FeMASS
FACULDADE PROF. MIGUEL ÂNGELO DA SILVA SANTOS

Projeto de Sistemas I

Faculdade Prof. Miguel Ângelo da Silva Santos

Material 6 – Orientação a Objetos em Java (Polimorfismo)

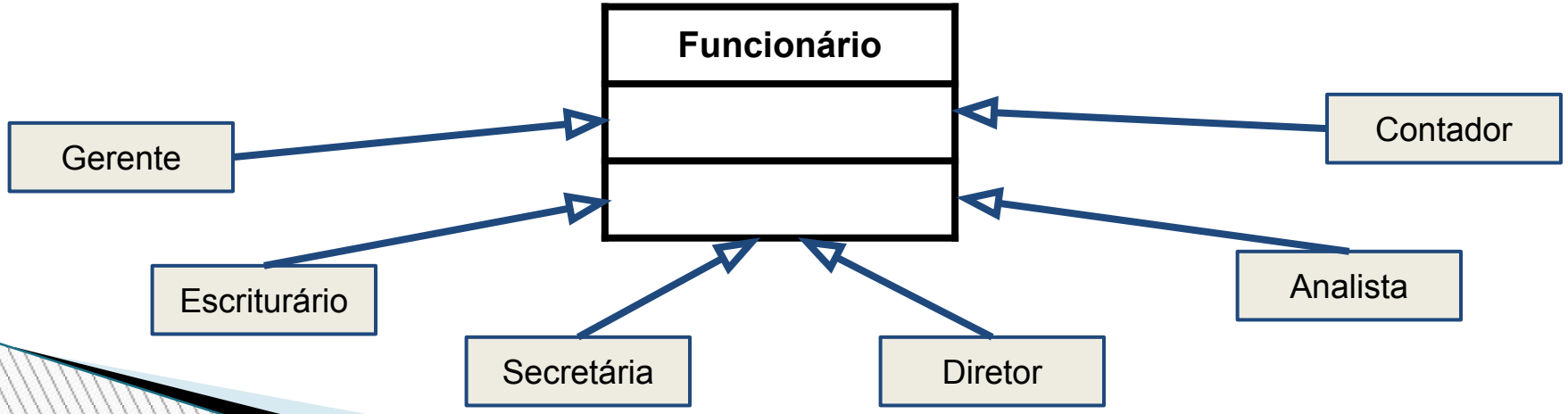
Professor: Isac Mendes Lacerda, M.Sc., PMP, CSM
e-mail: isac.curso@gmail.com

