



Creación de Objetos y UML

Loor Ramírez Jair Paul

Universidad de Las Fuerzas Armadas – ESPE

NRC-1323 Programación Orientada a Objetos-POO

Ing. Luis Enrique Jaramillo Montaña

8 de diciembre del 2024

Código ,UML:

Empresa.java

```
package com.mycompany.empresa;  
import java.util.ArrayList;  
import java.util.List;
```

```
/**
```

```
 * @author jar */
```

```
/* Representa a la empresa
```

```
public class Empresa {
```

```
    public String nombre;
```

```
    public List<Empleado> empleados;
```

```
    public Empresa(String nombre) {
```

```
        this.nombre = nombre;
```

```
        this.empleados = new ArrayList<>();
```

```
    }
```

```
    public String getNombre() {
```

```
        return nombre;
```

```
    }  
    public void contratarEmpleado(Empleado empleado) {
```

```
        empleados.add(empleado);
```

```
    }
```

```
    public void mostrarEmpleados() {
```

```
        System.out.println("Empleados en la empresa" + nombre  
            + ".");
```

```
        for(Empleado e: empleados) {
```

```
            System.out.println(e.getDetalles());
```

```
        }
```

```
    }
```

```
}
```

Empleado.java

package com.mycompany.empresa;

/**

* @author jair */

/* clase que representa al empleado de la empresa */

public abstract class Empleado {

public String nombre;

public int edad;

public Empleado(String nombre, int edad) {

this.nombre = nombre;

this.edad = edad;

}

public String getNombre() {

return nombre;

}

public int getEdad() {

return edad;

}

/* para obtener detalles del empleado */

public abstract String getDetalles();

}

Director.java

```
package com.my company. empresa;
```

```
/**  
 * @author javi */
```

```
public class Director extends Empleado {  
    public double bono;
```

```
    public Director(String nombre, int edad, double bono) {  
        super(nombre, edad);  
        this.bono = bono;
```

```
    }
```

```
    public double getBono() {  
        return bono;
```

```
    }
```

```
@Override
```

```
    public String getDetalles() {
```

```
        return "Director:" + getNombre() + ", Edad:" + getEdad() +  
            ", Bono: $" + bono;
```

```
    }
```

```
}
```

Proyecto.java

package com.mycompany.empresa;

/** @author jair */

/* En esta clase, se le asigna un proyecto al trabajador */

public class Proyecto {

public String nombre;

public Trabajador trabajador;

public Proyecto (String nombre, Trabajador trabajador) {

this.nombre = nombre;

this.trabajador = trabajador;

}

public void mostrar Proyecto() {

System.out.println ("Proyecto: " + nombre);

System.out.println ("Asignado a: " + trabajador.getNombre()
"(" + trabajador.getCargo() + ")");

}

}

Main.java

package com.mycompany.empresa;

/**@author jair*/

public class Main {

// Crear una Empresa

Empresa empresa = new Empresa("Tech Solutions");

// Crear empleados

Trabajador t1 = new Trabajador("Juan", 30, "Desarrollador", 2500);

Trabajador t2 = new Trabajador("Ana", 28, "Diseñador", 2300);

Director d1 = new Director("Carlos", 45, 5000);

// Contratar empleados

empresa.contratarEmpleado(t1);

empresa.contratarEmpleado(t2);

empresa.contratarEmpleado(d1);

// Mostrar empleados de la empresa

empresa.mostrarEmpleados();

