Orientação a Objetos

Polimorfismo e Interfaces

EDUCAÇÃO METODISTA

Polimorfismo

Mesmo método, comportamento diferente

Introdução ao Polimorfismo

- Polimorfismo é o conceito que descreve a capacidade de um tipo A ser usado como um tipo B.

- Polimorfismo é o nome dado quando uma operação de uma superclasse é implementada de maneira diferente por duas ou mais subclasses.
  - Essas implementações diferentes fazem com que um mesmo método chamado de um objeto do tipo da superclasse tenha comportamentos diferentes

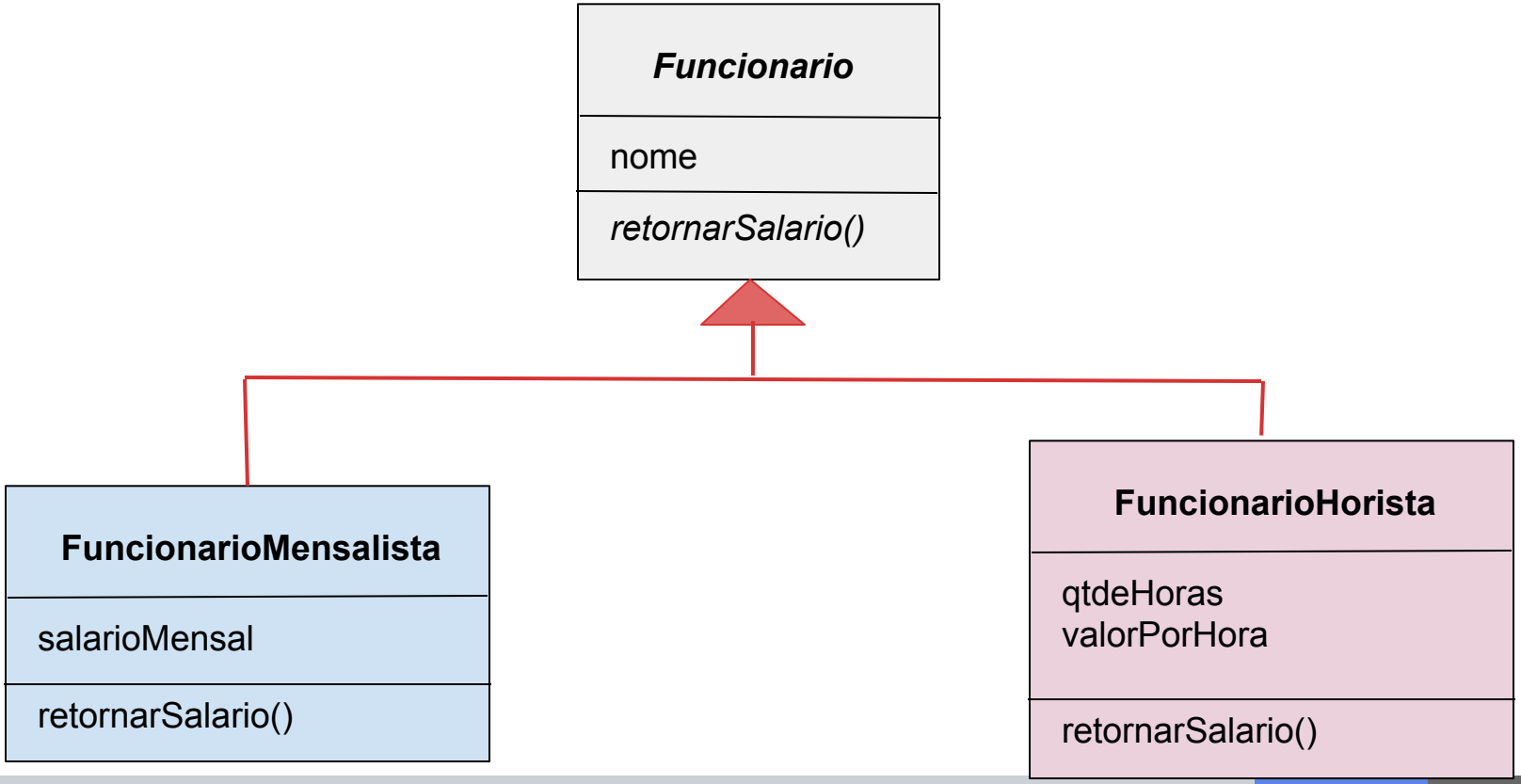
Introdução ao Polimorfismo

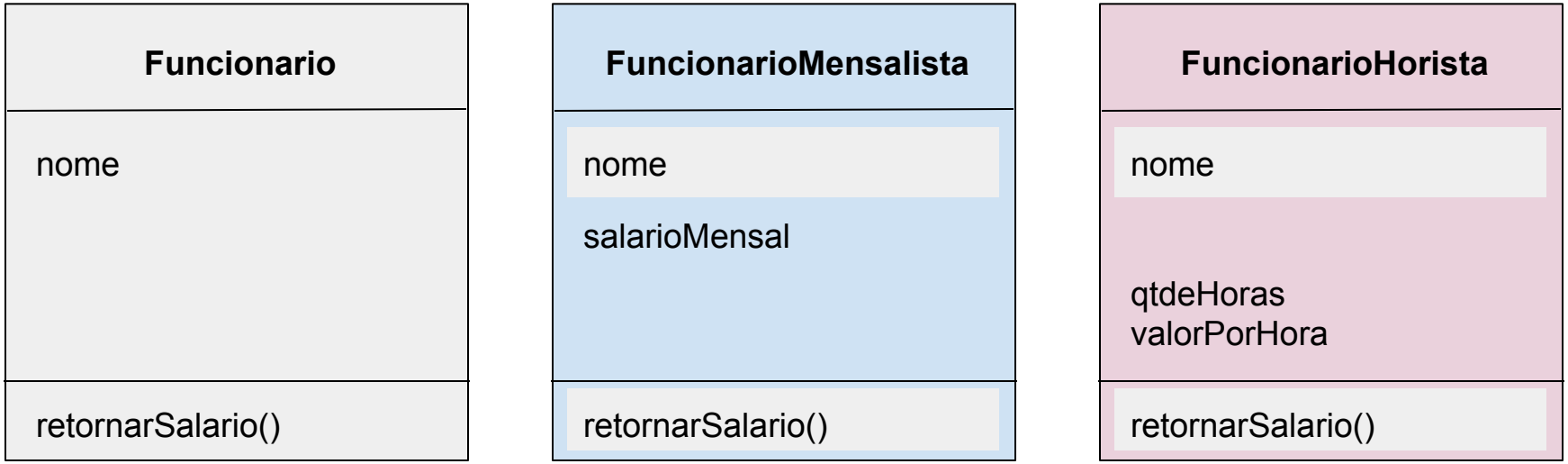
