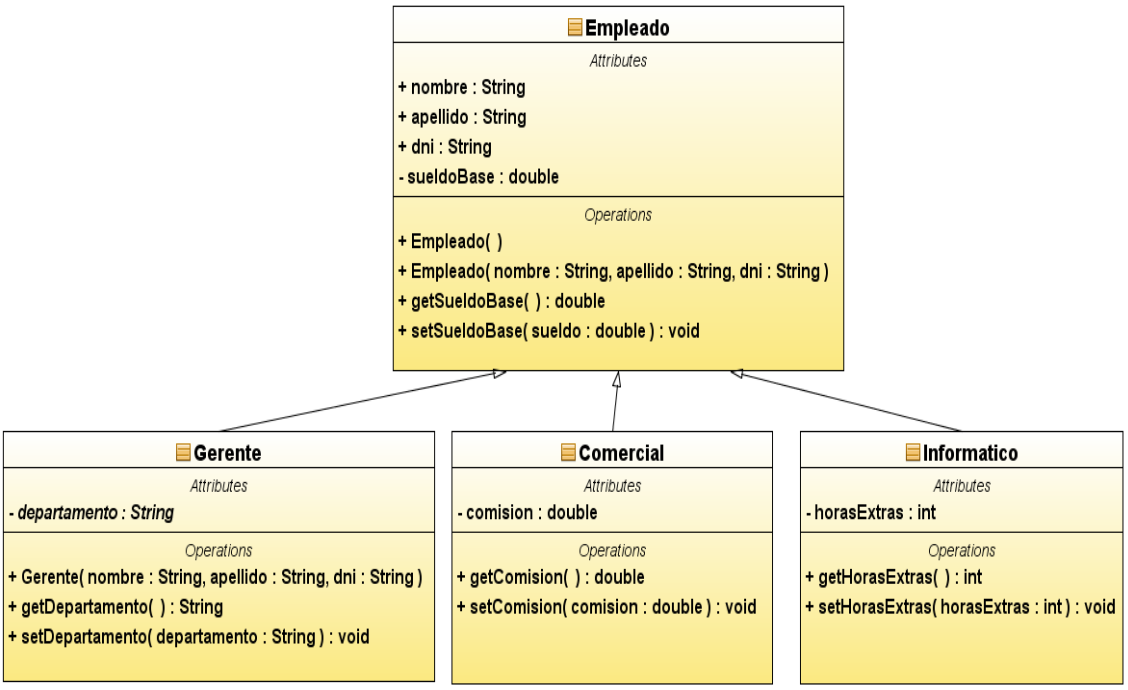
Herencia y polimorfismo

En el siguiente ejercicio desarrollaremos un programa Java que instancie varios objetos y que muestre la información de dichas instancias. Utilizaremos cuatro clases: Gerente, Informatico, Comercial y Empleado. Como se aprecia en el siguiente diagrama, la clase Empleado es superclase del resto de las clases, por lo que tendremos que tener en cuenta la jerarquía de herencia.



Los empleados tendrán un nombre, apellido, DNI y sueldo base. Cada gerente debe tener un nombre de departamento asignado, los comerciales tendrán comisiones por las ventas realizadas, y los Informáticos, los cuales además de trabajar en la oficina trabajan desde casa, deben reflejar el número de horas trabajadas en casa. También se indicará en cada clase los correspondientes constructores así como los métodos de acceso a los atributos privados.