// Crear proyectos

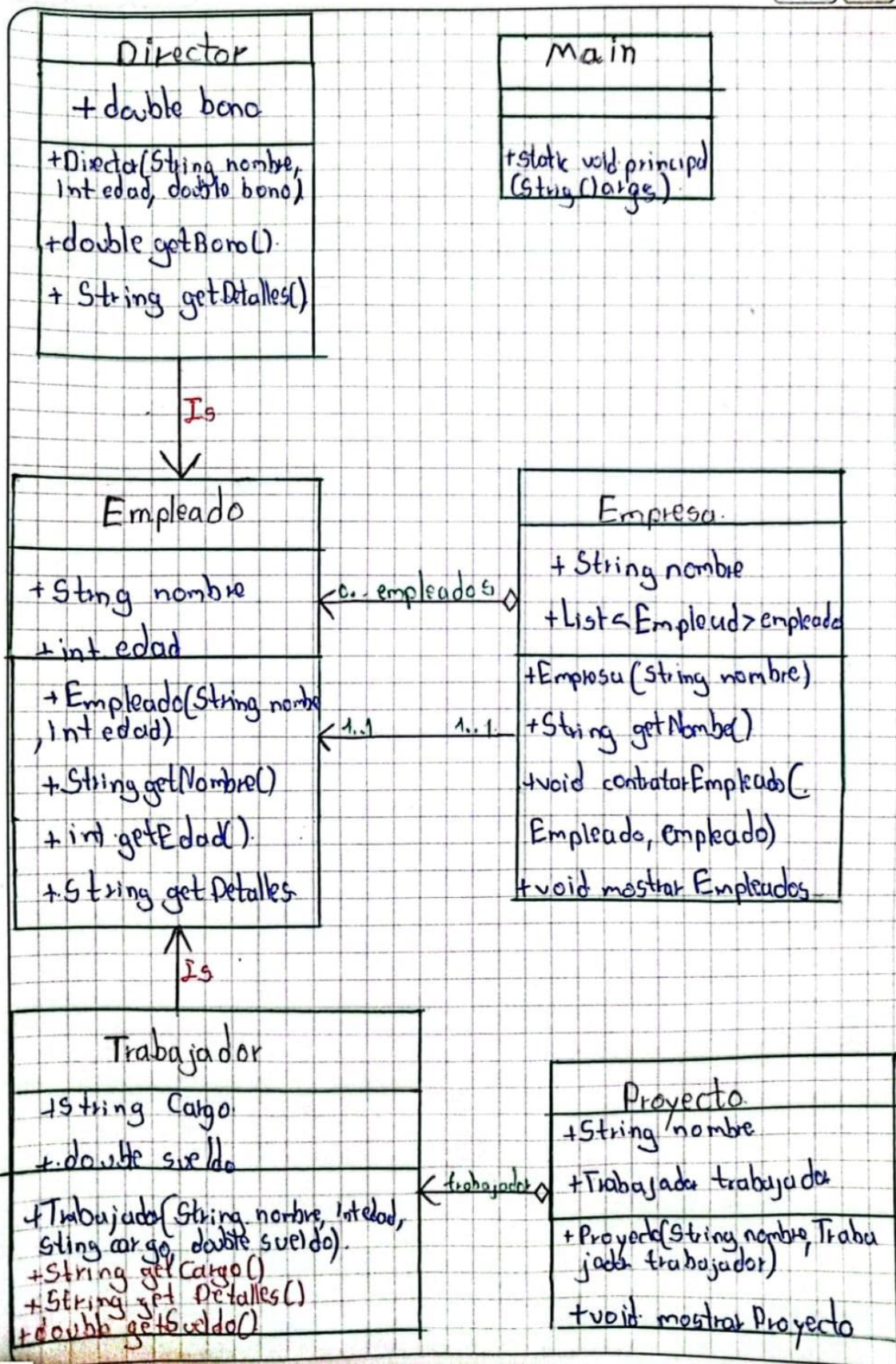
Proyecto p1 = new Proyecto("App Movil", t1);

Proyecto p2 = new Proyecto("Diseño web", t2);

// detalles de los proyectos

p1.mostrarProyecto;

p2.mostrarProyecto;



Resumen de resolución:

Los sistemas de gestión empresarial están diseñados para proporcionar una visión clara y organizada de la estructura y las interacciones de su empresa. Se implementan cinco clases principales:

Empresa: Representa una entidad básica con atributos como nombre y dirección. Administre su lista de empleados y proyectos relacionados.

Empleados: modele los empleados de una empresa utilizando datos como nombre, título y salario.

Director: amplía la clase de Empleado con roles adicionales que le permiten supervisar proyectos y tomar decisiones estratégicas.

Proyectos: identifique los proyectos desarrollados por la empresa con detalles como el nombre del proyecto, el presupuesto y el estado.

Tarea: Representa actividades específicas del proyecto asignadas a los empleados.

El diseño que hice sigue los principios de programación orientada a objetos, enfatizando la encapsulación, la herencia y el polimorfismo. Las clases están relacionadas entre sí mediante composición y asociación, lo que hace que el sistema sea más fácil de gestionar y ampliar. Le permite registrar empleados, asignar tareas y monitorear proyectos, mejorando la organización y eficiencia de su empresa.