Tópicos

- Polimorfismo

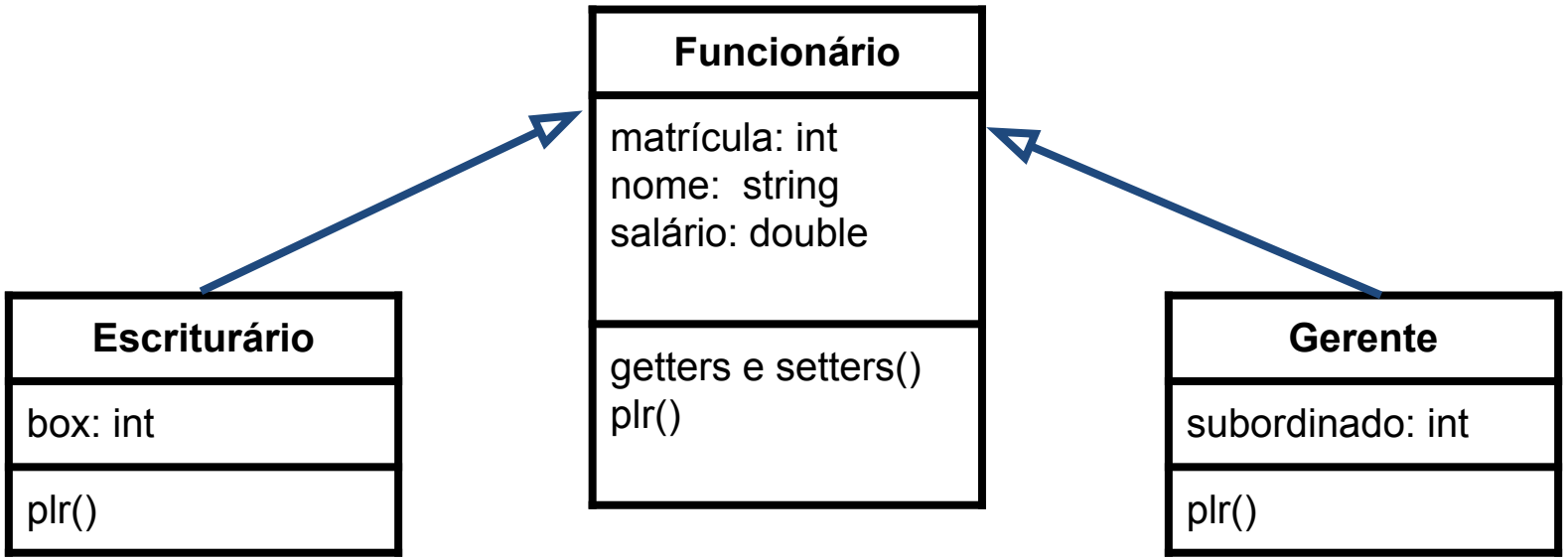
Polimorfismo

- Como você já sabe polimorfismo é uma maneira de variar o comportamento, especialmente entre objetos que utilizam hierarquia de classes.



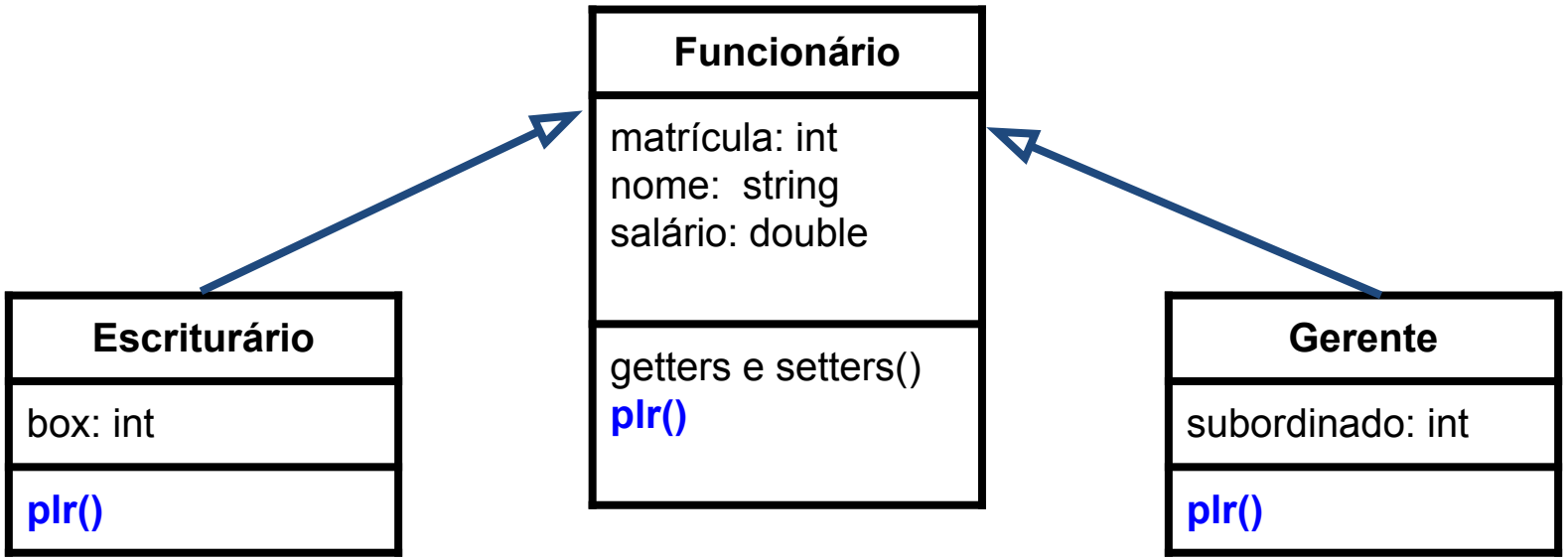
Polimorfismo

- Em Java, sobrescrever métodos dá suporte ao polimorfismo.