IMPLEMENTACIÓN

```java
// Archivo Empleado.java

public class Empleado {

 public String nombre;
 public String apellido;
 public String dni;
 private double sueldoBase;

 public Empleado () { //#1

 }

 public Empleado (String nombre,String apellido, String dni) { //#1

    this.nombre = nombre; //#2
    this.apellido = apellido; //#2
    this.dni = dni; //#2
 }

 public double getSueldoBase () {

    return sueldoBase;
 }

 public void setSueldoBase (double sueldoBase) {

    this.sueldoBase = sueldoBase; //#2
 }

}



// Archivo Gerente.java

public class Gerente extends Empleado { //#3

 private String departamento;


 public Gerente (String nombre,String apellido, String dni) {

    super.nombre = nombre; //#4
    super.apellido = apellido; //#4
    super.dni = dni; //#4
 }


 public String getDepartamento () {

    return departamento;
 }

 public void setDepartamento (String departamento) {

    this.departamento = departamento; //#2
 }
```

```
}

// Archivo Comercial.java

public class Comercial extends Empleado{ //#3

 private double comision;

 public double getComision () {

    return comision;
 }

 public void setComision (double comision) {

    this.comision = comision; //#2
 }

}

// Archivo Informatico.java

public class Informatico extends Empleado{ //#3

 private int horasExtras;

 public int getHorasExtras () {

    return horasExtras;
 }

 public void setHorasExtras (int horasExtras) {

    this.horasExtras = horasExtras; //#2
 }

}

// Archivo Empresa.java.
// La clase Empresa la utilizaremos para instanciar las clases y
mostrar la información de las instancias.


 public class Empresa {

 public static void main(String[] args) {

    Empresa ema= new Empresa (); //#5
    Gerente g1= new Gerente("Dario","Sanz","12345678G"); //#5
    Comercial c1 = new Comercial(); //#5
    Informatico i1 = new Informatico(); //#5

    ema.insertarGerentes(g1,"Financiero",2000); //#6
    ema.insertarComercial(c1,"Mara","Runa","1111111C",250,2500); //#6
    ema.insertarInformatico(i1,"Juan","Pio","22222222I",24,2000); //#6
```

```
    ema.imprimirInformacion(g1); //#7
    ema.imprimirInformacion(c1); //#7
    ema.imprimirInformacion(i1); //#7

  }


 public void insertarGerentes (Gerente ger, String departamento,
double sueldo){

    ger.setDepartamento(departamento);
    ger.setSueldoBase(sueldo);

  }


 public void insertarComercial (Comercial co, String nombre, String
apellido,String dni,double comision, double sueldo){

    co.nombre= nombre;
    co.apellido= apellido;
    co.dni= dni;
    co.setComision(comision);
    co.setSueldoBase(sueldo);

     }


 public void insertarInformatico (Informatico inf, String nombre,
String apellido,String dni,int horas, double sueldo){

     inf.nombre= nombre;
     inf.apellido= apellido;
     inf.dni= dni;
     inf.setHorasExtras(horas);
     inf.setSueldoBase(sueldo);

      }


 public void imprimirInformacion(Empleado e){

     if ( e instanceof Gerente ) { //#8

        Gerente g1 = (Gerente) e; //#9

        System.out.println("Nombre gerente: " + g1.nombre + "\n" +
"Apellido gerente: " + g1.apellido + "\n" + "DNI gerente: " + g1.dni +
"\n" + "Departamento gerente: " + g1.getDepartamento() + "\n" +
"Sueldo base gerente: " + g1.getSueldoBase() + "\n");
          }

     if ( e instanceof Comercial ) { //#8

         Comercial c1 = (Comercial) e; //#9

         System.out.println("Nombre comercial: " + c1.nombre + "\n" +
"Apellido comercial: " + c1.apellido + "\n" + "DNI comercial: " +
c1.dni + "\n" + "Comisión comercial: " + c1.getComision() + "\n" +
"Sueldo base comercial: " + c1.getSueldoBase() + "\n");
          }
```

```
     if ( e instanceof Informatico ) { //#8

          Informatico i1 = (Informatico) e; //#9

          System.out.println("Nombre informático: " + i1.nombre + "\n"
+ "Apellido informático: " + i1.apellido + "\n" + "DNI informático: "
+ i1.dni + "\n" + "Horas extras informático: " + i1.getHorasExtras() +
"\n" + "Sueldo base informático: " + i1.getSueldoBase());
           }
     }
}
```

COMENTARIOS:

#1: No hay problema en declarar más de un constructor, ya que java soporta la sobrecarga de constructores.
#2: Con la palabra clave this podemos hacer referencia a las variables miembro de la propia clase. Por lo que resolveremos la ambigüedad entre las variables de instancia y los parámetros.
#3: Para especificar que una clase es subclase de otra, lo haremos agregando la cláusula extends después del nombre de dicha clase.
#4: En estas líneas de código estamos utilizamos la palabra clave super para referenciar a las variables miembro de la superclase.
#5: Instanciación de los correspondiente objetos. Nótese que la clase Gerente tiene un único constructor con 3 parámetros de tipo String, por lo que el intentar invocar a un constructor sin parámetros provocará un error de compilación. Por defecto, toda clase Java tiene un constructor sin parámetros, pero en el momento en el que el programador define un constructor, el constructor por defecto se elimina y es sustituido por el constructor definido.
#6: Llamadas a los métodos de inserción de datos de la clase Empresa. En estas líneas simplemente hemos añadido al objeto ema, el cual es una instancia de la clase Empresa, el nombre de los correspondientes métodos y proporcionándoles los argumentos adecuados.
#7: Llamadas al método imprimirInformacion para mostrar los datos en pantalla. Este método sólo posee un parámetro de tipo Empleado, por lo que los argumentos proporcionados pueden ser subtipos de Empleado, en nuestro caso serán de tipo Gerente, Comercial e Informatico.
#8: Utilizaremos el operador instanceof para saber si el objeto e es instancia de la clase Gerente, Informatico o Comercial. Con este operador podemos consultar si un objeto es una instancia de una determinada clase.
#9: Una vez comprobado el tipo de la instancia a través del operador instanceof, podemos realizar la conversión del objeto de tipo Empleado a cualquier subtipo (Gerente, Informatico o Comercial).