Índice

Introducción	10
Objetivo General	10
Objetivos Específicos.....	10
Desarrollo.....	11
Conclusiones	19
Recomendaciones	19
Bibliografía	20

Introducción

Mediante este informe se explica cómo hacer un proyecto en Java usando NetBeans, con programación orientada a objetos (POO) Mediante este proyecto, se crea y muestra un diagrama UML con la herramienta easyUML, así se representa clases, atributos, métodos y relaciones fácilmente. El objetivo principal es entender cómo funciona un sistema básico que organiza una empresa en diferentes grupos como empresa, empleado, trabajador, director y proyecto, mostrando cómo se conectan entre sí las clases y objetos.

Objetivo General

Familiarizarse con el entorno de desarrollo NetBeans para diseñar e implementar sistemas en Java, aplicando conceptos de programación orientada a objetos (POO) y generando un diagrama UML para representar la estructura del sistema.

Objetivos Específicos

- Desarrollar un proyecto en Java que incluya clases, atributos, métodos y relaciones.
- Utilizar NetBeans como herramienta de desarrollo para la creación, depuración y gestión de este proyecto en java.
- Generar diagramas UML mediante easyUML para representar de forma gráfica la estructura del sistema y sus relaciones

Desarrollo

Crear un proyecto en .java para realizar la actividad, para esto.

Abrimos NetBeans.