- O objetivo do polimorfismo é implementar um estilo de programação baseado em passagem de mensagens no qual objetos de diferentes tipos definem uma mesma interface de operações.

Classes

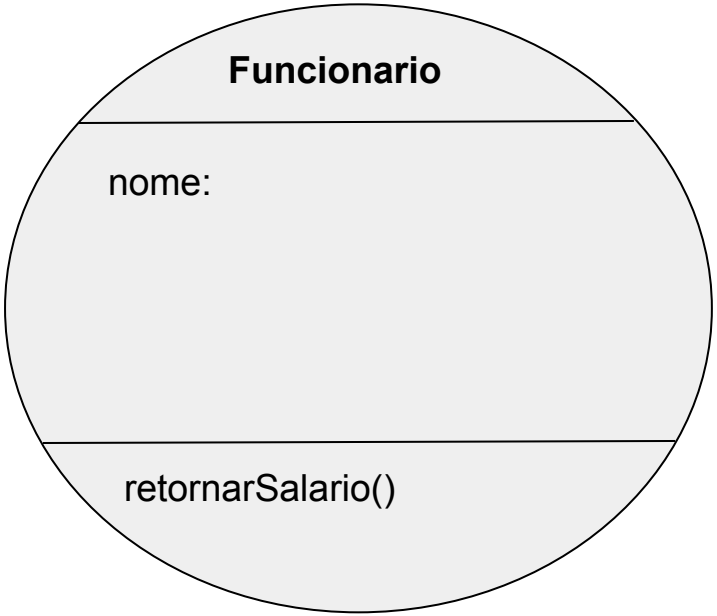
- Funcionario
  - String nome
  - abstract void retornarSalario()
- FuncionarioMensalista : Funcionario
  - double salarioMensal
  - void retornarSalario()
- FuncionarioHorista : Funcionario
  - double qtdeHoras
  - double valorPorHora
  - void retornarSalario()

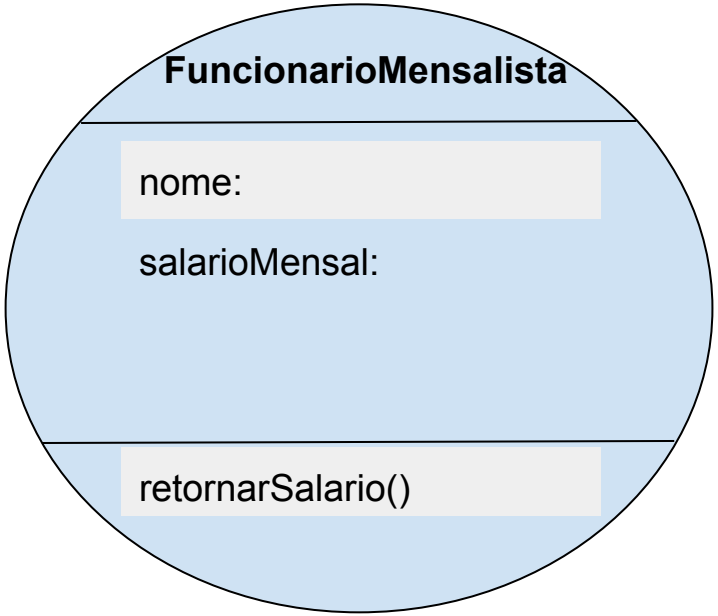
Classes


Funcionario
nome
retornarSalario()
FuncionarioMensalista
salarioMensal
retornarSalario()
FuncionarioHorista
qtdeHoras
valorPorHora
retornarSalario()

