

ArgoUML

Es una herramienta de modelado UML . Se descarga desde la URL <https://argouml.uptodown.com/windows>

Alguna de sus características son:

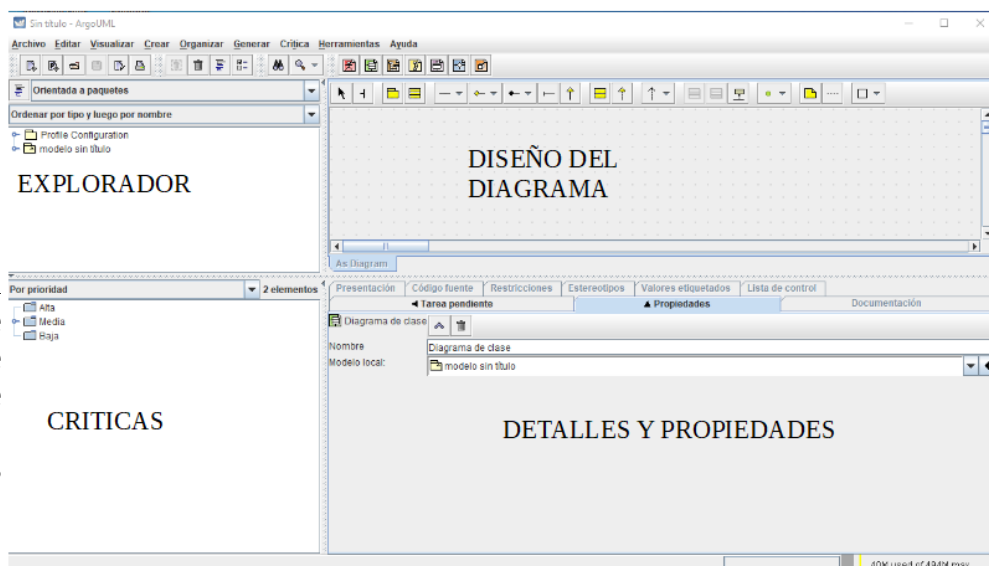
- Permite crear los siguientes tipos de diagramas: casos de uso, clases , secuencia, colaboración, estado, actividades y despliegue.
- Compatible con el estándar UML 1.4
- Proporciona la generación de código para Java, C++, C#, SQL, PHP4 y PHP5.
- Genera fichero PNG, GIF, JPG, SVG, EPS desde los diagramas.
- Dispone de críticas de diseño, que analizan el diseño en el que se trabaja y sugieren posibles mejoras. Estas sugerencias van desde la indicación de errores de sintaxis, a los recordatorios para cumplir directrices de estilo.

Creación de diagrama de clase

Desde la barra de botones de diseño se podrán insertar los elementos al diagrama,

basta con seleccionar el elemento y pinchar en el diseño.

Para crear una asociación , se selecciona el tipo de asociación y se pincha de la clase origen a destino.



Crear paquete



Crear clase



Asociación



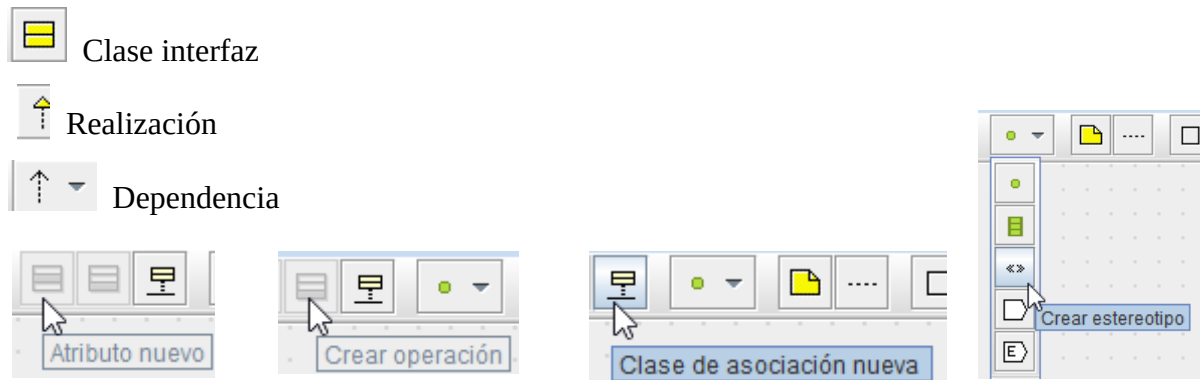
Agregación



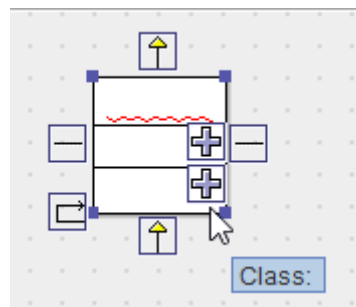
Composición




Generalización



Una forma rápida de crear clases y asociaciones automáticas es hacer clic sobre los iconos de asociación que se muestran al seleccionar la clase y pasar con el ratón.

