RETO 4

Consideraciones Generales:

- El reto deberá ser desarrollado de forma individual por cada tripulante
- Debe usar los conocimientos vistos en la semana 4
- Debe tener un contexto real
- Debe tener un conjunto de requerimientos funcionales bien definidos
- El reto deberá ser distinto por cada grupo de tripulantes (curso)
- El reto será calificado con una nota de 0.0 a 5.0 y corresponderá al 20% de la nota final del ciclo 2
- Debe entregarse y explicarse al beneficiario al finalizar la semana 5
- El tripulante contará con una semana de plazo para el desarrollo del reto

Problemática a resolver (Contexto):

Se pretende desarrollar un conjunto de clases que representen, de forma simplificada, a una hipotética empresa dedicada a vender un producto. A continuación, se describen las características básicas de estas clases:
1. **Empleado**. Clase básica que describe a un empleado. Incluye sus datos
personales (nombre, apellidos, documento y dirección)
Tendrá, al menos, los siguientes métodos:
• Constructores para definir correctamente un empleado, a partir de su nombre,
apellidos, documento y dirección.
• Imprimir (A través de los operadores de E/S redefinidos)
2. **Secretario**. Tiene despacho y número de teléfono.
Tendrá, al menos, los siguientes métodos:
• Constructores (debe rellenar la información personal y los datos principales)
• Imprimir (debe imprimir sus datos personales, despacho y número de teléfono).
3. **Vendedor**. Tiene área de venta y comisión.
Tendrá, al menos, los siguientes métodos:
• Constructores (debe rellenar la información personal y los datos principales)
• Imprimir (debe imprimir sus datos personales, área de venta y comisión).
4. **Jefe de zona**. Tiene despacho y email.
Tendrá, al menos, los siguientes métodos:
• Constructores (debe rellenar la información personal y los datos principales)
• Imprimir (debe imprimir sus datos personales, despacho y email).

Todos los empleados son vendedores, jefes de zona o secretarios. Hacer un programa de prueba que muestre como funciona. A partir del código java generar el **Diagrama de Clases UML**.

Requerimientos:

En la realización del reto de la Semana 4, se propone la creación de un problema el cual el tripulante desarrolle un programa en JAVA Netbeans 8.2 aplicando los conceptos de herencia, polimorfismo y MOO con UML.

Evaluación:

La evaluación del reto 4 se desarrollará de forma manual
El tripulante deberá adjuntar archivo comprimido con su estructura lógica en la actividad correspondiente al reto 4.
La calificación resultante estará entre 0.0 y 5.0

¡ÉXITOS!

M.Ed. NÉSTOR ANAYA CHÁVEZ

SOLUCIÓN

Solución modificable a un enunciado como el siguiente:

Se pretende desarrollar un conjunto de clases que representen, de forma simplificada, a una hipotética empresa dedicada a vender un producto.

A continuación, se describen las características básicas de estas clases:

1. **Empleado**. Clase básica que describe a un empleado. Incluye sus datos personales (nombre, apellidos, DNI, dirección) y algunos datos tales como los años de antigüedad, teléfono de contacto y su salario. Incluye también información de quién es el empleado que lo supervisa (Empleado *). Tendrá, al menos, las siguientes funciones miembro: • Constructores para definir correctamente un empleado, a partir de su nombre, apellidos, DNI, dirección, teléfono y salario. • Imprimir (A través de los operadores de E/S redefinidos) • Cambiar supervisor • Incrementar salario.

2. **Secretario**. Tiene despacho, número de fax e incrementa su salario un 5% anual. Tendrá, al menos, las siguientes funciones miembro: • Constructores (debe rellenar la información personal y los datos principales) • Imprimir (debe imprimir sus datos personales y su puesto en la empresa).

3. **Vendedor**. Tiene coche de la empresa (identificado por la matricula, marca y modelo), teléfono móvil, área de venta, lista de clientes y porcentaje que se lleva de las ventas en concepto de comisiones. Incrementa su salario un 10% anual. Tendrá, al menos, las siguientes funciones miembro: • Constructores (debe rellenar la información personal y los datos principales) • Imprimir (debe imprimir sus datos personales y su puesto en la empresa). • Dar de alta un nuevo cliente. • Dar de baja un cliente. • Cambiar de coche.

