

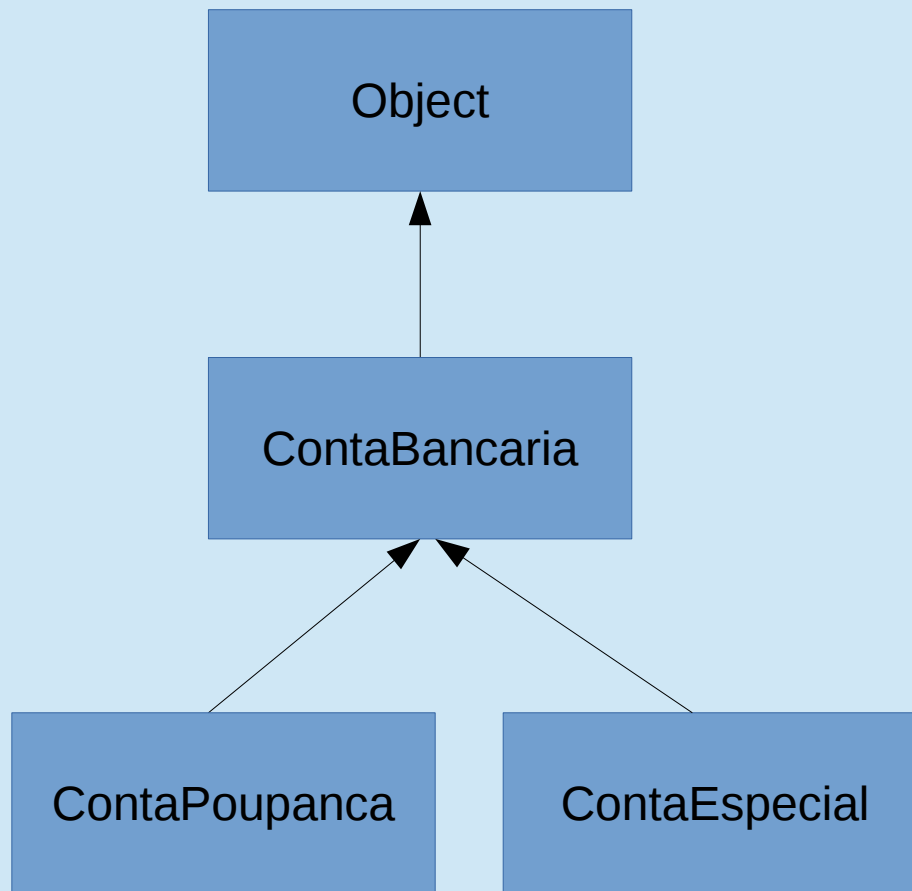
Polimorfismo

POO

Prof. Marcio Delamaro

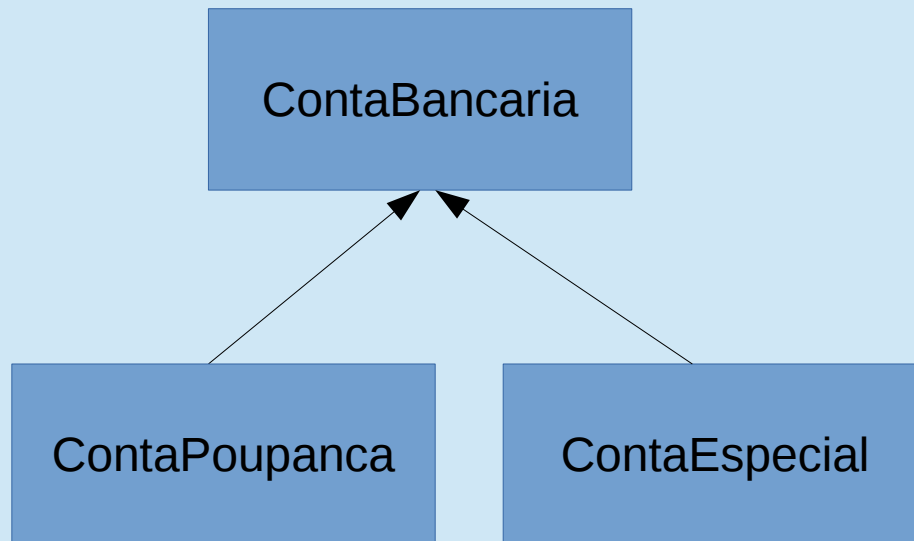
Recordando

- Sistema de contas bancárias



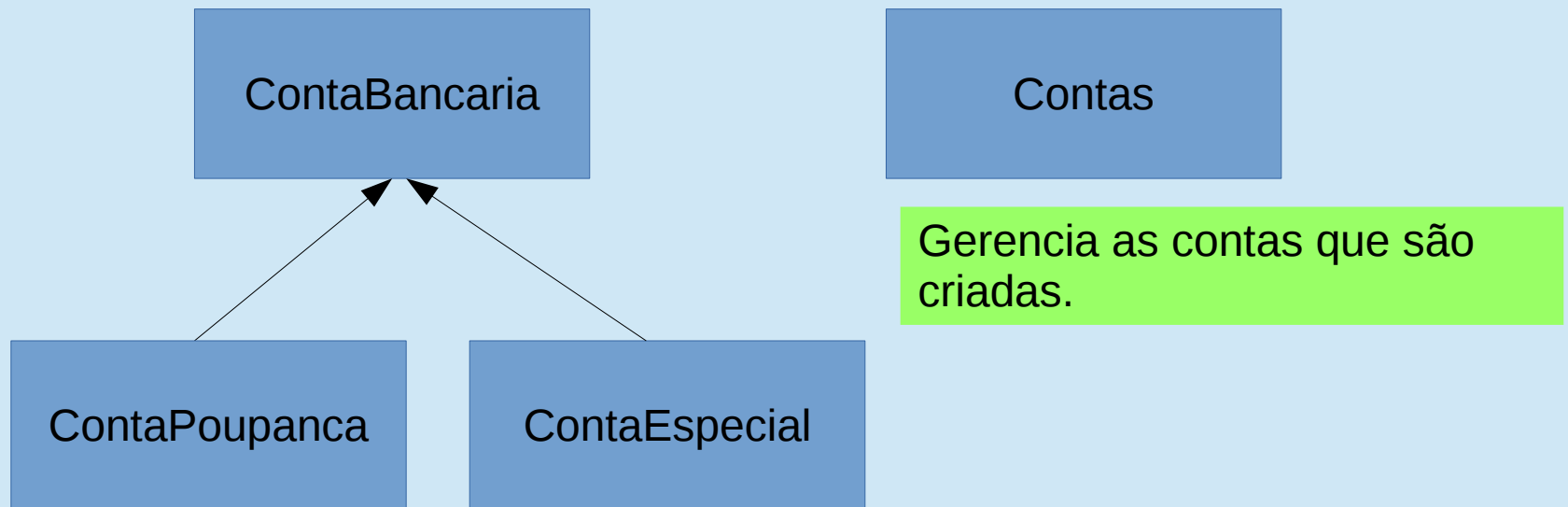
Recordando

- Sistema de contas bancárias



Recordando

- Sistema de contas bancárias



Polimorfismo

- po.li.mor.fis.mo
sm (poli1+morfo1+ismo) 1 Propriedade ou estado do que é polimorfo
- po.li.mor.fo
adj (poli1+morfo1) 1 Que se apresenta ou ocorre sob formas diversas. 2 Que assume ou passa por várias formas, fases etc. 3 Crist Que cristaliza em duas ou mais formas fundamentais.

Polimorfismo

- Na programação orientada a objetos, o polimorfismo permite que referências de tipos de classes mais abstratas representem o comportamento das classes concretas que referenciam. Assim, é possível tratar vários tipos de maneira homogênea (através da interface do tipo mais abstrato). O termo polimorfismo é originário do grego e significa "muitas formas" (poli = muitas, morphos = formas).
- [Wikipedia](#)

Instanciando

- ContaBancaria cb;
ContaPoupanca cp;
ContaEspecial ce;
- cp = new ContaPoupanca(...);
ce = new ContaEspecial(...);
cb = new ContaBancaria(...);
- cp.saca(); ce.saca(); cb.saca(); cp.atualiza();

Instanciando

- ContaBancaria cp = new ContaPoupanca(...);
 - Posso fazer isso???