Ilustración 1

Creamos un nuevo proyecto, elegimos java y damos en next

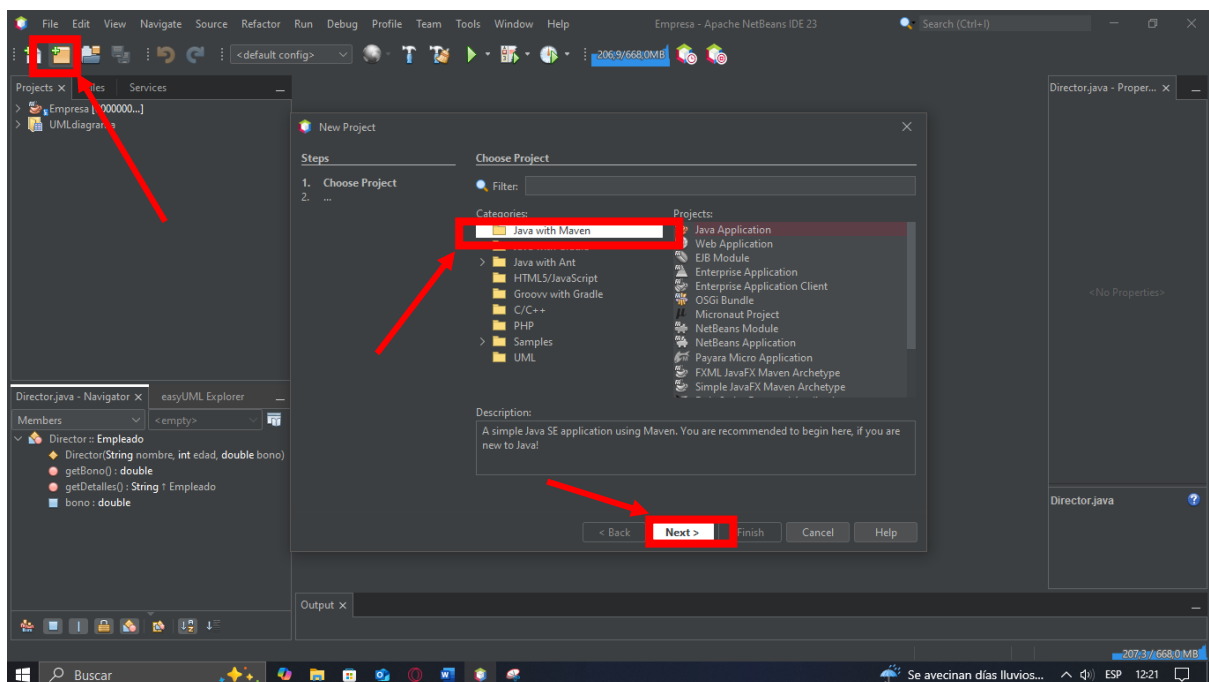


Ilustración 2

Le damos un nombre al proyecto y en finalizar

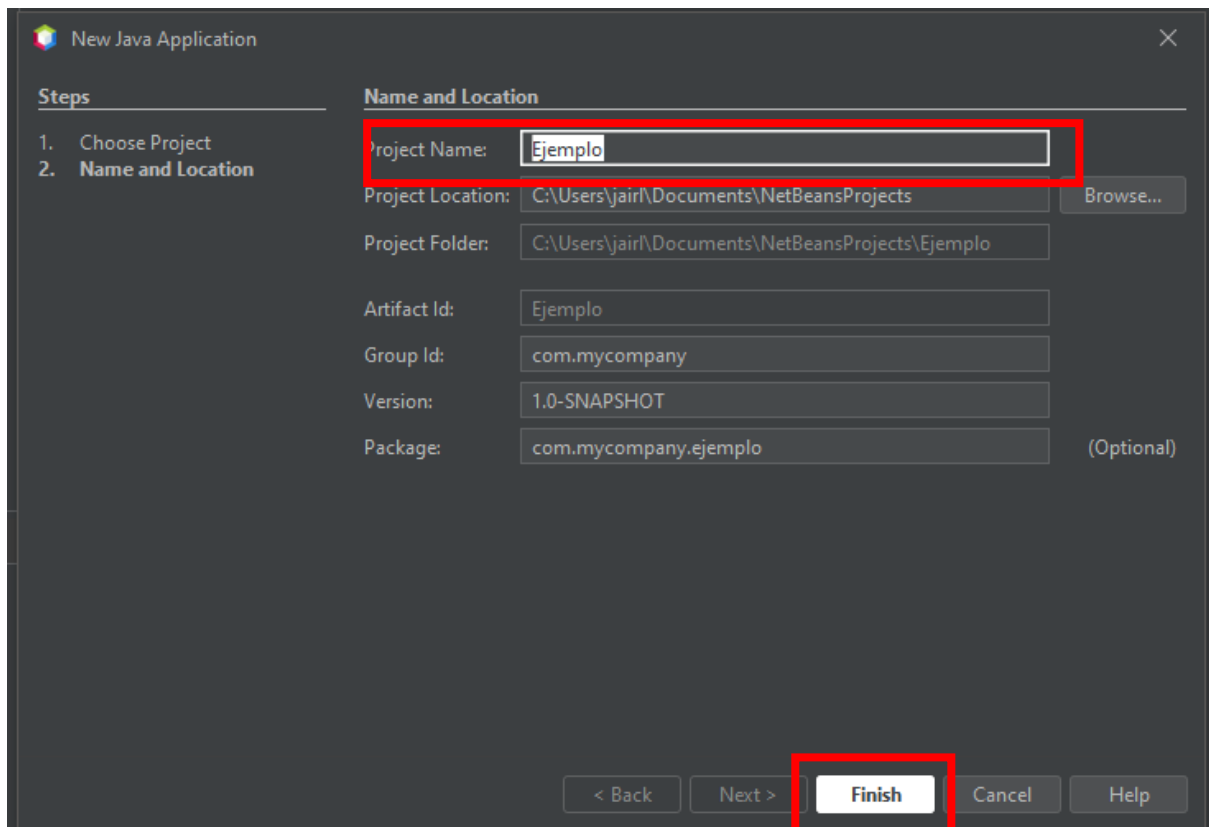


Ilustración 3

Creación de las clases, dentro de com.mycompany.empresa



Ilustración 4

Aquí, creamos las clases que se usaran, con clic derecho y con clic izquierdo elegimos, asignamos un nombre y listo