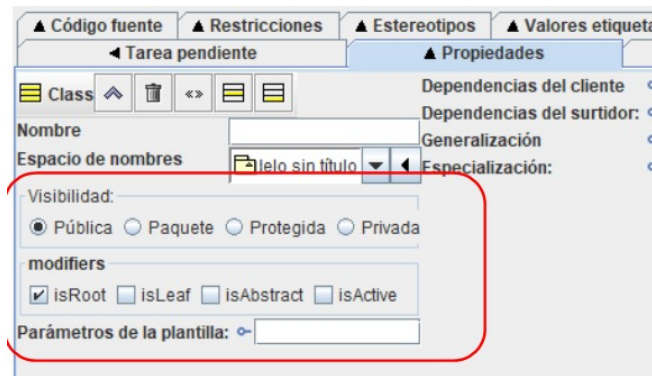


Al crear clase lo primero que hay que escribir es el nombre, y seguidamente añadir los atributos y operaciones, para ello se hace clic sobre los iconos de añadir  de la clase o se pulsa a los botones correspondientes desde la barra de diseño.

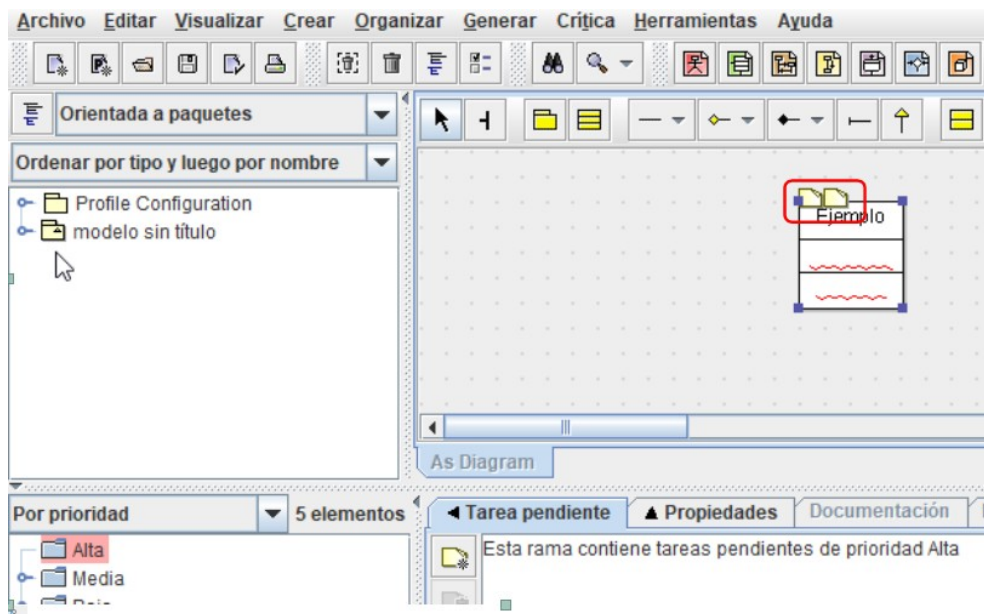
Al crear la clase en las propiedades podemos indicar la visibilidad de la misma (publica, paquete, protegida o privada) y el tipo de clase o modificadores .

Los modificadores son los siguientes:

- **isRoot:** para indicar que la clase no tiene antecesores , es una clase raíz.
- **Is leaf(hoja):** para indicar que la clase no puede en un futuro ser una especialización.
- **Is Abstract:** para indicar si la clase es una clase abstracta.
- **Is Active:** para indicar si la clase es una clase activa. Una clase activa es aquella cuyos objetos poseen uno o más procesos o threads (hilos) y por lo tanto pueden inicial una actividad de control, son objetos activos. Si los objetos de la clase mantienen los datos pero no inician actividad se dice que son objetos pasivos.