Polimorfismo

- Em Java, sobrescrever métodos dá suporte ao polimorfismo.



Polimorfismo

```java
public class Funcionario {
    int matricula;
    String nome;
    double salario;

    public Funcionario(int m, String n, double s){
        this.matricula = m;
        this.nome = n;
        this.salario = s;
    }
    public double calcular_plr(){
        return(this.salario * 0.05);
    }
}
```

```java
public class Gerente extends Funcionario{
    int subordinado;

    public Gerente(int m, String n, double s, int sub){
        super(m, n, s);
        this.subordinado = sub;
    }
    @Override
    public double calcular_plr(){
        return(this.salario * 0.25);
    }
}
```

```java
public class Escriturario extends Funcionario{
    int box;

    public Escriturario(int m, String n, double s, int box){
        super(m, n, s);
        this.box = box;
    }
    @Override
    public double calcular_plr(){
        return(this.salario * 0.15);
    }
}
```

Polimorfismo

```
13  public class Funcionario {
14      int matricula;
15      String nome;
16      double salario;
17
18      public Funcionario(int m, String n, double s){
19          this.matricula = m;
20          this.nome = n;
21          this.salario = s;
22      }
23      public double calcular_plr(){
24          return(this.salario * 0.05);
25      }
26  }
```

```
12  public class Gerente extends Funcionario{
13      int subordinado;
14
15      public Gerente(int m, String n, double s, int sub){
16          super(m, n, s);
17          this.subordinado = sub;
18      }
19      @Override
20      public double calcular_plr(){
21          return(this.salario * 0.25);
22      }
23  }
```

```
12  public class Escriturario extends Funcionario{
13      int box;
14
15      public Escriturario(int m, String n, double s, int box){
16          super(m, n, s);
17          this.box = box;
18      }
19      @Override
20      public double calcular_plr(){
21          return(this.salario * 0.15);
22      }
23  }
```

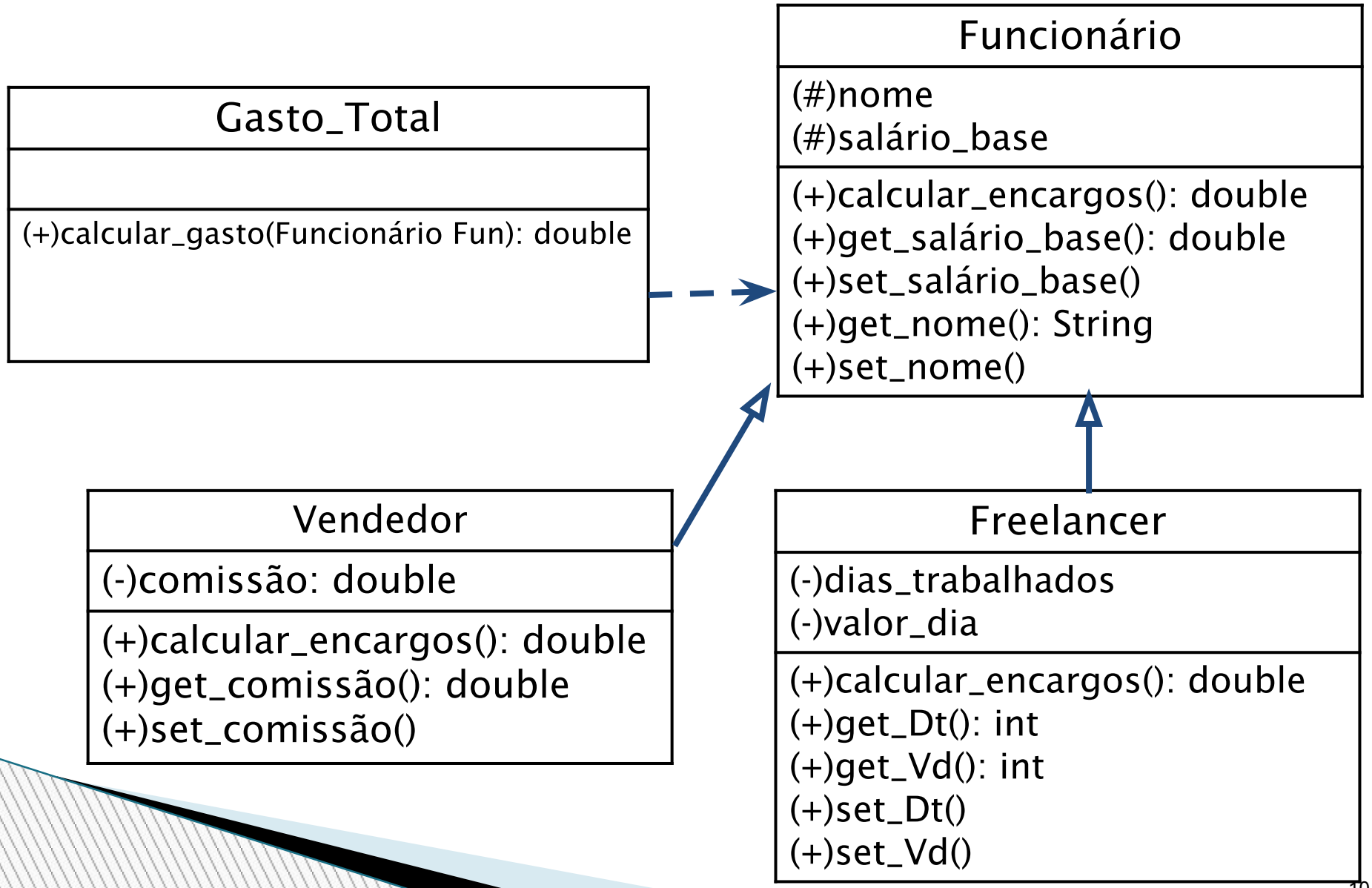
Demonstração complementar Netbeans (está no vídeo)...

