Informe sobre la Instalación y Uso de Umbrello para la Creación de Diagramas de Clases

1. Introducción

En este trabajo se aborda el proceso de instalación y uso de la herramienta Umbrello, así como el desarrollo de un diagrama de clases orientado a la programación orientada a objetos (POO) para un sistema específico. La herramienta Umbrello, un software de modelado UML, permite generar diagramas visuales que facilitan la comprensión de la estructura y relaciones entre las clases que componen el sistema. En particular, se ha utilizado para representar las clases involucradas en un modelo simplificado que involucra entidades como *Empleado*, *Departamento* y *Empresa*.

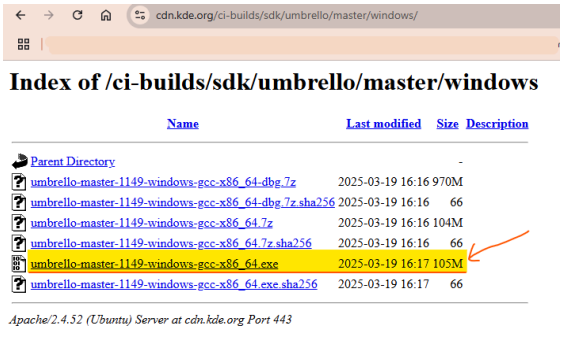
A lo largo de este trabajo, se realiza una análisis detallado del proceso de instalación y configuración de Umbrello, el desarrollo del diagrama de clases que refleja las relaciones entre las entidades del sistema, y finalmente, la generación de código desde el diagrama realizado. Este ejercicio tiene como objetivo proporcionar una base sólida para la comprensión de los principios fundamentales de la POO, a la vez que demuestra cómo una herramienta UML puede ser utilizada para simplificar la creación de modelos de sistemas complejos.

El diagrama de clases elaborado destaca las relaciones entre las clases *Empleado*, *Departamento* y *Empresa*, utilizando los principios de herencia, agregación y asociación, que son esenciales en la programación orientada a objetos. Además, se generó código en Java a partir del diagrama, permitiendo ver cómo la estructura visual puede traducirse en código funcional.

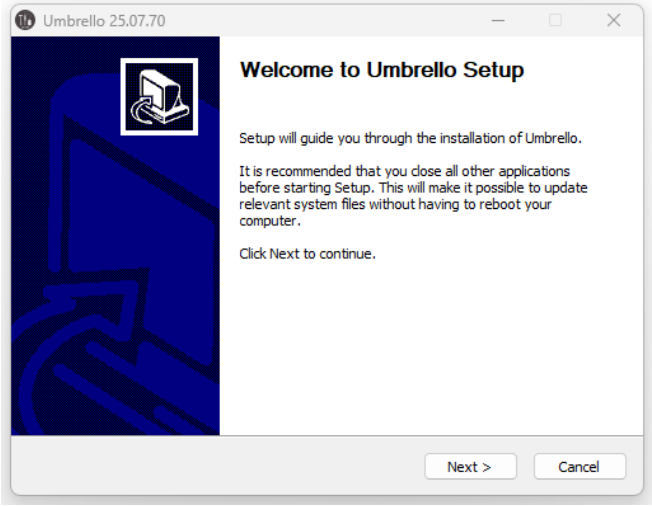
2. Instalación y Configuración de Umbrello

Paso 1: Descargar el software Umbrello desde la página oficial de KDE.

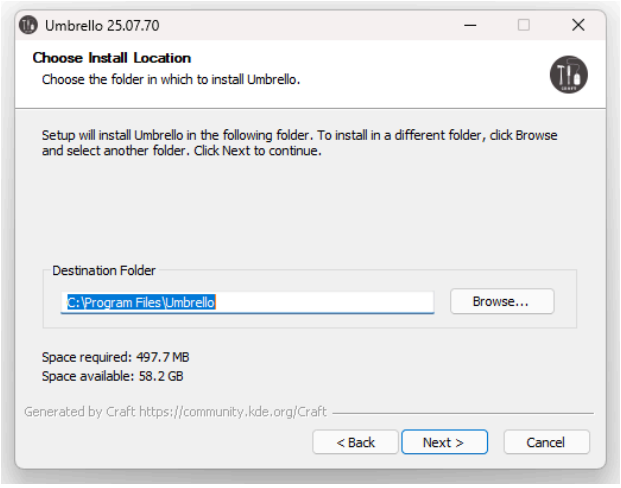
- URL de descarga: https://cdn.kde.org/ci-builds/sdk/umbrello/master/windows/