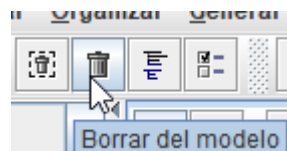


Si el diseño que se está creando , es decir , las clases, los nombres, las asociaciones, etc, no cumple con las reglas establecidas para UML 1.4 , aparecerán las críticas y las sugerencias en el apartado de críticas



Cuando una clase tiene crítica aparecerán unos iconos en la parte superior de la clase que indica que se han detectado críticas. Por defecto las críticas se visualizaran por prioridades , también se pueden visualizar por decisión , por objetivo, por causante, por emisor, o por tipo de conocimiento, basta con seleccionar el tipo de visualización de la lista desplegable que aparecen en la parte superior del panel de críticas.

NOTA: Cuando estemos realizando un diseño y se desea eliminar elementos incorporados al diseño no vale con eliminarlos pulsando la tecla suprimir , esta los elimina de la vista pero permanece en el modelo, Hay que eliminarlos pulsando el botón “borrar modelo” de la barra de botones.



EJERCICIO 1

Se trata de realizar un diagrama de clases para representar las relaciones entre empleados y departamentos.

- Consideramos que un empleado trabaja en un departamento y en el departamento trabajan muchos empleados.
- Datos de los empleados son código, nombre, oficio y salario.
- Datos de los departamentos son código, nombre y localidad.
- Además un empleado puede ser jefe de varios empleados.
- Se necesita crear los métodos para asignar datos a los empleados y departamento y devolverlos (setter y getter)

Al crear la asociación entre las clases se puede añadir el nombre de la asociación, los nombres de los roles de las clases y la multiplicidad.



Cuando se selecciona la asociación, se puede acceder al panel de propiedades

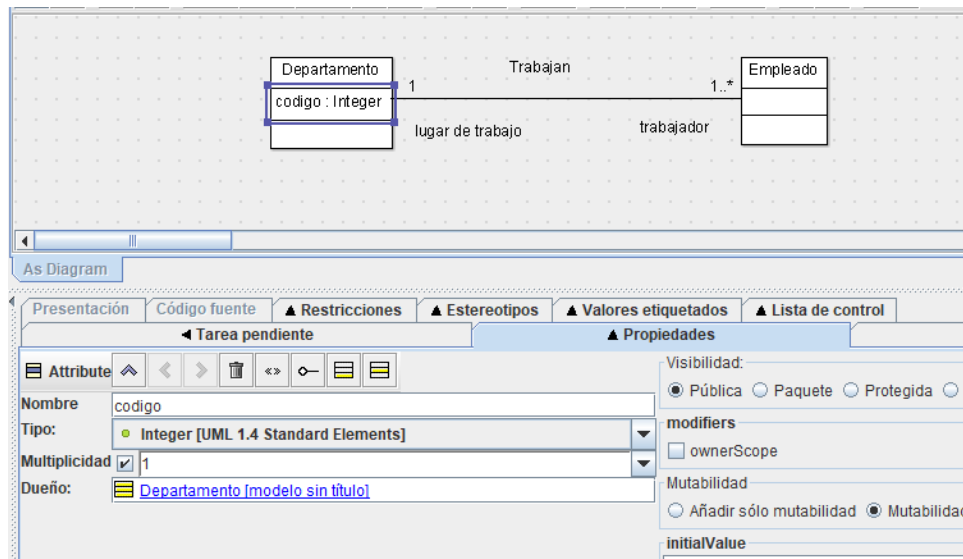
Si hacemos doble clic en la conexión

Vemos que la

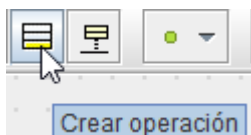
multiplicidad es 1 y es navegable. Para movernos a la conexión destino pulsamos el botón