Exercício 1

Desconsiderando a notação de encapsulamento, implemente o modelo do próximo slide e crie um objeto de cada classe, testando todos os seus respectivos métodos.
Observações:

- Os métodos *calcular_encargos()*, contidos nas classes de funcionários devem ter diferenças de lógica (seja criativo), para justificar a sobrescrição.
- Na classe *Gasto_Total*, o método *calcular_gasto()* deve receber como argumento qualquer tipo de funcionário e contabilizar todo o valor gasto com esse funcionário (salário e encargos). A relação entre a classe *Gasto_Total* e *Funcionário* é do tipo dependência.

Exercício 1

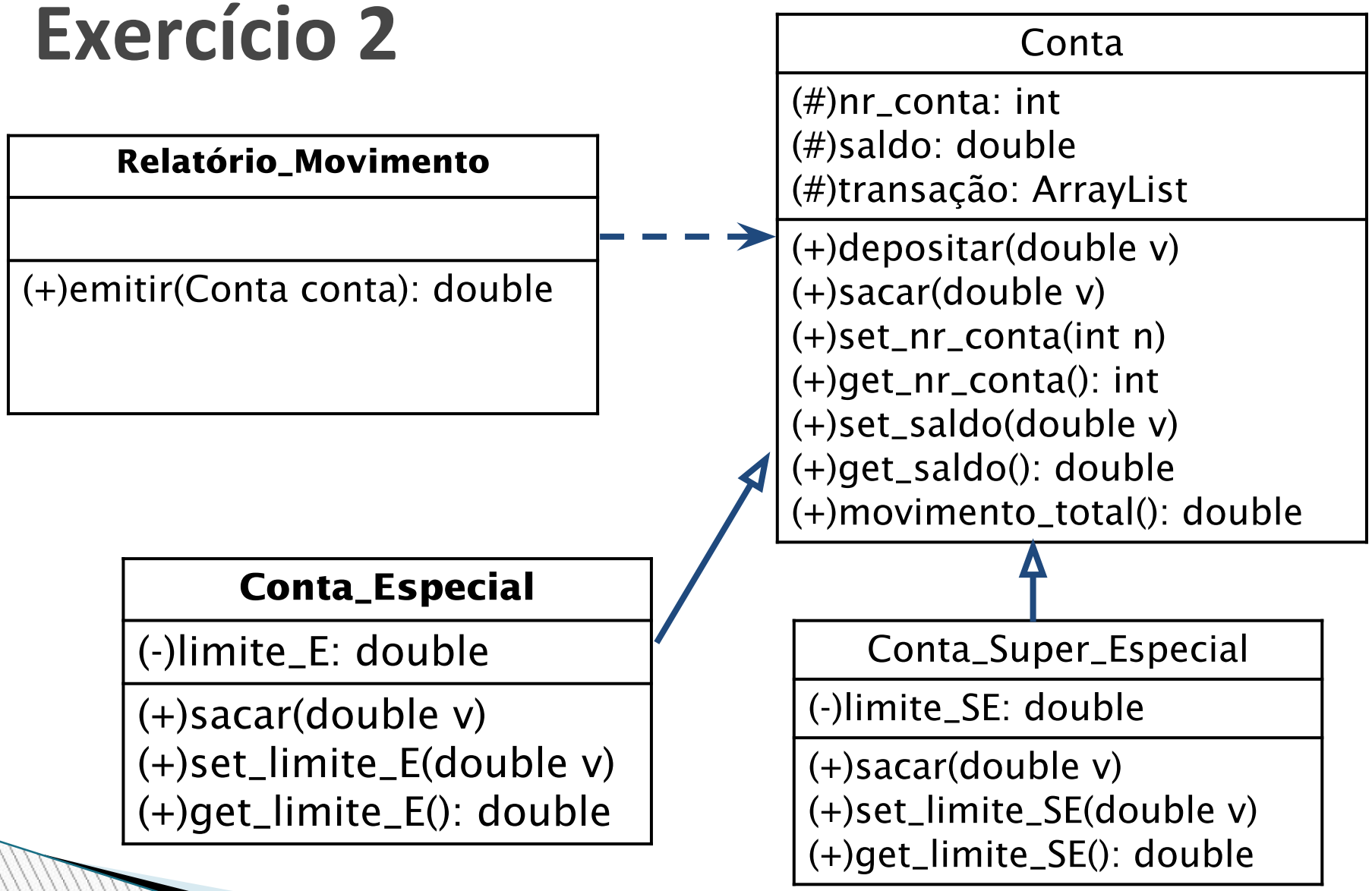

Funcionário
(#)nome
(#)salário_base
(+)calcular_encargos(): double
(+)get_salário_base(): double
(+)set_salário_base()
(+)get_nome(): String
(+)set_nome()
Gasto_Total
(+)calcular_gasto(Funcionário Fun): double
Vendedor
(-)comissão: double
(+)calcular_encargos(): double
(+)get_comissão(): double
(+)set_comissão()
Freelancer
(-)dias_trabalhados
(-)valor_dia
(+)calcular_encargos(): double
(+)get_Dt(): int
(+)get_Vd(): int
(+)set_Dt()