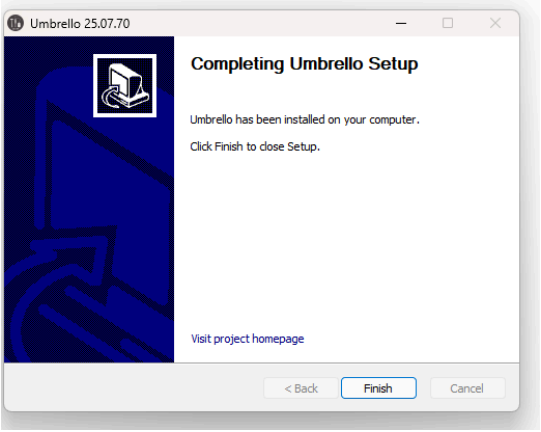
Paso 2: Ejecutar el instalador y seguir los pasos del asistente de instalación.

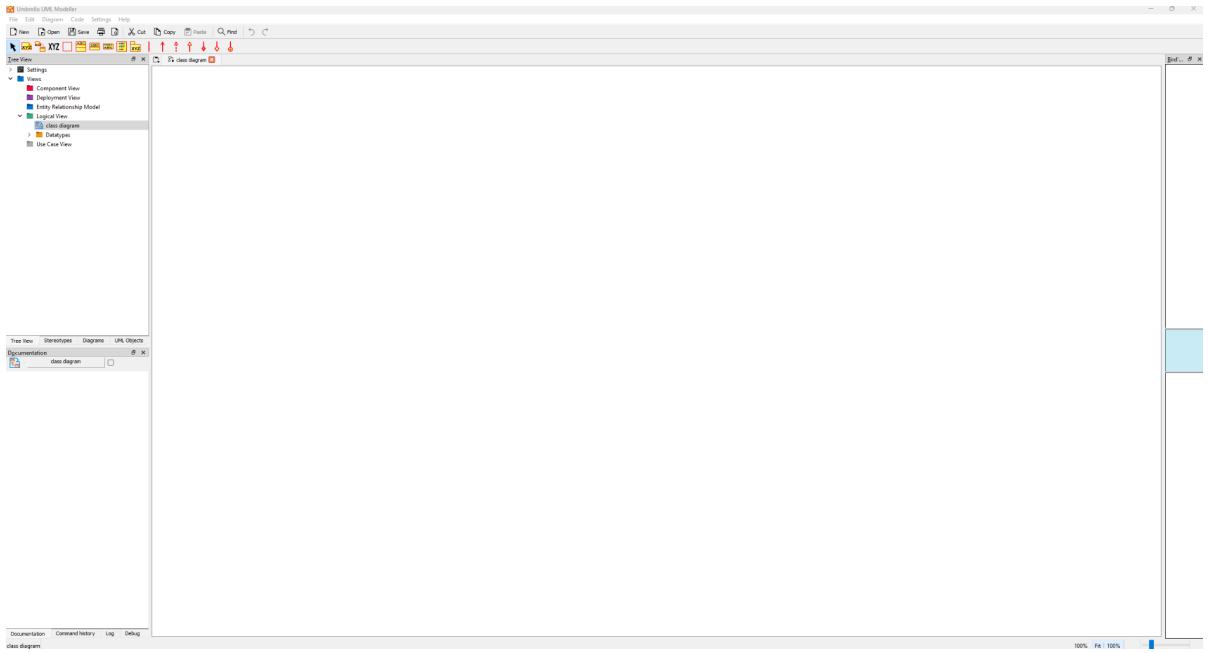


- Seleccionar el directorio de instalación.



- Finalizar la instalación y ejecutar Umbrello para verificar que funciona correctamente.