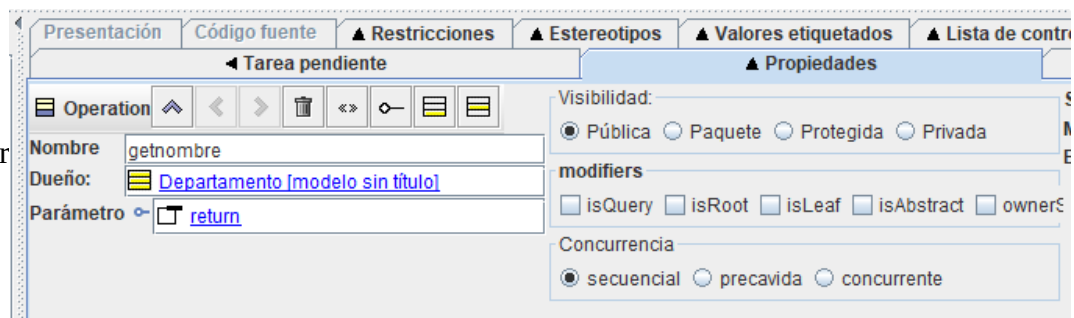
Añadimos los atributos



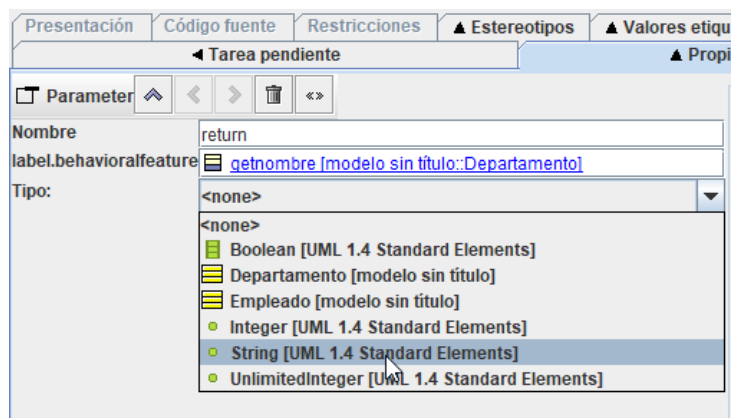
Añadimos los métodos



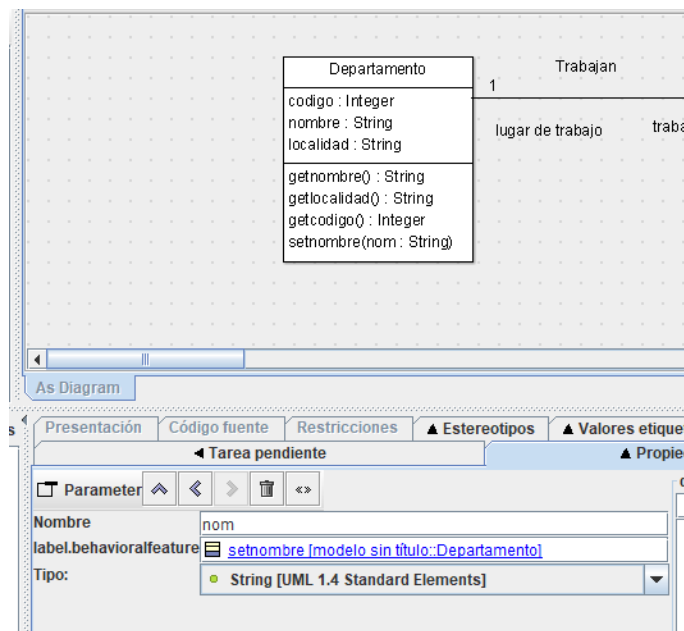
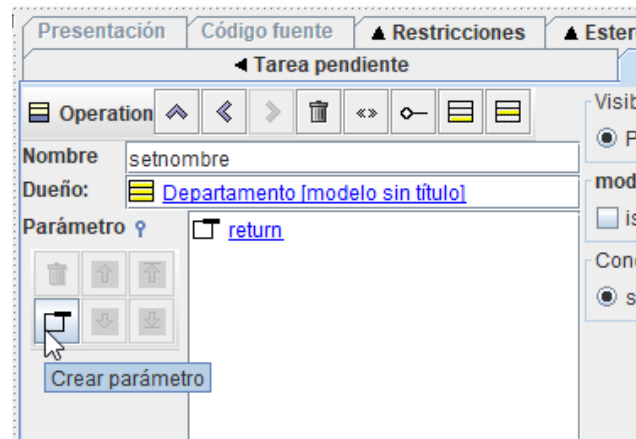
Para
indicar
el valor
que