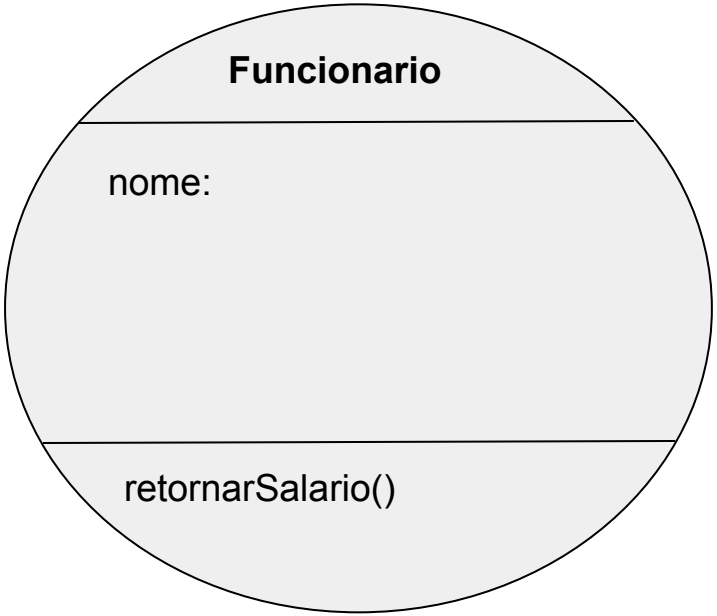

Funcionario
nome
retornarSalario()
FuncionarioMensalista
nome
salarioMensal
retornarSalario()
FuncionarioHorista
nome
qtdeHoras
valorPorHora
retornarSalario()

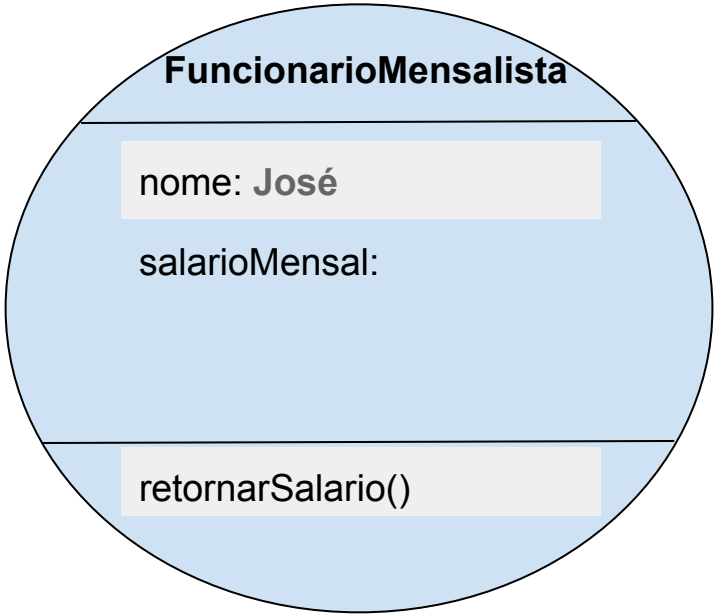
```
Funcionario func = new FuncionarioMensalista ();
```