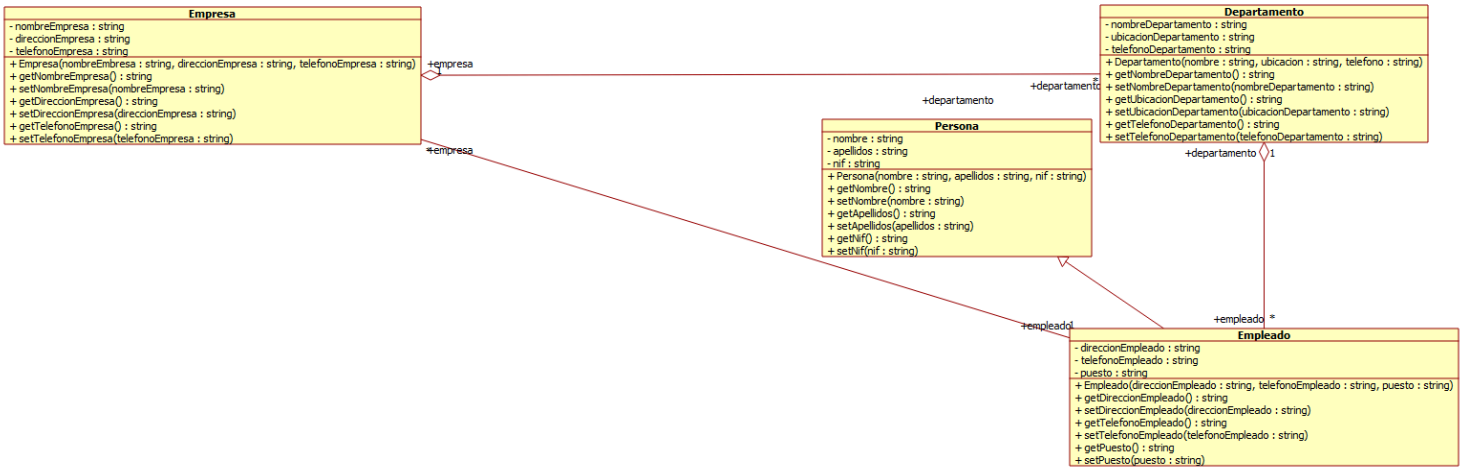


3. Creación del Diagrama de Clases

El diagrama de clases se realizó siguiendo la guía proporcionada para ArgoUML, pero adaptado a Umbrello. Se utilizó la página web: http://jbgarcia.webs.uvigo.es/asignaturas/TO/usoArgoUML/index.html

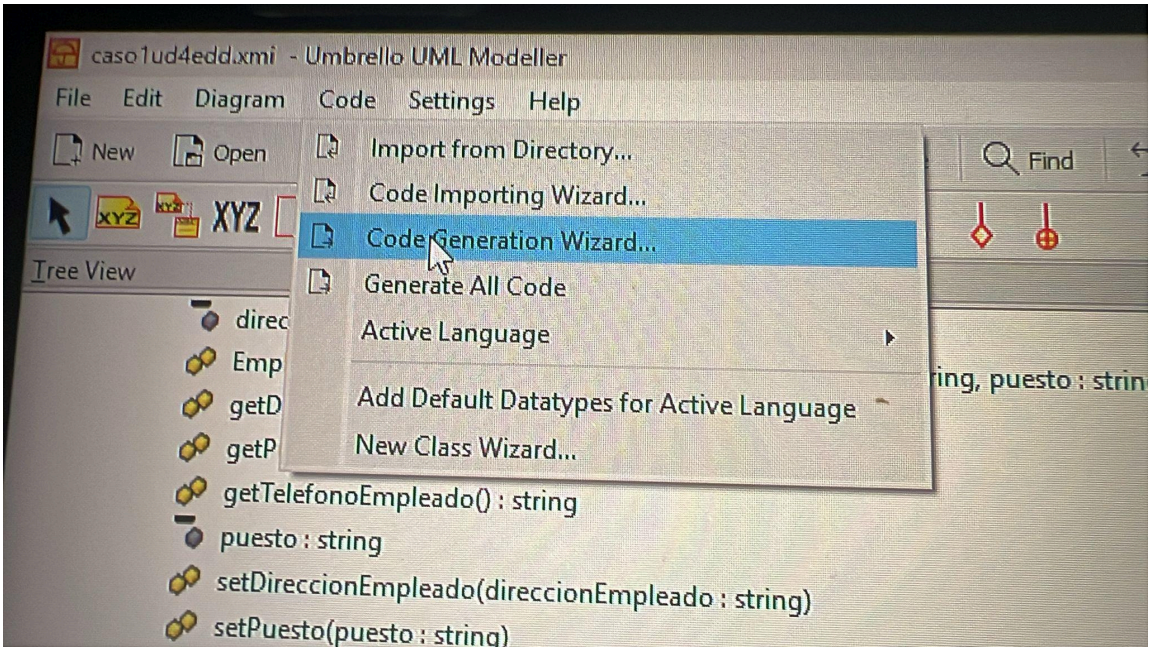
Elementos del diagrama:

- **Clases:** Se crearon las clases principales con sus atributos y métodos.
- **Relaciones:** Se implementaron las relaciones de herencia, asociación y agregación según la estructura indicada en la guía.

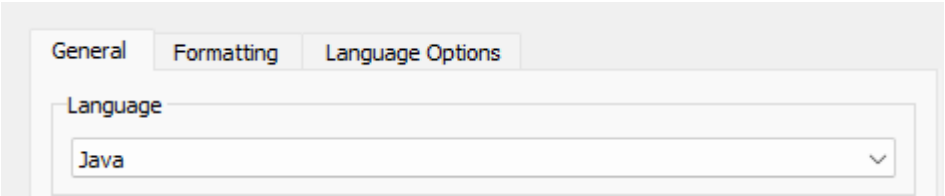


4. Generación de Código Fuente

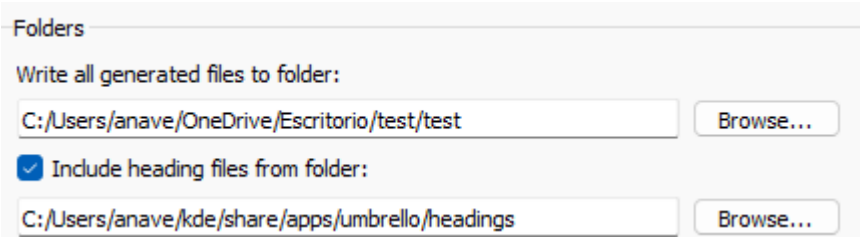
Paso 1: Seleccionar la opción de generación de código en Umbrello.



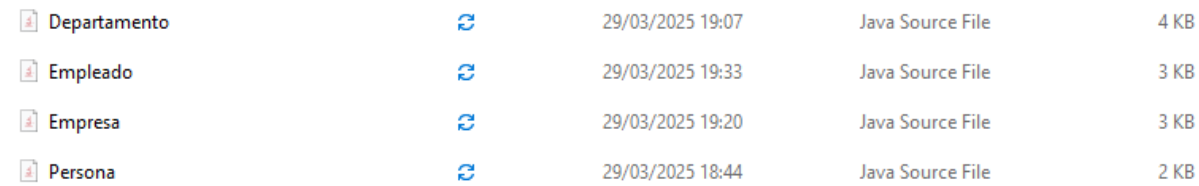
Paso 2: Elegir el lenguaje de programación (en este caso, Java).



Paso 3: Indicar la ruta donde se guardará el código generado.



Paso 4: Verificar que los archivos .java se generen correctamente.