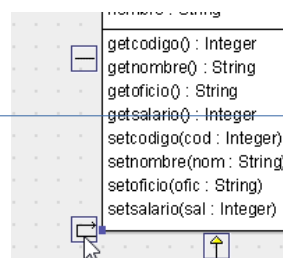
devuelve hacemos clic en return



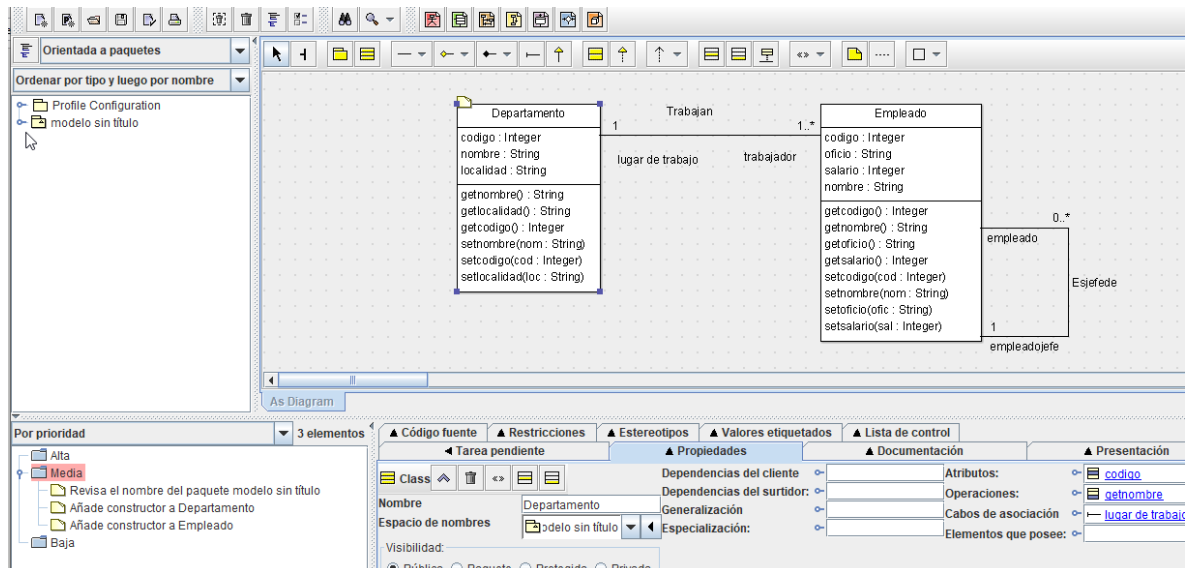
Para indicar el parámetro de entrada hacemos clic



Para crear la asociación reflexiva de empleado con empleado, es decir un empleado puede ser jefe de muchos empleados pulsamos en

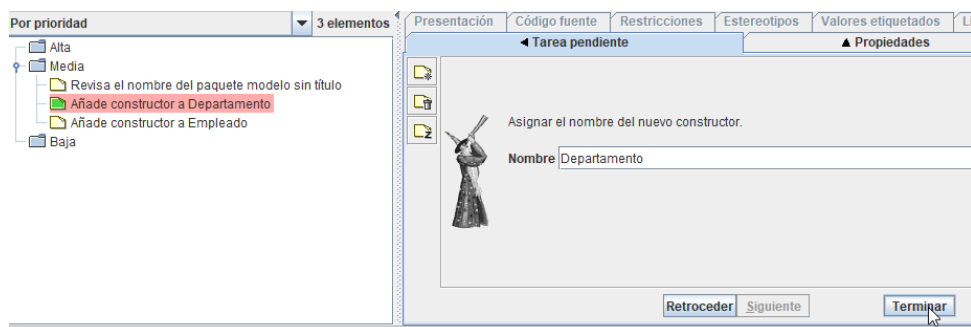
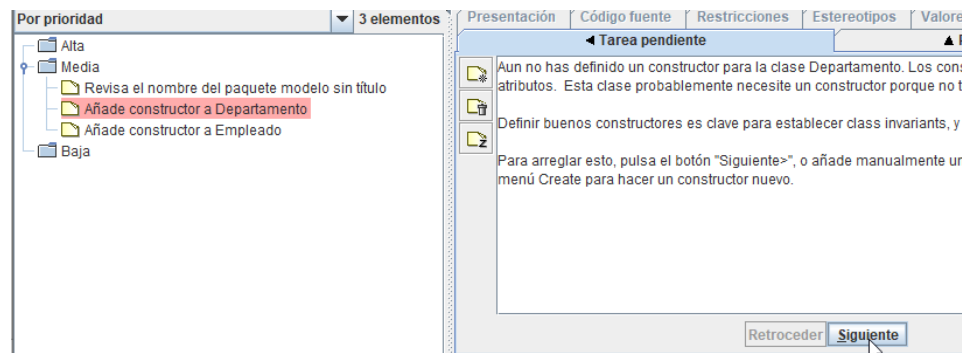


