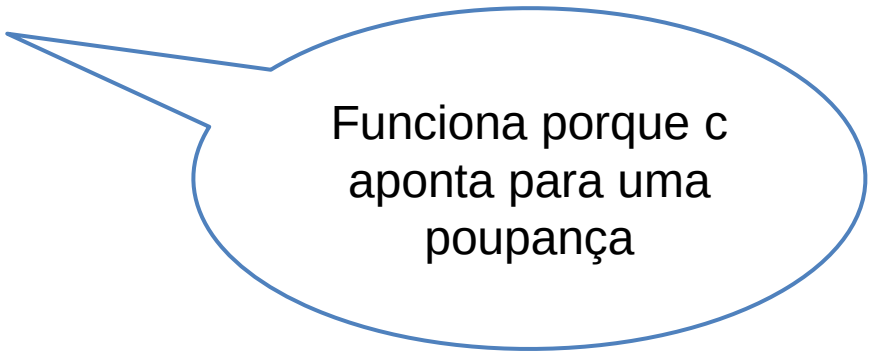
- Referência de supertipo não converte naturalmente para subtipo
  - subtipo pode ter mais atributos e métodos
- Exemplo:

```
Conta c1 = new Poupanca(...);
Poupanca p = c1;
```

Coerção de referência

- Conversão de supertipo para subtipo deve usar operador de casting
- Exemplo: convertendo referência do Tipo Conta para o Tipo Poupanca

```
Conta c = new Poupanca("1245-9");
Poupanca p = (Poupanca) c;
p.renderJuros();
```



ClassCastException

- Erro na tentativa de converter supertipo em subtipo
  - Supertipo não aponta para o tipo esperado
- Exemplo: converter conta em poupança

```
Conta c = new Conta("25552-X");
Poupanca p = (Poupanca) c;
p.renderJuros();
```

instanceof

- Operador de Java que testa o tipo do objeto apontado por uma referência
- Exemplos:

```
boolean isAssalariado(Object o) {
  return o instanceof EmpregadoComissionado;
}
```

```
boolean isPoupanca(Conta c) {
  return c instanceof Poupanca;
}
```

Sobrescrevendo equals

```
public class EmpregadoAssalariado {
    //atributos, construtor e métodos
    @Override
    public boolean equals(Object obj) {
            boolean igual = false;
            if (obj instanceof EmpregadoAssalariado) {
                    EmpregadoAsslariado e = (EmpregadoAssalariado) obj;
                    igual = e.cpf.equals(this.cpf);
            }
            return igual;
    }
}
```