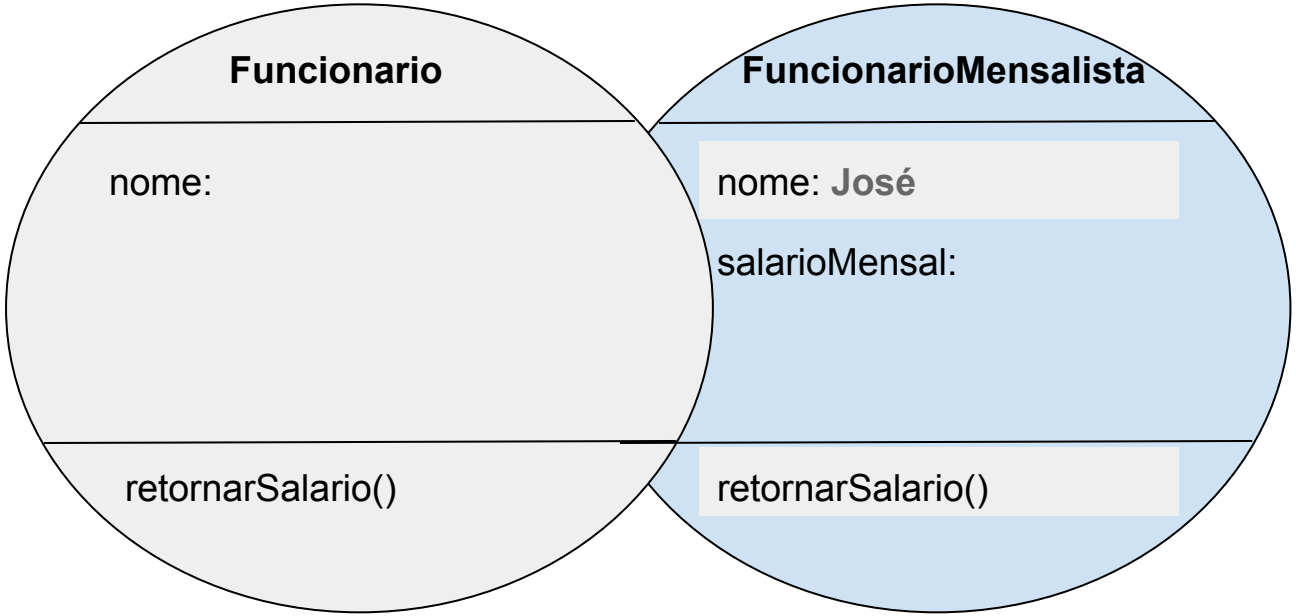


```
Funcionario func = new FuncionarioMensalista ();
func.setNome("José");
```

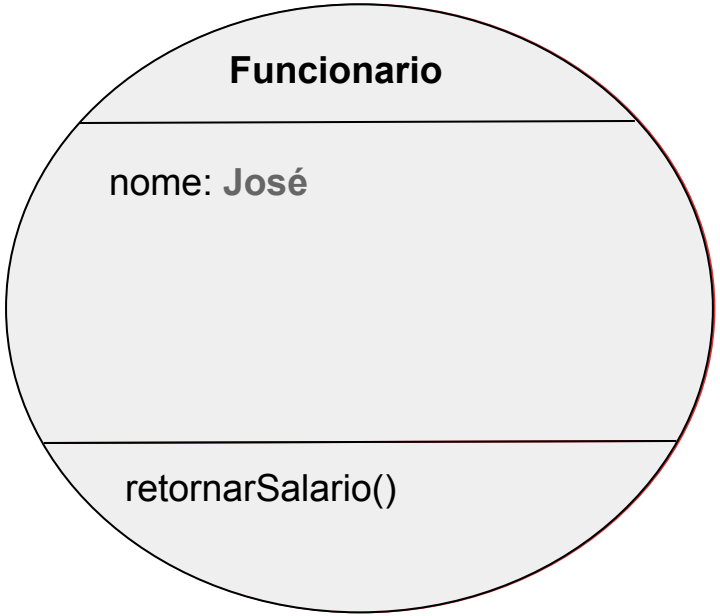




```
Funcionario func = new FuncionarioMensalista ();
func.setNome("José");
```



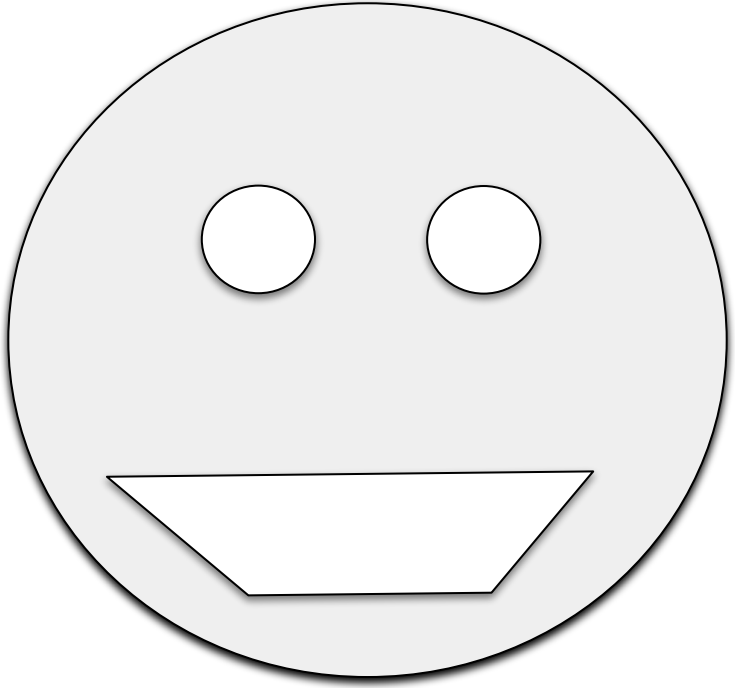
```
Funcionario func = new FuncionarioMensalista ();
func.setNome("José");
```