Instanciando

- `ContaBancaria cp = new ContaPoupanca(...);`
 - Posso fazer isso???
- Posso. Isso porque um objeto pode ser tratado de diversas formas.
- `cp` é uma variável do tipo `ContaBancaria`
- Mas o objeto que ela “contém” é do tipo `ContaPoupanca`

Instanciando

- `ContaBancaria cb = new ContaPoupanca (...);`
 - Posso fazer isso???
- Posso. Isso porque um objeto pode ser tratado de diversas formas.
- `cb` é uma variável do tipo *ContaBancaria*
- Mas o objeto que ela “contém” é do tipo *ContaPoupanca*
- `cp.atualiza(0.05);` ?????

Como fazer?

- Se eu tenho um objeto do tipo ContaPoupanca, como fazer para chamar o método *atualiza*?

Como fazer?

- Se eu tenho um objeto do tipo `ContaPoupanca`, como fazer para chamar o método *atualiza*?
- ```
ContaBancaria cb = new ContaPoupanca(...);
ContaPoupanca cp = (ContaPoupanca) cb;
cp.atualiza(0.005);
```
- Nesse caso, o objeto não está sendo modificado.
- Apenas estou notificando o compilador que o objeto em *cb* é do tipo *ContaPoupanca*.

# Classe contas

```
public class Contas {
 private ContaBancaria contas[] = new ContaBancaria[100];
 private int nContas = 0;
```

# Classe contas

```
public class Contas {
 private ContaBancaria contas[] = new ContaBancaria[100];
 private int nContas = 0;

 public static void main(String[] args) throws Exception {
 int op = 0;
 Contas ct = new Contas();
 while (op != 7) {
 op = leOpcao();
 switch (op)
 {
 case 1: ...
 ...
 }
 }
 }
}
```

# Classe contas – main

- 1) Criar poupança
- 2) Criar conta especial
- 3) Realizar saque
- 4) Realizar deposito
- 5) Atualizar poupanças
- 6) Mostrar saldos
- 7) Sair

Digite a opção desejada ==>

# Opção 1 – Criar poupança

case 1:

```
System.out.println("Numero da conta: ");
int conta = EntradaTeclado.leInt();
System.out.println("Nome do correntista: ");
String s = EntradaTeclado.leString();
System.out.println("Dia de vencimento: ");
int dia = EntradaTeclado.leInt();
ContaPoupanca cp = new ContaPoupanca(s, conta, dia);
ct.add(cp);
System.out.println("***** Conta criada.*****");
break;
```



# Adicionar uma conta poupança

```
private void add(ContaPoupanca c) {
 contas[nContas++] = c;
}
```

# Opção 2 – Criar conta especial

- Façam, por favor.

# Opção 2 – Criar conta especial

case 2:

```
System.out.println("Numero da conta: ");
conta = EntradaTeclado.leInt();
System.out.println("Nome do correntista: ");
s = EntradaTeclado.leString();
System.out.println("Limite de saque: ");
double limite = EntradaTeclado.leDouble();
ContaEspecial ce = new ContaEspecial(s, conta, limite);
ct.add(ce);
System.out.println("***** Conta criada.*****");
break;
```

# Adicionar uma conta especial

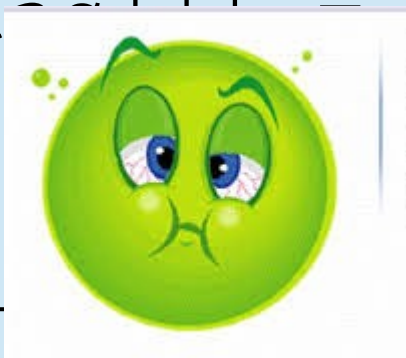
```
private void add(ContaPoupanca c) {
 contas[nContas++] = c;
}

private void add(ContaEspecial c) {
 contas[nContas++] = c;
}
```