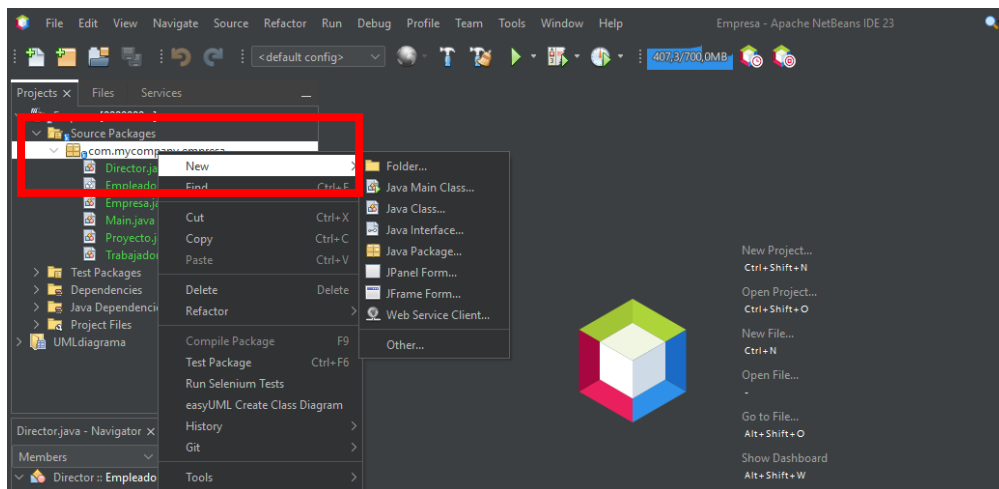


Ilustración 5

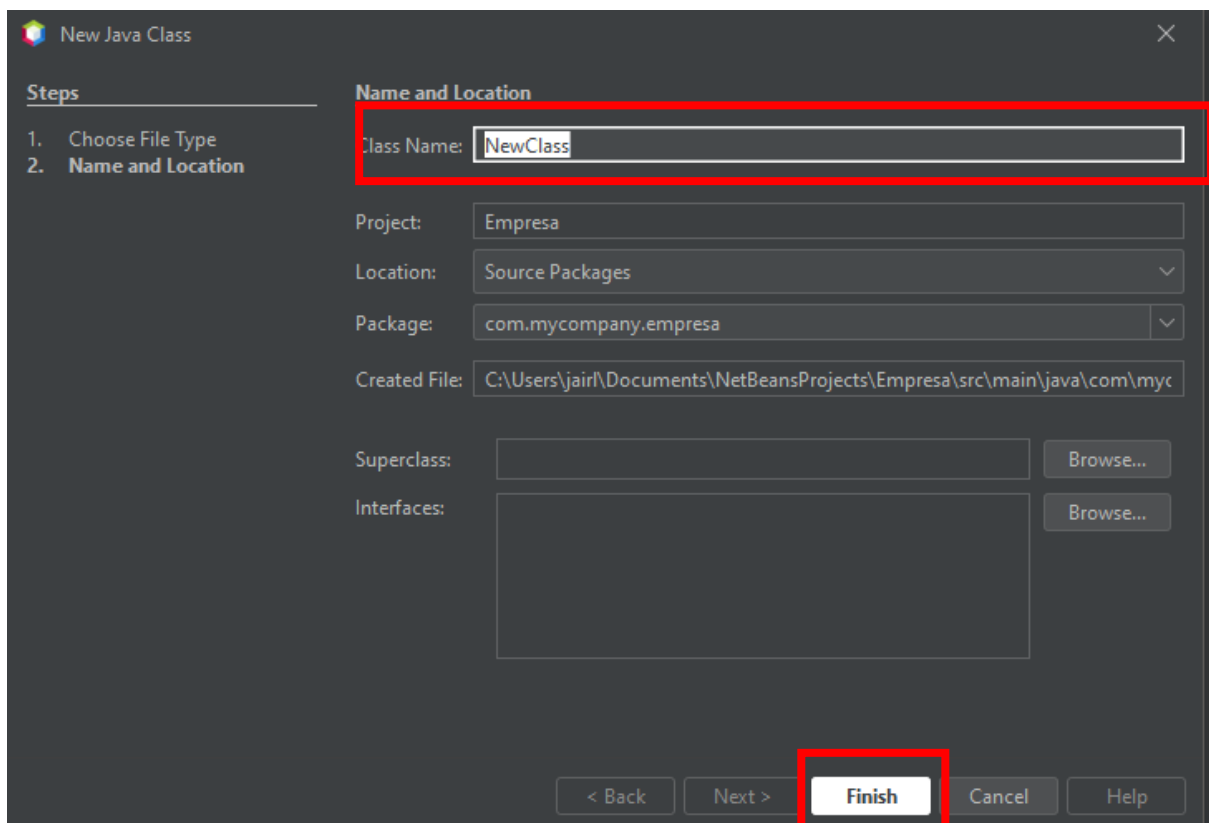


Ilustración 6

Continuamos, ahora en el proyecto se considera realizar 5 objetos con sus atributos, diferentes con su correspondiente diagrama UML, asegurándose de mostrar las relaciones entre ellos.