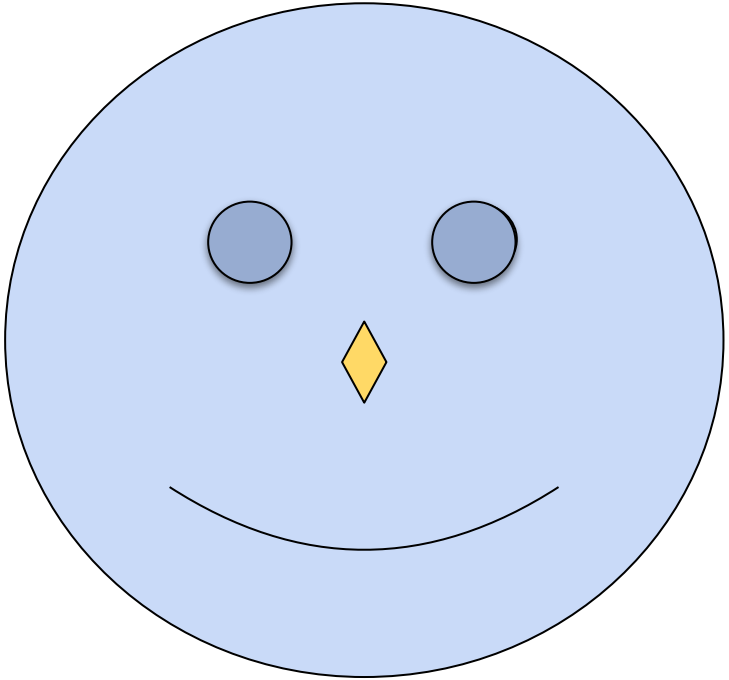


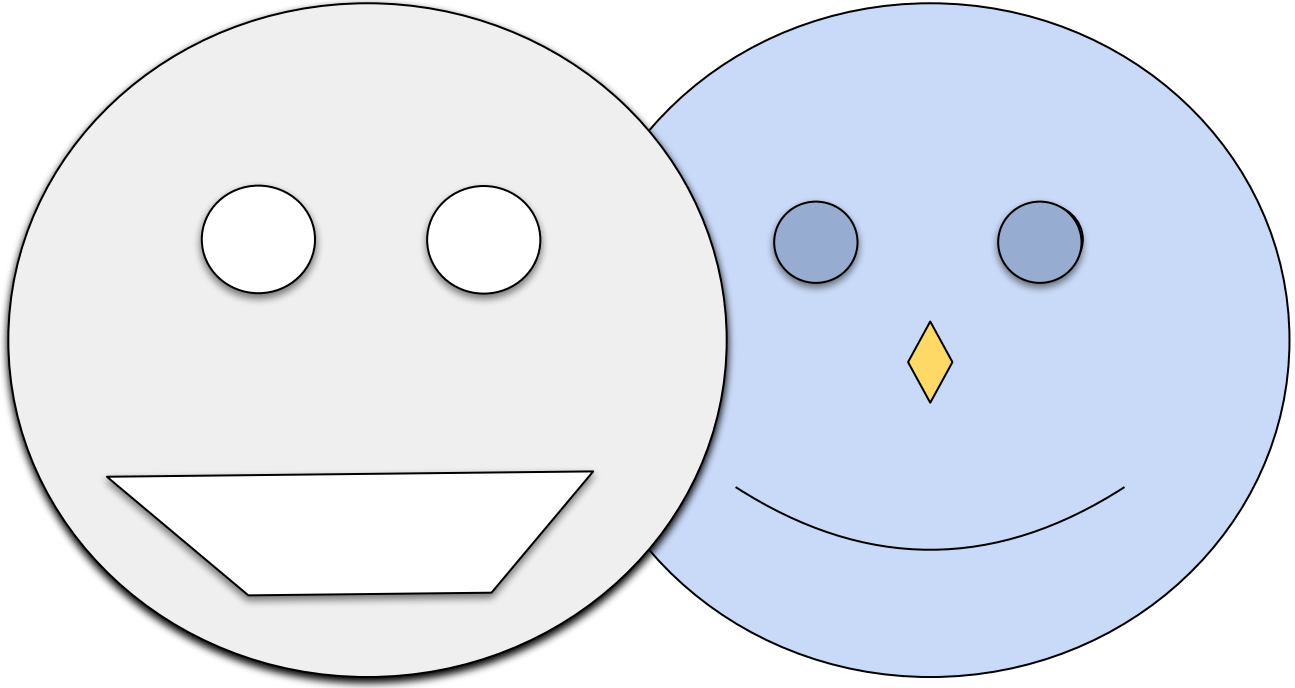
obj = new

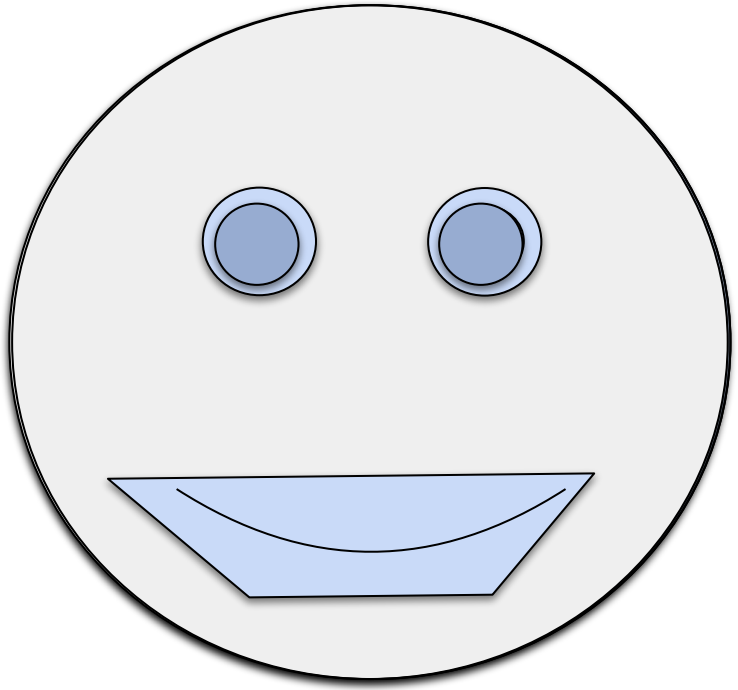
obj = new

```
{
  OlhoEsquerdo =  ,
  OlhoDireito  =  ,
  CorPele      =  ,
  Nariz        =
};
```