# Adicionar uma conta especial

```
private void add(ContaPoupanca c) {
 contas[nContas++] = c;
}

private void add(ContaEspecial c) {
 contas[nContas++] = c;
}
```



# Adicionar uma conta qualquer

```
private void add(ContaBancaria c) {
 contas[nContas++] = c;
}
```

# Depositar

```
case 4:
System.out.println("Numero da conta: ");
conta = EntradaTeclado.leInt();
System.out.println("Valor a depositar: ");
valor = EntradaTeclado.leDouble();
cb = ct.procura(conta);
if (cb == null)
{
 System.out.println("***** Conta não existe *****");
 break;
}
cb.deposita(valor);
System.out.println("***** Depósito realizado *****");
break;
```

# Depositar

case 4:

```
System.out.println("Numero da conta: ");
```

```
conta = EntradaTeclado.leInt();
```

```
System.out.println("Valor a depositar: ");
```

```
valor = EntradaTeclado.leDouble();
```

```
cb = ct.procura(conta);
```

Devolve uma conta bancária genérica.

```
if (cb == null)
```

```
{
```

```
 System.out.println("***** Conta não existe *****");
```

```
 break;
```

```
}
```

```
cb.deposita(valor);
```

```
System.out.println("***** Depósito realizado *****");
```

```
break;
```



# Sacar

Case 3: // lê os dados....

```
ContaBancaria cb = ct.procura(conta);
if (cb == null) ...
try {
 cb.saca(valor);
 System.out.println("***** Saque realizado
*****");
}
catch (Exception e) {
 System.out.println("***** Saque não
realizado *****");
 System.out.println(e.getMessage());
}
break;
```

# Sacar

- No caso do saque, o objeto utilizado é um `CntaBancaria`
- Mas não existe um objeto desse tipo
- Assim, o tipo real do objeto é `ContaPoupanca` ou `ContaEspecial`
- Assim, o método realmente chamado depende do tipo real do objeto retornado

# Mostrar os saldos

```
case 6:
 ct.printSaldos();
 break;

private void printSaldos() {
 for (ContaBancaria ctb : contas){
 if (ctb == null) break;
 System.out.println("Numero da conta:" +
 ctb.getNumConta());
 System.out.println("Titular: " + ctb.getNomeCliente());
 System.out.println("Saldo: " + ctb.getSaldo());
 System.out.println();
 }
}
```

# Atualizar poupança

case 5:

```
System.out.println("Qual o valor da taxa? ");
double tx = EntradaTeclado.leDouble();
ct.atualizaPoupança(tx);
System.out.println("Saldos atualizados");
break;
```

# Atualizar poupança

- Percorrer todas as contas
- Se ela for poupança, devemos fazer a atualização
  - `cp.atualiza(tx);`
- Como saber se podemos ou não chamar o método?

# Atualizar

```
private void atualizaPoupança(double tx) {
 for (ContaBancaria ctb : contas) {
 if (ctb == null) break;
 if (ctb instanceof ContaPoupanca) {
 ContaPoupanca cp = (ContaPoupanca) ctb;
 cp.atualiza(tx);
 }
 }
}
```

# Atualizar

```
private void atualizaPoupança(double tx) {
 for (ContaBancaria ctb : contas) {
 if (ctb == null) break;
 if (ctb instanceof ContaPoupanca) {
 ContaPoupanca cp = (ContaPoupanca) ctb;
 cp.atualiza(tx);
 }
 }
}
```

O trecho marcado só pode ser executado quando temos certeza que o objeto pode ser atualizado, ou seja, quando é do tipo ContaPoupanca.

# Atualizar

```
private void atualizaPoupança(double tx) {
 for (ContaBancaria ctb : contas) {
 if (ctb == null) break;
 if (ctb instanceof ContaPoupanca) {
 ContaPoupanca cp = (ContaPoupanca) ctb;
 cp.atualiza(tx);
 }
 }
}
```

O trecho marcado só pode ser executado quando temos certeza que o objeto pode ser atualizado, ou seja, quando é do tipo ContaPoupanca.