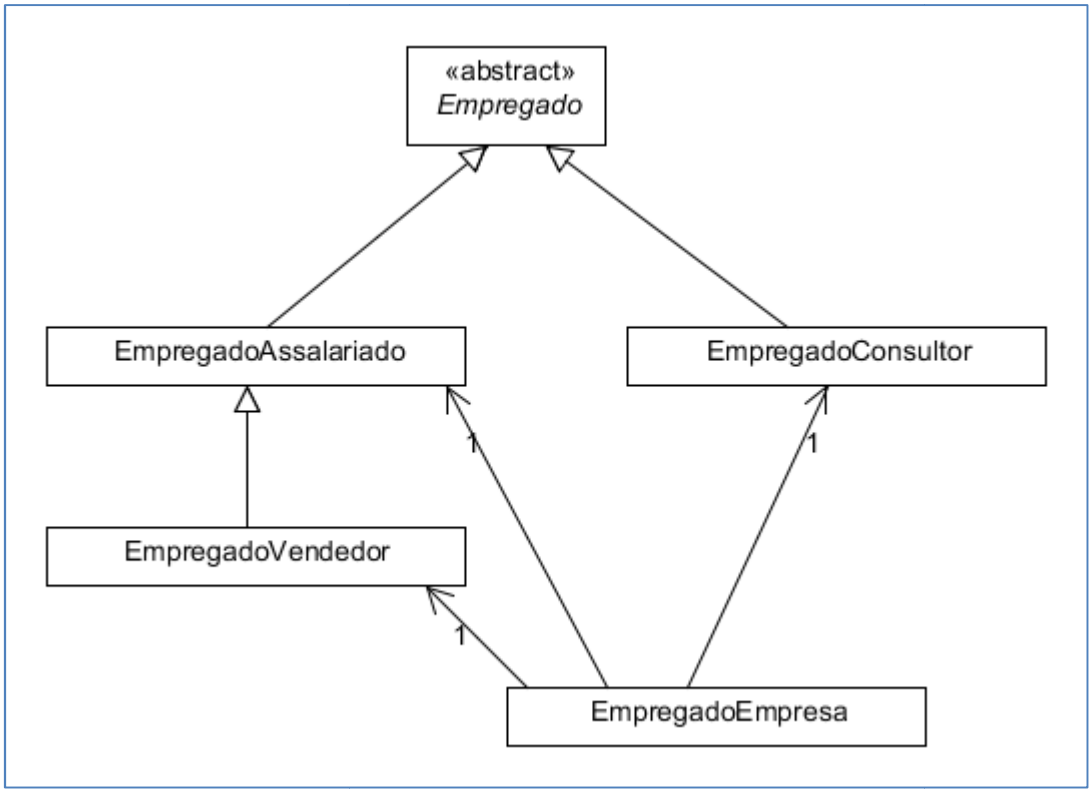
Programação em Java – Fundamentos

Capítulo 3

Exercícios

Exercício Empregado

Neste exercício vamos criar as classes apresentadas neste diagrama:



O exercício pretende praticar a definição de classe, a aplicação do mecanismo de herança, a utilização de abstract, a criação de objetos e o trabalho com as suas variáveis e métodos.

Classe Empregado

Vamos considerar a classe **Empregado** listada abaixo:

```
package capitulo3;
public abstract class Empregado {

    public static long ultimoNumEmp=0;

    public String nome;
    public long numero;

    public Empregado(String s) {
        super();
        this.nome = s;
        this.numero = ++ultimoNumEmp;
    }

    //declaro método abstrat
```

```
    public abstract float calcularSalarioBruto();
}
```

Observações:

- Métodos e variáveis recebem qualificador o qualificador **public.** No próximo capítulo abordaremos o tema dos qualificadores, pelo que para já ficam public;
- A classe é abstract porque recebe o método calcularSalarioBruto() como abstract. Desta forma todas as subclasses são obrigadas a implementar este método

Classe EmpregadoAssalariado

Defina a classe EmpregadoAssalariado como extensão de Empregado. Esta classe representa um empregado que ganha em função do número de horas que trabalha. Para isso vamos acrescentar:

1. Variáveis de objeto:
    a. numHoras - numero de horas de trabalho;
    b. salHora – salário/hora
2. Um construtor que recebe o nome do empregado, o número de horas de trabalho e o respetivo salário/hora. Na execução começa por invocar o construtor de Empregado (atenção que esta classe não tem construtor por omissão) e atribui os parâmetros recebidos às variáveis de objeto com o mesmo nome;
3. O método calcularSalarioBruto() que devolve o produto entre o número de horas e o salário por hora.;

Segue uma possível implementação desta classe:

```
package capitulo3;
public class EmpregadoAssalariado extends Empregado {

    public float numHoras;
    public float salHora;

    public EmpregadoAssalariado(String s, float numHoras, float salHora) {
        super(s);  //evocar construtor superclass para inicializar dados
        this.numHoras = numHoras;
        this.salHora = salHora;
    }

    @Override
    public float calcularSalarioBruto() {
        return (numHoras*salHora);
    }
}
```

Repare que o construtor de EmpregadoAssalariado tem que chamar o construtor de Empregado utilizando a instrução **super(s), visto que a classe Empregado não possui construtor por omissão**.

Se não redefinir o método calcularSalárioBruto() obterá um erro de compilação, alertando-o para a necessidade de colocar a classe como **abstract**. Isto é uma consequência de termos declarado esse método como **abstract** na super classe.

Classe EmpregadoVendedor

Vamos definir a classe EmpregadoVendedor como extensão de EmpregadoAssalariado. Esta classe representa um empregado que trabalha um número de horas variável e também recebe salário em função das suas vendas. Para isso vamos acrescentar:

1) Variáveis de objeto:
   a) vendas – total de vendas feitas pelo vendedor, em euros;
   b) comissão – valor da comissão que o vendedor vai receber, representado sob a forma de percentagem do total de vendas;
2) Um construtor que recebe o nome do empregado, o número de horas de trabalho, o respetivo salário/hora, a comissão nas vendas e o valor total de vendas. Na execução começa por invocar o construtor de EmpregadoAssalariado (atenção que esta classe não tem construtor por omissão) e atribui os parâmetros recebidos às variáveis de objeto com o mesmo nome;
3) Método calcularComissaoVendas() que devolve o produto entre o valor das vendas e a comissão das vendas (em percentagem);
4) Redefinir calcularSalárioBruto() pois agora há uma comissão de vendas que tem que ser adicionada ao salário base;

Sugestão: na implementação do método calcularSalarioBruto() aproveite o método da super classe sendo adicionada a comissão de vendas.

Classe EmpregadoConsultor

Vamos criar a classe EmpregadoConsultor que herda as características de Empregado, mas que possui um salário fixo mensal. Para isso vamos acrescentar:

1) Variável de objeto:
   a) salMensal – representa o salário mensal fixo e bruto em euros;
2) Construtor que invoca o construtor de empregado;
3) Redefinir calcularSalarioBruto() que devolve o salário mensal:

Classe Empresa

Vamos criar a classe **Empresa** que instancia objetos das classes anteriores e testa o seu funcionamento. Para isso deve ser criado um objeto de cada classe usando o construtor e deve verificar os valore das variáveis e testar o funcionamento dos métodos.

```
package capitulo3;
public class EmpregadoEmpresa {
    public static void main(String[] args) {
        System.out.println("========== EmpregadoAssalariado =============");
        EmpregadoAssalariado assal1=
                  new EmpregadoAssalariado("António", 176.5f, 3.5f);
```

```
        System.out.println("Nome..........="+assal1.nome);
        System.out.println("Numero........="+assal1.numero);
        System.out.println("Numero de Horas="+assal1.numHoras);
        System.out.println("Salario Hora...="+assal1.salHora);
        System.out.println("Salario Bruto..="+assal1.calcularSalarioBruto());
        System.out.println("=== Assalariado com novo " +
                           "nome, mais horas e mais salario/hora");
        assal1.numHoras=200f;
        assal1.salHora=4.75f;
        assal1.nome="António Gonçalves";
        System.out.println("Nome..........="+assal1.nome);
        System.out.println("Numero........="+assal1.numero);
        System.out.println("Numero de Horas="+assal1.numHoras);
        System.out.println("Salario Hora...="+assal1.salHora);
        System.out.println("Salario Bruto..="+assal1.calcularSalarioBruto());




    }
}
```

Cotação:

Empregado – 10%

EmpregadoAssalariado-20%

EmpregadoVendedor-30%

EmpregadoConsultor-10%

EmpregadoEmpresa-30%