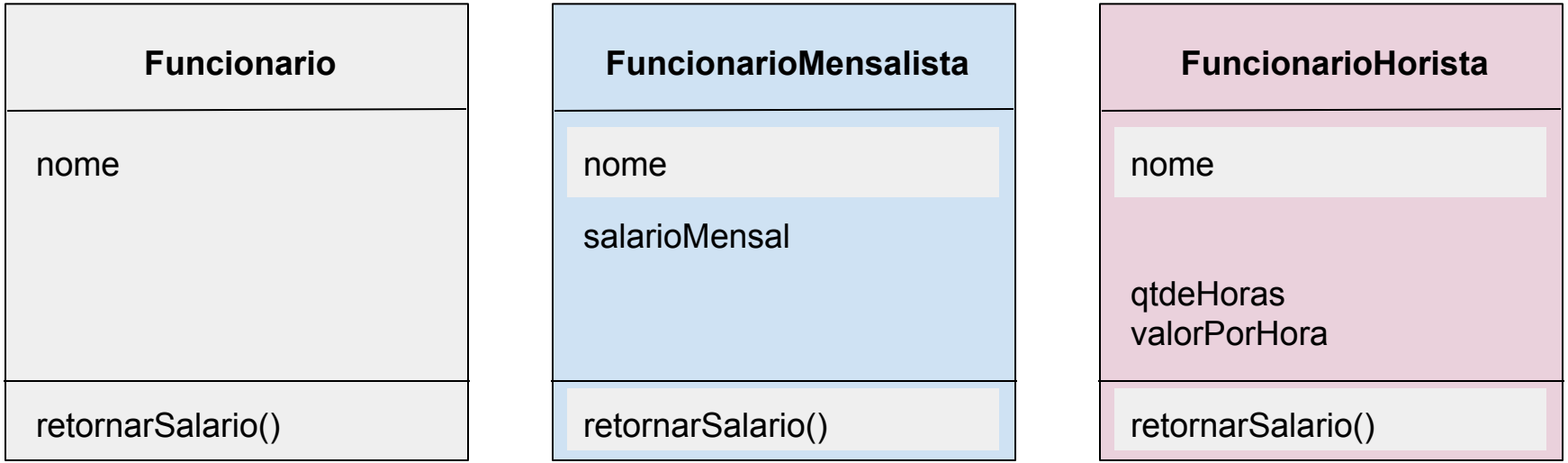




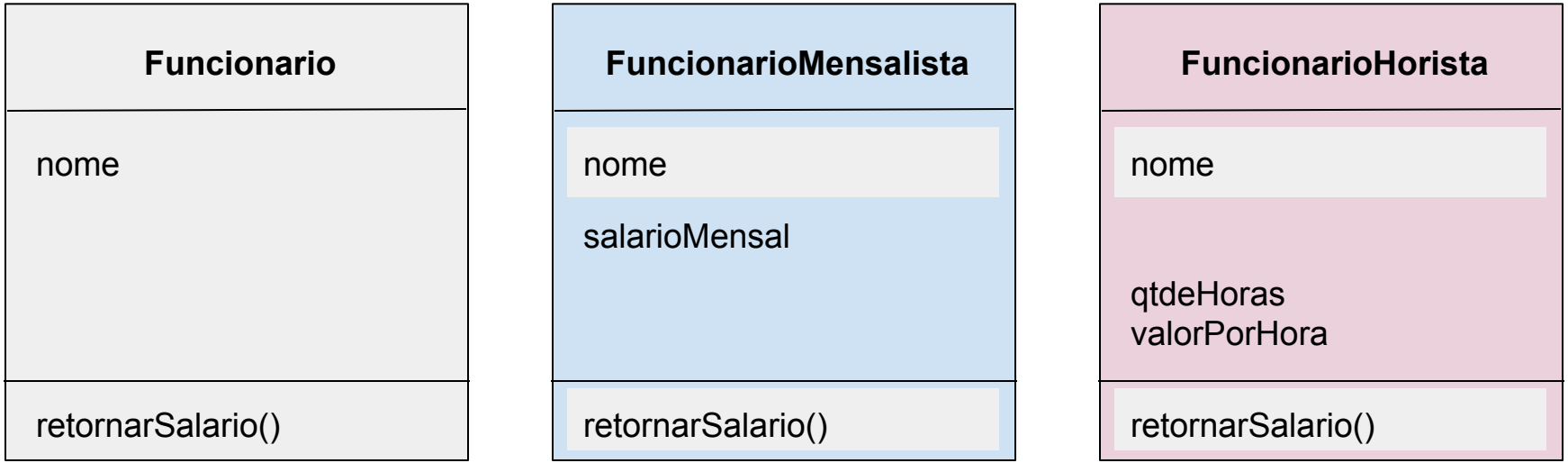




```
Funcionario func = new FuncionarioMensalista ();
```



```
Funcionario func = new FuncionarioHorista ();
```



Como implementar isso?

Classe Funcionario

```
public abstract class Funcionario {
    public abstract double retornarSalario();

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    private String nome;
}
```

Classe FuncionarioMensalista

```
public class FuncionarioMensalista extends Funcionario {
    public double getSalarioMensal() {
        return salarioMensal;
    }

    public void setSalarioMensal(double salarioMensal) {
        this.salarioMensal = salarioMensal;
    }

    @Override
    public double retornarSalario() {
        return salarioMensal;
    }

    private double salarioMensal;
}
```

Classe FuncionarioHorista

```
public class FuncionarioHorista extends Funcionario {
    private int qtdeHoras;

    public int getQtdeHoras() {
        return qtdeHoras;
    }

    public void setQtdeHoras(int qtdeHoras) {
        this.qtdeHoras = qtdeHoras;
    }

...
```

Classe FuncionarioHorista

```
...
    private double valorPorHora;

    public double getValorPorHora() {
        return valorPorHora;
    }

    public void setValorPorHora(double valorPorHora) {
        this.valorPorHora = valorPorHora;
    }

    @Override
    public double retornarSalario() {
        return qtdeHoras * valorPorHora;
    }
}
```

Exemplo Polimorfismo 1

```
public class Main {
    static void imprimeFuncionario(Funcionario func)
    {