4. **Jefe de zona**. Tiene despacho, tiene un secretario a su cargo, una lista de vendedores a su cargo y tiene coche de la empresa (identificado por la matrícula, marca y modelo). Incrementa su salario un 20% anual. Tendrá, al menos, las siguientes funciones miembro: • Constructores (debe rellenar la información personal y los datos principales) • Imprimir (debe imprimir sus datos personales y su puesto en la empresa). • Cambiar de secretario. • Cambiar de coche. • Dar de alta y de baja un nuevo vendedor en su zona. Todos los empleados son vendedores, jefes de zona o secretarios.

Hacer un programa de prueba que muestre como funciona. Probar, especialmente, que el método incrementar salario se comparta bien, según el empleado que sea así es la subida.

Clase empleado

```
public abstract class empleado {                    /* con mayusculo porfa la clase por buenas practicas */
   private String apellido;
   private String direccion;
   private String nombre;
   private String telefono;
   private String dni;
    private  int años ;

    double salario ;

   public Empleado(String apellido, String direccion, String nombre, String telefono, String dni, int años, double
salario) {
        this.apellido = apellido;
        this.direccion = direccion;
        this.nombre = nombre;
        this.telefono = telefono;
        this.dni = dni;
        this.años = años;
        this.salario = salario;
   }

   public String getApellido() {
        return apellido;
   }

   public String getDireccion() {
        return direccion;
   }

   public String getNombre() {
        return nombre;
```

```
    }

    public String getTelefono() {
        return telefono;
    }

    public String getDni() {
        return dni;
    }

    public int getAños() {
        return años;
    }

    public double getSalario() {
        return salario;
    }


    public void imprimir() {
            System.out.println(nombre+" || "+apellido+" ||  "+direccion+" || "+telefono+" || "+dni+" || " +años+"
||"+salario);
    }
}
```

//

Clase del jefe

```
public  class JefeDeZona extends Empleado {    /* hereda de la clase empleado*/

    public JefeDeLaZona(String secreTarioAcargo, String listaVende, String apellido, String direccion, String
nombre, String telefono, String dni, int años, double salario) {
```

```
        super(apellido, direccion, nombre, telefono, dni, años, salario);
        this.secretarioAcargo = secreTarioAcargo;
        this.listaVende = listaVende;
    }



        private String secretarioAcargo, listaVende;



         public void imprimir() {
        super.imprimir();
            System.out.println(secretarioAcargo+" -- "+listaVende);

         }
         public void cambiarSecretario(){}
         public void cambiarCarro(){}
         public void  darAlta(){}
public void darBaja(){}

    public double incrementoSalario() {
       return salario*0.2+salario;
    }

    public String getSecretarioAcargo() {
        return secretarioAcargo;
    }

    public String getListaVende() {
        return listaVende;
    }

    public double getSalario() {
        return salario;
    }
```

```
    void salario(double d) {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
    }
    public  class JefeDeLaZona extends Empleado {

     public JefeDeLaZona(String secreTarioAcargo, String listaVende, String apellido, String direccion, String nombre, String telefono, String dni, int años, double salario) {
        super(apellido, direccion, nombre, telefono, dni, años, salario);
        this.secretarioAcargo = secreTarioAcargo;
        this.listaVende = listaVende;
    }


        private String secretarioAcargo, listaVende;


         public void imprimir() {
        super.imprimir();
            System.out.println(secretarioAcargo+" -- "+listaVende);

         }
         public void cambiarSecretario(){}
         public void cambiarCarro(){}
         public void  darAlta(){}
public void darBaja(){}

    public double incrementoSalario() {
       return salario*0.2+salario;
    }

    public String getSecretarioAcargo() {
        return secretarioAcargo;
```

```
    }

    public String getListaVende() {
        return listaVende;
    }

    public double getSalario() {
        return salario;
    }

    void salario(double d) {
            throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
    }
```

teniendo en cuenta estas dos clases lo restante seria copiar y pegar básicamente lo mismo para las otras clases.

otra solución en el siguiente link:

https://www.aprendeaprogramar.com/mod/forum/discuss.php?d=3721