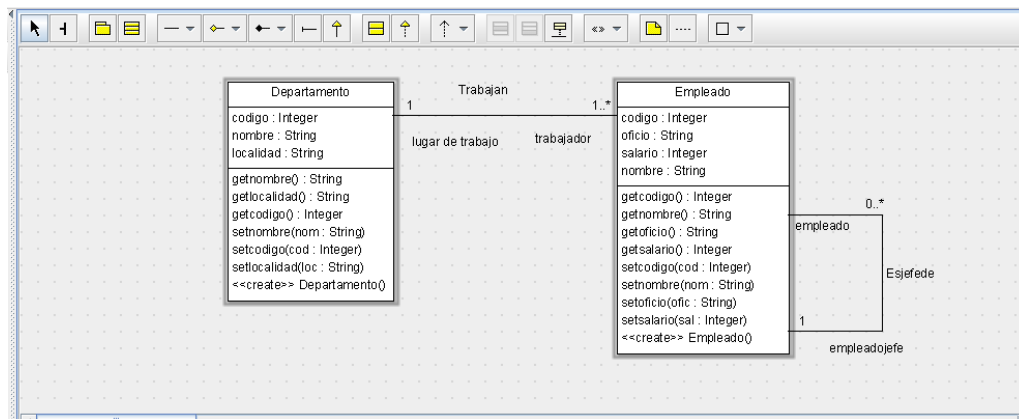
Una vez creadas las clases con los atributos y operaciones conviene revisar si hay alguna crítica .



Comprobamos que tenemos una crítica en cada clase para añadir el constructor.

Pulsamos en siguiente para leer la crítica y la sugerencia de solución, y en la siguiente pantalla hacemos clic en terminar para aceptar el cambio.





Para ver el código generado, se selecciona la clase y en la pestaña código fuente se mostrará el código, hay que asegurarse de seleccionar el lenguaje apropiado .

```

import java.util.Vector;

public class Departamento {

    public Integer codigo;

    public String nombre;

    public String localidad;

    /**
     *
     * @element-type Empleado
     */
    public Vector trabajador;

    public String getnombre() {
        return null;
    }

    public String getlocalidad() {
        return null;
    }

    public Integer getcodigo() {
        return null;
    }

    public void setnombre(String nom) {
    }

    public void setcodigo(Integer cod) {
    }

    public void setlocalidad(String loc) {
    }
}

```



```
}

public Departamento() {
}

}

import java.util.Vector;

public class Empleado {

    public Integer codigo;

    public String oficio;

    public Integer salario;

    public String nombre;

    public Departamento lugar de trabajo;
    public Empleado empleadofefe;
    /**
     *
     * @element-type Empleado
     */
    public Vector empleado;

    public Integer getcodigo() {
        return null;
    }

    public String getnombre() {
        return null;
    }

    public String getoficio() {
        return null;
    }

    public Integer getsalario() {
        return null;
    }

    public void setcodigo(Integer cod) {
    }

    public void setnombre(String nom) {
    }

    public void setoficio(String ofic) {
    }

    public void setsalario(Integer sal) {
    }

    public Empleado() {
```

```

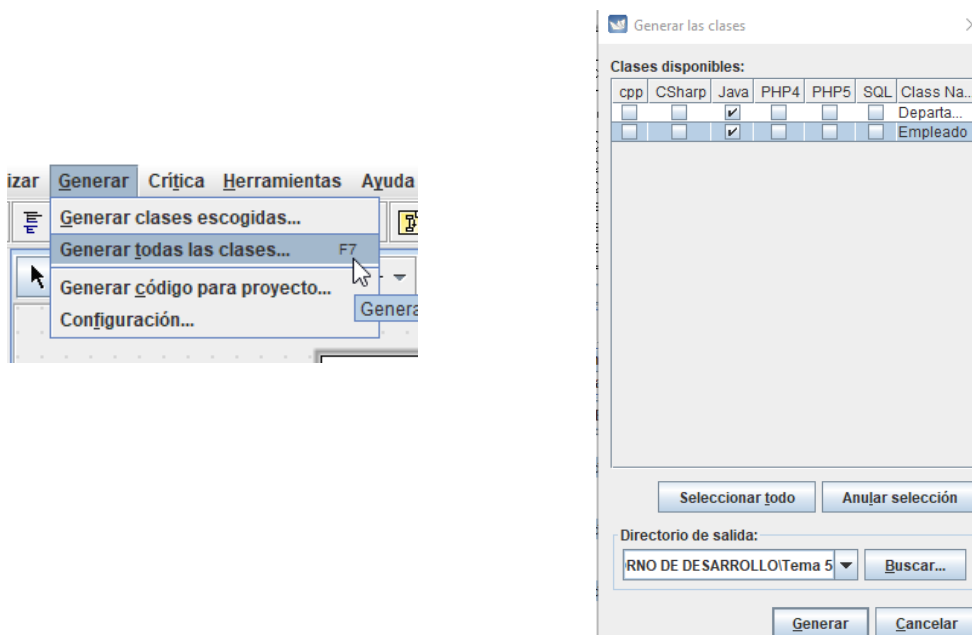
    }
}