        System.out.printf("Funcionário: %s, salário: %.2f\n",
            func.getNome(), func.retornarSalario());
    }
...
```

Exemplo Polimorfismo 1

```
...
    public static void main(String[] args) {
        FuncionarioMensalista func1 = new FuncionarioMensalista();
        func1.setNome("Bruno");
        func1.setSalarioMensal(3000);

        imprimeFuncionario(func1);

        FuncionarioHorista func2 = new FuncionarioHorista();
        func2.setNome("Carlos");
        func2.setQtdeHoras(120);
        func2.setValorPorHora(30);

        imprimeFuncionario(func2);
    }
}
```

Exemplo Polimorfismo 2

```
...
ArrayList<Funcionario> funcionarios = new ArrayList<Funcionario>();
funcionarios.add(func1);
funcionarios.add(func2);

double totalFolhaPagamentos = 0;

for (Funcionario func: funcionarios) {
   totalFolhaPagamentos += func.retornarSalario();
}

System.out.printf("Total da folha de pagamentos: %.2f\n",
       totalFolhaPagamentos);
```

Exercícios

1) No exemplo apresentado anteriormente, adicione um funcionário diarista, cujo salário mensal é calculado pela multiplicação do número de dias pelo valor da diária.

Exercícios

2) Utilizando o projeto do mini sistema bancário, que trabalhamos em aulas anteriores, crie um método na classe Main que recebe por parâmetro um objeto do tipo da superclasse, solicita ao usuário um valor de saque e realiza o saque no objeto recebido por parâmetro. No método main, chame este método duas vezes: uma passando uma instância da classe ContaCorrente e outra passando uma instância da classe ContaPoupanca.