4.1. Código fuente generado por clase

4.1.1. *Clase Persona*

```java
/**
 * Class Persona
 */
public class Persona {

  //
  // Fields
  //

  private String nombre;
  private String apellidos;
  private String nif;

  //
  // Constructors
  //
  public Persona () { };

  //
  // Methods
  //

  //
   // Accessor methods
  //

  /**
   * Set the value of nombre
   * @param newVar the new value of nombre
   */
  public void setNombre (String newVar) {
    nombre = newVar;
  }

  /**
   * Get the value of nombre
   * @return the value of nombre
   */
  public String getNombre () {
    return nombre;
  }

  /**
    * Set the value of apellidos
   * @param newVar the new value of apellidos
   */
  public void setApellidos (String newVar) {
     apellidos = newVar;
   }
```

```java
/**
 * Get the value of apellidos
 * @return the value of apellidos
 */
public String getApellidos () {
  return apellidos;
}

/**
 * Set the value of nif
 * @param newVar the new value of nif
 */
public void setNif (String newVar) {
  nif = newVar;
}

/**
 * Get the value of nif
 * @return the value of nif
 */
public String getNif () {
  return nif;
}

//
// Other methods
//

/**
  * @return       String
 */
public String getNombre()
{
}


/**
 * @param        nombre
 */
public void setNombre(String nombre)
{
}


/**
 * @return       String
 */
 public String getApellidos()
{
}

 /**
 * @param        apellidos
 */
public void setApellidos(String apellidos)
{
```

```
  }

  /**
   * @return       String
   */
  public String getNif()
  {
  }


  /**
   * @param        nif
   */
  public void setNif(String nif)
  {
  }

}
```

4.1.2 *Clase Empleado*

```
import java.util.*;


/**
 * Class Empleado
 */
public class Empleado extends Persona {

  //
  // Fields
  //

  private String direccionEmpleado;
  private String telefonoEmpleado;
  private String puesto;

  public Vector empresaVector = new Vector();

  public Vector empresaVector = new Vector();
  // asociaci?n entre empleado y departamento


  public Departamento m_departamento;

  public Departamento m_departamento;

  //
  // Constructors
  //
  public Empleado () { };

  //
  // Methods
  //
```

```
//
// Accessor methods
//

/**
 * Set the value of direccionEmpleado
 * @param newVar the new value of direccionEmpleado
 */
public void setDireccionEmpleado (String newVar) {
  direccionEmpleado = newVar;
}

/**
 * Get the value of direccionEmpleado
 * @return the value of direccionEmpleado
 */
public String getDireccionEmpleado () {
  return direccionEmpleado;
}

/**
 * Set the value of telefonoEmpleado
 * @param newVar the new value of telefonoEmpleado
 */
public void setTelefonoEmpleado (String newVar) {
  telefonoEmpleado = newVar;
}

/**
 * Get the value of telefonoEmpleado
 * @return the value of telefonoEmpleado
 */
public String getTelefonoEmpleado () {
  return telefonoEmpleado;
}

/**
 * Set the value of puesto
 * @param newVar the new value of puesto
 */
public void setPuesto (String newVar) {
  puesto = newVar;
}

/**
 * Get the value of puesto
 * @return the value of puesto
 */
public String getPuesto () {
  return puesto;
}

/**
 * Add a Empresa object to the empresaVector List
```

```
 */
public void addEmpresa (Empresa new_object) {
  empresaVector.add(new_object);
}

/**
 * Remove a Empresa object from empresaVector List
 */
public void removeEmpresa (Empresa new_object)
{
  empresaVector.remove(new_object);
}

/**
 * Get the List of Empresa objects held by empresaVector
 * @return List of Empresa objects held by empresaVector
 */
public List getEmpresaList () {
  return (List) empresaVector;
}


/**
 * Add a Empresa object to the empresaVector List
 */
public void addEmpresa (Empresa new_object) {
  empresaVector.add(new_object);
}

/**
 * Remove a Empresa object from empresaVector List
 */
public void removeEmpresa (Empresa new_object)
{
  empresaVector.remove(new_object);
}

/**
 * Get the List of Empresa objects held by empresaVector
 * @return List of Empresa objects held by empresaVector
 */
public List getEmpresaList () {
  return (List) empresaVector;
}


/**
 * Set the value of m_departamento
 * @param newVar the new value of m_departamento
 */
public void setDepartamento (Departamento newVar) {
  m_departamento = newVar;
}

/**
 * Get the value of m_departamento
 * @return the value of m_departamento
```

```
 */
public Departamento getDepartamento () {
  return m_departamento;
}

/**
 * Set the value of m_departamento
 * @param newVar the new value of m_departamento
 */
public void setDepartamento (Departamento newVar) {
  m_departamento = newVar;
}

/**
 * Get the value of m_departamento
 * @return the value of m_departamento
 */
public Departamento getDepartamento () {
  return m_departamento;
}

//
// Other methods
//

/**
 * @param        direccionEmpleado
 * @param        telefonoEmpleado
 * @param        puesto
 */
public void Empleado(String direccionEmpleado, String telefonoEmpleado, String puesto)
{
}


/**
 * @return       String
 */
public String getDireccionEmpleado()
{
}


/**
 * @param        direccionEmpleado
 */
public void setDireccionEmpleado(String direccionEmpleado)
{
}


/**
 * @return       String
 */
public String getTelefonoEmpleado()
{
}
```

```
  /**
   * @param        telefonoEmpleado
   */
  public void setTelefonoEmpleado(String telefonoEmpleado)
  {
  }


  /**
   * @return       String
   */
  public String getPuesto()
  {
  }


  /**
   * @param        puesto
   */
  public void setPuesto(String puesto)
  {
  }


}
```