(+)set_Vd()

Exercício 2

Desconsiderando a notação de encapsulamento, implemente o modelo do próximo slide e crie um objeto de cada classe, testando todos os seus respectivos métodos.
Observações:

- Na classe *Conta_Genérica*, o atributo *Transação* deve armazenar cada valor movimentado (de saque ou depósito, de modo a manter o histórico).
- Na classe *Relatório_Movimento*, o método *emitir()* deve receber como argumento qualquer tipo de conta e contabilizar todo o valor da sua movimentação. A relação entre a classe *Gasto_Total* e *Funcionário* é do tipo dependência.

Exercício 2


Relatório_Movimento
(+)emitir(Conta conta): double
Conta
(#)nr_conta: int
(#)saldo: double
(#)transação: ArrayList
(+)depositar(double v)
(+)sacar(double v)
(+)set_nr_conta(int n)
(+)get_nr_conta(): int
(+)set_saldo(double v)
(+)get_saldo(): double
(+)movimento_total(): double
Conta_Especial
(-)limite_E: double
(+)sacar(double v)
(+)set_limite_E(double v)
(+)get_limite_E(): double
Conta_Super_Especial
(-)limite_SE: double
(+)sacar(double v)
(+)set_limite_SE(double v)
(+)get_limite_SE(): double