Interfaces

Criando um contrato de implementação

Interfaces

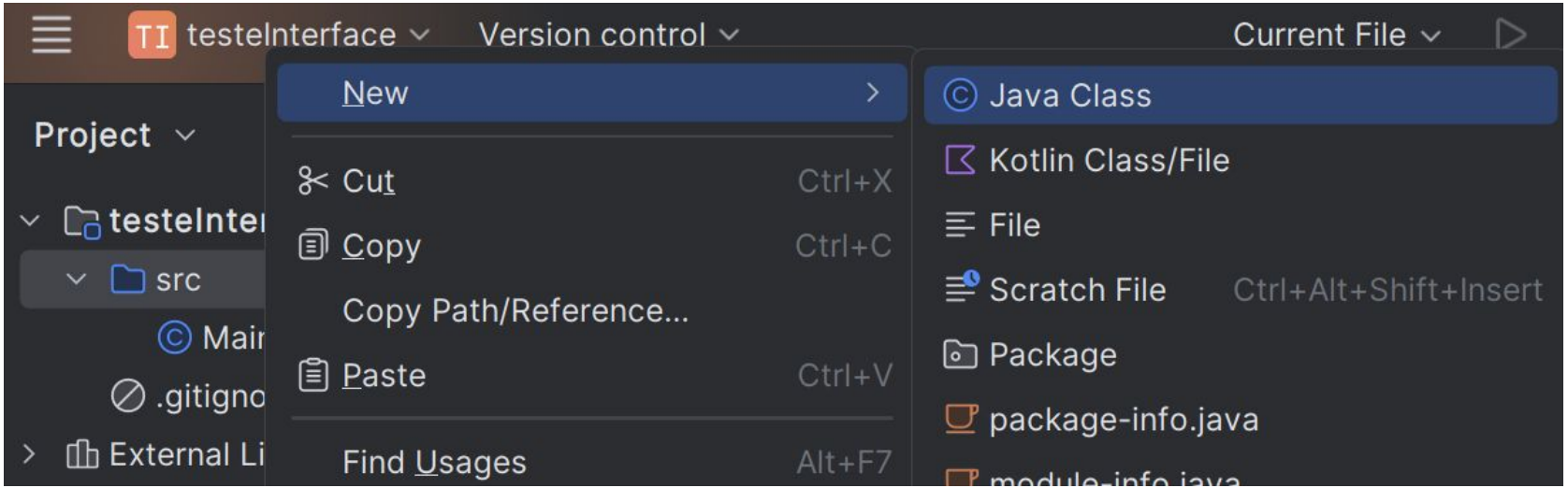
- Uma interface cria um contrato de implementação onde a classe que implementa a interface precisa possuir os métodos definidos na mesma.
- Uma interface é semelhante a uma classe abstrata onde todos os métodos são abstratos, porém, em Java, uma classe só pode herdar de uma única superclasse enquanto uma classe pode implementar várias interfaces.

Interfaces

- Uma variável do tipo da interface pode receber uma instância de qualquer classe que implemente essa interface, logo uma interface pode ser utilizada no lugar de uma superclasse quando se deseja comportamento polimórfico.

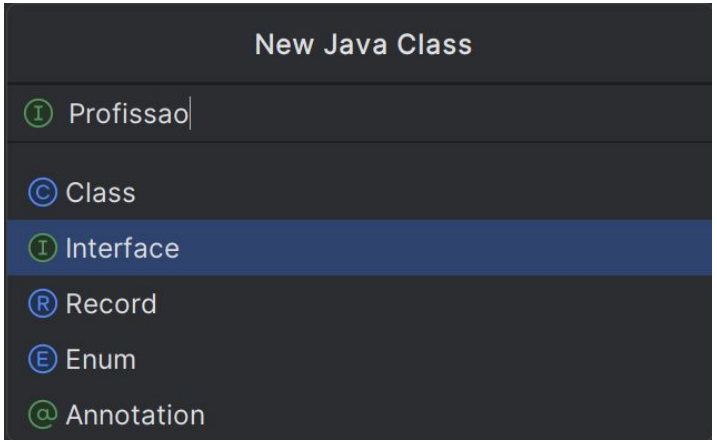
Interfaces

- Adicionando interfaces no IntelliJ:
  - Clique com o botão direito do mouse na pasta src
    - Escolha New → Java Class



Interfaces

- Pressione a seta para baixo do teclado para selecionar “Interface” na lista que aparece.
  - Em seguida, digite o nome da interface, neste caso, digitei “Profissao”.



Interfaces - default

- Em Java, é possível inserir em uma interface uma implementação padrão para determinados métodos.
  - Para isso, utilize a palavra reservada default:

```
default void depositar(double quantia) throws Exception {
    if (quantia <= 0) {
        throw new Exception("O valor precisa ser positivo.");
    }

    setSaldo(getSaldo() + quantia);
}
```

Exercícios

3) Altere o sistema bancário de forma que a classe Conta seja transformada em uma interface.

Fim de material

Dúvidas?

Você também pode me contactar:

Telegram: @camillofalcao

camillofalcao@gmail.com