```

Si observamos el código, la clase Empleado tiene un objeto Departamento que es Lugar de trabajo, donde se indica el departamento donde trabaja el empleado, y un objeto Empleado que es empleado jefe, donde se indica el empleado que es su jefe. Además cuenta con el vector empleado, que almacenará todos los empleados de los que es jefe, caso de que el empleado sea jefe.

La clase Departamento contiene un vector trabajador del tipo Empleado, que almacenará todos los empleados que trabajan en ese departamento.

Desde el menú Generar podemos generar las clases y crear los archivos correspondiente



Ejercicio 2

Se desea realizar el análisis de un sistema de gestión informática de una pequeña empresa formada por empleados y departamentos. Para ellos se dispone de una base de datos donde están almacenados los datos de los empleados y los departamentos. Consideramos las clases y relaciones del ejercicio anterior (Empleado y Departamento)

Requisitos funcionales: el sistema a analizar debe permitir el acceso al operador para mantener la información de la BD, y para generar informes. El operador es un empleado y es el único autorizado a entrar en el sistema y operar. Será el encargado del mantenimiento de los datos de la base de datos. Las operaciones a realizar son las siguientes:

- El mantenimiento de datos de Empleados incluye altas, bajas, modificaciones y consultas.
- El mantenimiento de datos de Departamentos incluye altas, bajas, modificaciones y consultas.
- Gestión de informes.

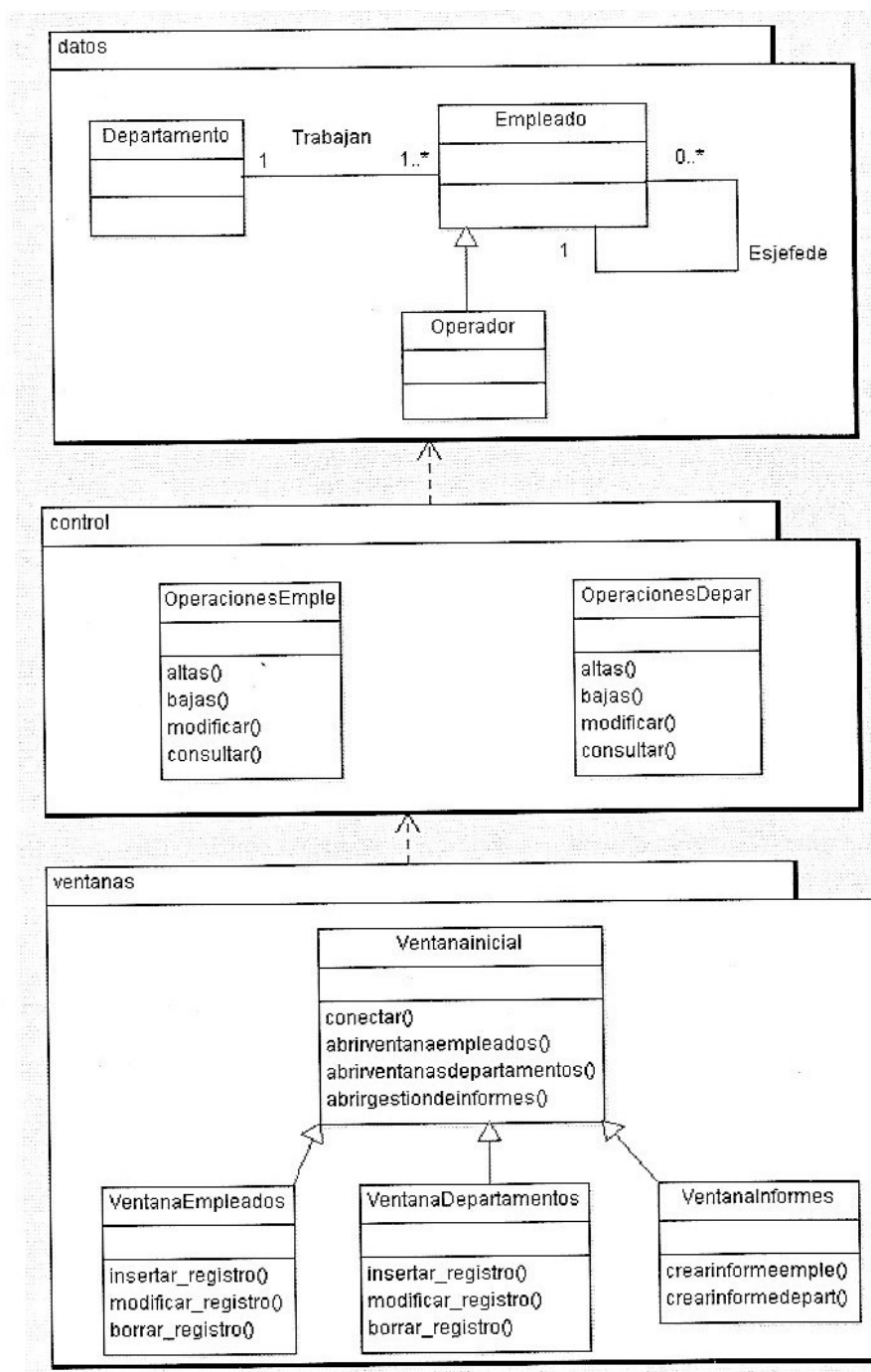
Identificación de clases de diseño: consideramos los tres tipos distintos de clases de análisis para la definición de las clases del ejercicio Entity , Control y Boundary. Así pues se crearán las siguientes clases :