Mi trabajo se trata de un sistema básico para gestionar una empresa, incluye empleados, un director y proyectos asignados

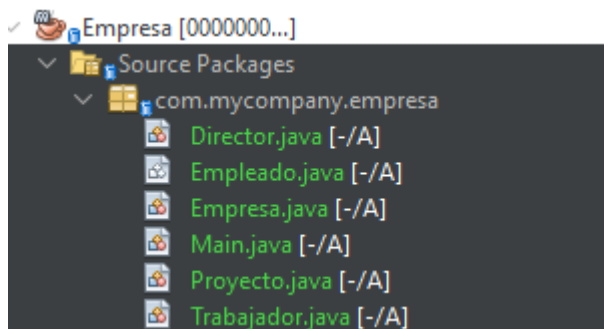


Ilustración 7

Mediante la utilización de conceptos de POO, el sistema fue implementado en Apache NetBeans y su diseño fue representado mediante un diagrama UML generado con la herramienta EasyUML

Diseño

Objetos y relaciones del sistema:

Empresa esta contiene una lista de empleados.


```

1 package com.mycompany.empresa;
2 import java.util.ArrayList;
3 import java.util.List;
4 /**
5  * @author jair */
6  /*Representa a la empresa*/
7  public class Empresa {
8      public String nombre;
9      public List<Empleado> empleados;
10     public Empresa(String nombre) {
11         this.nombre = nombre;
12         this.empleados = new ArrayList<>();
13     }
14     public String getNombre() {
15         return nombre;
16     }
17     public void contratarEmpleado(Empleado empleado) {
18         empleados.add(empleado);
19     }
20     public void mostrarEmpleados() {
21         System.out.println("Empleados en la empresa " + nombre + ":");
22         for (Empleado e : empleados) {
23             System.out.println(e.getDetalles());
24         }
25     }
26 }

```

Ilustración 8

Empleado, esta clase base de la cual derivan los tipos de empleados.

```

1 package com.mycompany.empresa;
2 /**
3  * @author jair */
4  /*
5   * clase que representa al empleado de la empresa
6   */
7  public abstract class Empleado {
8      public String nombre;
9      public int edad;
10     public Empleado(String nombre, int edad) {
11         this.nombre = nombre;
12         this.edad = edad;
13     }
14     public String getNombre() {
15         return nombre;
16     }
17     public int getEdad() {
18         return edad;
19     }
20     //para obtener los detalles del empleado
21     public abstract String getDetalles();
22 }

```

Ilustración 9

Clase Trabajador, esta hereda de `Empleado` y representa un trabajador común con un cargo y sueldo.

```

1 package com.mycompany.empresa;
2 /**
3  * @author jair*/
4  /*representa un trabajador normal de la empresa*/
5  public class Trabajador extends Empleado {
6      public String cargo;
7      public double sueldo;
8      public Trabajador(String nombre, int edad, String cargo, double sueldo) {
9          super(nombre, edad);
10         this.cargo = cargo;
11         this.sueldo = sueldo;
12     }
13     public String getCargo() {
14         return cargo;
15     }
16     public double getSueldo() {
17         return sueldo;
18     }
19     @Override
20     public String getDetalles() {
21         return "Trabajador: " + getNombre() + ", Edad: " + getEdad() +
22             ", Cargo: " + cargo + ", Sueldo: $" + sueldo;
23     }
24 }

```

Ilustración 10

Clase director o CEO, se hereda de empleado y representa al director de la empresa.

```

1 package com.mycompany.empresa;
2 /**
3  * @author jair*/
4  public class Director extends Empleado {
5      public double bono;
6
7      public Director(String nombre, int edad, double bono) {
8          super(nombre, edad);
9          this.bono = bono;
10     }
11     public double getBono() {
12         return bono;
13     }
14     @Override
15     public String getDetalles() {
16         return "Director: " + getNombre() + ", Edad: " + getEdad() +
17             ", Bono: $" + bono;
18     }
19 }