- Clases del tipo Entidad, serán las clases persistentes utilizadas para almacenar la información , asociadas a la BD, consideraremos Departamento, Empleado y Operador que será una especialización de Empleado.
- Clases de Control, su objetivo es controlar las operaciones que se hacen con los datos de la BD (altas, bajas, modificaciones, consultas). Consideramos una clase para las operaciones con datos de empleados y otra para las operaciones con departamentos.
- Clase Interfaz, clases que se diseñarán para la interacción del operador con el sistema, es decir las interfaces gráficas. Se podría contar con una ventana de conexión o inicial, con un menú que de paso al resto de ventanas de la aplicación. Estas ventanas serán una para el mantenimiento de empleados, otra para el mantenimiento de departamento , y otra para la gestión de informes. Estas tres ventanas heredarán la ventana principal.

Identificación de paquetes. Finalmente en este último paso agruparemos las clases en paquetes , se creará un paquete para las clases Entidad, otro para las clases de Control y otro para las clases Interfaz, la idea es separar los datos, la lógica y la interfaz.

Para crear el diagrama primero se crean los paquetes pulsando el botón “Crear paquete” y las clases se van insertando en los paquetes.

- El paquete datos contiene las clases que contienen la información Departamento, Empleado y Operador, el Operador es una especialización de Empleado, por lo que se crea una generalización . No es necesario añadir los atributos y las operaciones , pues no se detallan en el enunciado.
- El paquete control contiene las clases que operarán con los datos de la base de datos, clases responsables de la gestión de información: la clase OperacionesEmple, operará con los datos de empleado y las operaciones serán altas, bajas, modificaciones y consultas, estas operaciones se indican en el enunciado , por lo que es necesario escribirlas. Atributos no se añaden pues no se indican en el enunciado. La clase OperacionesDepar hace lo mismo pero con la clase Departamento.
- El paquete ventanas contiene las clases que permitirán al actor operador interactuar con el sistema, agrupa las clases que manejan las interfaces gráficas de usuario. Consideramos una Ventanainicia en la que se realizará la conexión al sistema y que además contendrá los menús correspondiente para abrir el resto de ventanas. Se indican solo los métodos para conectarse y abrir las ventanas VentanaEmpleados, VentanaDepartamento y VentanaInformes. Se crea una jerarquía para que estas ventanas hereden la ventana principal y sus menús. Las ventanas VentanaEmpleado , VentanaDepartamento y VentanaInforme serán las encargadas de llamar a las operaciones del paquete de control para operar con los datos de la BD , por lo que podemos añadir algunas operaciones como insertar_registro, modificar_registro . borrar_registro , crearinformeemple o crearinformedepart, estas acciones se ejecutarían por ejemplo al pulsar un botón .



Los paquetes están relacionado por una asociación de dependencia. El paquete control depende del paquete datos puesto que sus clases acceden a atributos o invocan operaciones de las clases del paquete datos . Igualmente el paquete ventanas depende del paquete control , pues el paquete ventanas invocan a operaciones del paquete control.