```

Ilustración 11

Clase Proyecto se asocia a un trabajador y representa un proyecto asignado.

```

1 package com.mycompany.empresa;
2 /**
3  *
4  * @author jair
5  */
6 /*En esta clase, se le accina un proyecto al trabajador */
7 public class Proyecto {
8     public String nombre;
9     public Trabajador trabajador;
10
11     public Proyecto(String nombre, Trabajador trabajador) {
12         this.nombre = nombre;
13         this.trabajador = trabajador;
14     }
15
16     public void mostrarProyecto() {
17         System.out.println("Proyecto: " + nombre);
18         System.out.println("Asignado a: " + trabajador.getNombre() +
19             " (Cargo: " + trabajador.getCargo() + ")");
20     }
21 }

```

Ilustración 12

El main o clase principal actúa como el punto de entrada del programa, esta clase crea Empresa, Crea empleados y también crea al CEO, contrata a los empleados y muestra la lista de estos empleados, Asigna y crea proyectos para los Empleados y muestra los proyectos.

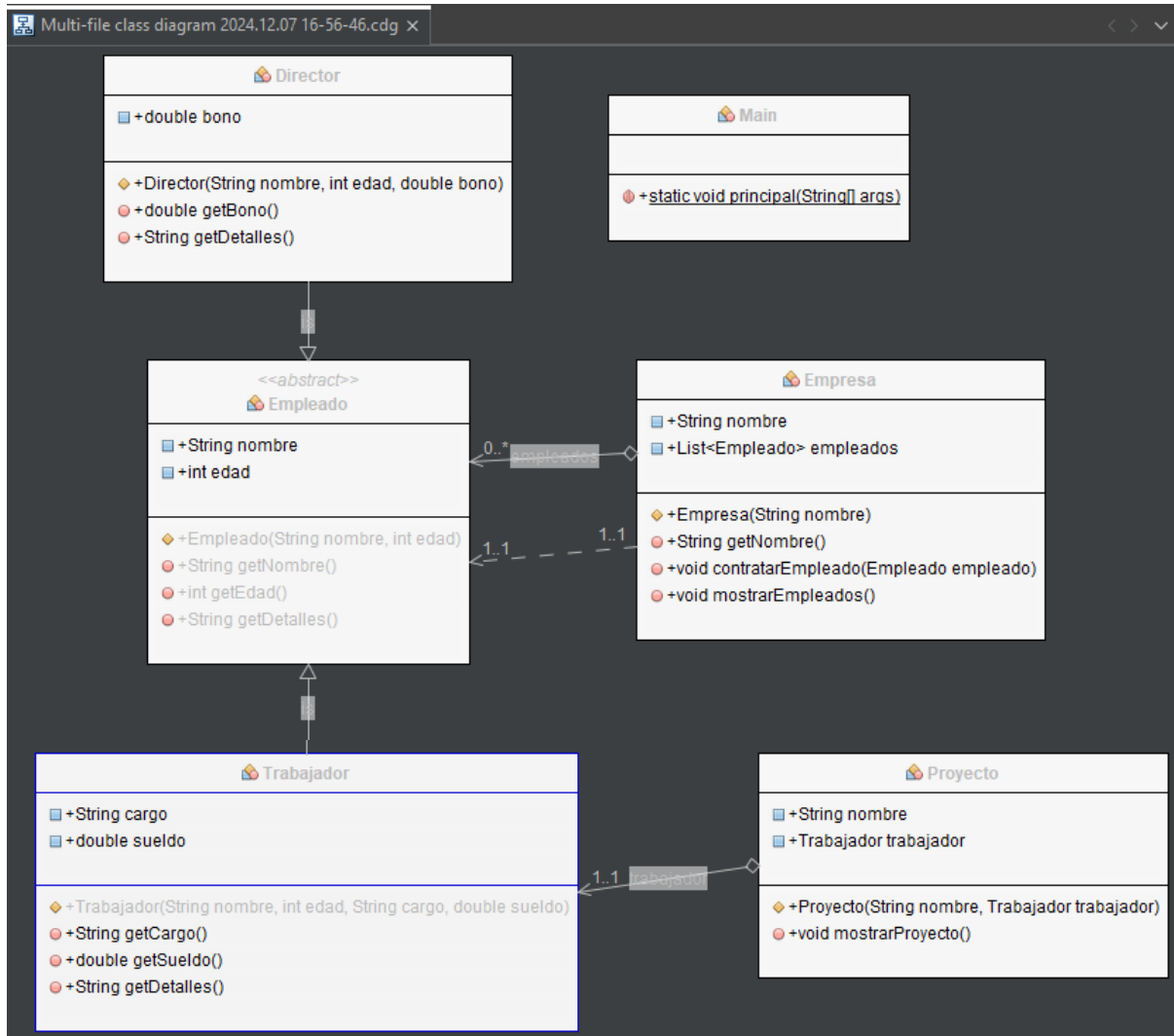
```

1 package com.mycompany.empresa;
2 /** @author jair */
3 public class Main {
4     public static void principal(String[] args) {
5         // Crear una empresa
6         Empresa empresa = new Empresa("Tech Solutions");
7         // Crear empleados
8         Trabajador t1 = new Trabajador("Juan", 30, "Desarrollador", 2500.0);
9         Trabajador t2 = new Trabajador("Ana", 28, "Diseñadora", 2300.0);
10        Director d1 = new Director("Carlos", 45, 5000.0);
11        // Contratar empleados
12        empresa.contratarEmpleado(t1);
13        empresa.contratarEmpleado(t2);
14        empresa.contratarEmpleado(d1);
15        // Mostrar empleados de la empresa
16        empresa.mostrarEmpleados();
17        // Crear proyectos
18        Proyecto p1 = new Proyecto("App Movil", t1);
19        Proyecto p2 = new Proyecto("Diseño Web", t2);
20        // Mostrar detalles de los proyectos
21        p1.mostrarProyecto();
22        p2.mostrarProyecto();
23    }
24 }

```

Ilustración 13

Diagrama en EasyUML, para ello vi un tutorial de YouTube, que explica como extraer el archivo EasyUML y poder ver El UML de un proyecto con clases (Charly Cimino, 2021)



Proyecto y diagrama UML

<https://drive.google.com/file/d/1XpqPqCsgRKM8NcmssyUGBRHw9DwvmSuW/view?usp=sharing>


Conclusiones

- Implementar un sistema basado en programación orientada a objetos (POO) permite estructurar y organizar el código de manera eficaz, facilitando la reutilización y el mantenimiento de dicho proyecto
- El sistema de gestión empresarial demostró cómo las relaciones entre clases (Objetos y atributos) pueden simular escenarios del mundo real.
- La generación de diagramas UML en Apache NetBeans es una herramienta importante para visualizar el diseño del sistema antes de la implementación

Recomendaciones

- Definir los requerimientos del sistema y diseñar el diagrama UML antes de iniciar la codificación.
- Al crear sistemas en el Apache NetBeans en .java, se debe seguir buenas prácticas de programación, como dar nombres descriptivos a las clases y métodos, mantener el código modular y utilizar comentarios para facilitar la comprensión y el mantenimiento.

Bibliografía

Charly Cimino. (2021, 8 noviembre). *Diagrama UML*  *Automático en ECLIPSE con*

ObjectAID [Vídeo]. YouTube. <https://www.youtube.com/watch?v=WTEDh-9HU20>

pildorasinformaticas. (2024, 17 enero). *Curso UML. Ejemplo práctico II. Vídeo 19* [Vídeo].

YouTube. <https://www.youtube.com/watch?v=kOzqw3hGwAM>

Diagrama UML: Qué es, cómo hacerlo y ejemplos / Miro. (s. f.). miro.com.

<https://miro.com/es/diagrama/que-es-diagrama-uml/>

Edraw. (2022, 13 junio). *Ejemplos de diagrama UML*. <https://www.edrawsoft.com/es/uml->

[diagrama-](https://www.edrawsoft.com/es/uml-diagrama-)

[examples.html?srsltid=AfmBOooQeR_zahF7GaQunaCIBQBsvPiOJ6b1MOcQQPbjG](https://www.edrawsoft.com/es/uml-examples.html?srsltid=AfmBOooQeR_zahF7GaQunaCIBQBsvPiOJ6b1MOcQQPbjG)

[jsbP5M9m0Y0](https://www.edrawsoft.com/es/uml-examples.html?srsltid=AfmBOooQeR_zahF7GaQunaCIBQBsvPiOJ6b1MOcQQPbjG)

Límaco, D. S. (2016, 13 abril). *UML en NetBeans*. *apuntesdejava*.

https://www.apuntesdejava.com/2015/12/uml-en-netbeans.html#google_vignette