4.1.3. *Clase Empresa*

```
import java.util.*;


/**
 * Class Empresa
 */
public class Empresa {

  //
  // Fields
  //


   private String nombreEmpresa;
  private String direccionEmpresa;
  private String telefonoEmpresa;

  public Empleado m_empleado;

  public Empleado m_empleado;

  public Departamento m_departamento;


   public Vector departamentoVector = new Vector();
```

```
public Empresa m_empresa;

//
// Constructors
//
public Empresa () { };

//
// Methods
//


//
// Accessor methods
//

/**
 * Set the value of nombreEmpresa
 * @param newVar the new value of nombreEmpresa
 */
public void setNombreEmpresa (String newVar) {
  nombreEmpresa = newVar;
}

/**
 * Get the value of nombreEmpresa
 * @return the value of nombreEmpresa
  */
public String getNombreEmpresa () {
  return nombreEmpresa;
}

/**
 * Set the value of direccionEmpresa
 * @param newVar the new value of direccionEmpresa
 */
public void setDireccionEmpresa (String newVar) {
  direccionEmpresa = newVar;
}


 /**
 * Get the value of direccionEmpresa
 * @return the value of direccionEmpresa
 */
public String getDireccionEmpresa () {
  return direccionEmpresa;
}

/**
 * Set the value of telefonoEmpresa
 * @param newVar the new value of telefonoEmpresa
 */
public void setTelefonoEmpresa (String newVar) {
  telefonoEmpresa = newVar;
}
```

```java
/**
 * Get the value of telefonoEmpresa
 * @return the value of telefonoEmpresa
 */
public String getTelefonoEmpresa () {
  return telefonoEmpresa;
}

/**
 * Set the value of m_empleado
 * @param newVar the new value of m_empleado
 */
public void setEmpleado (Empleado newVar) {
  m_empleado = newVar;
}

/**
 * Get the value of m_empleado
 * @return the value of m_empleado
 */
public Empleado getEmpleado () {
  return m_empleado;
}

/**
 * Set the value of m_empleado
 * @param newVar the new value of m_empleado
 */
public void setEmpleado (Empleado newVar) {
  m_empleado = newVar;
}

/**
 * Get the value of m_empleado
 * @return the value of m_empleado
 */
public Empleado getEmpleado () {
  return m_empleado;
}

/**
  * Set the value of m_departamento
 * @param newVar the new value of m_departamento
  */
public void setDepartamento (Departamento newVar) {
  m_departamento = newVar;
}

/**
 * Get the value of m_departamento
 * @return the value of m_departamento
 */
public Departamento getDepartamento () {
  return m_departamento;
}
```

```
  /**
   * Add a Departamento object to the departamentoVector List
   */
  public void addDepartamento (Departamento new_object) {
    departamentoVector.add(new_object);
  }

  /**
   * Remove a Departamento object from departamentoVector List
   */
  public void removeDepartamento (Departamento new_object)
  {
    departamentoVector.remove(new_object);
  }

  /**
   * Get the List of Departamento objects held by departamentoVector
   * @return List of Departamento objects held by departamentoVector
   */
  public List getDepartamentoList () {
    return (List) departamentoVector;
  }


  //
  // Other methods
  //

  /**
   * @param         nombreEmbresa
   * @param         direccionEmpresa
   * @param         telefonoEmpresa
   */
  public void Empresa(String nombreEmbresa, String direccionEmpresa, String
telefonoEmpresa)
   {
   }



   /**
   * @return        String
   */
  public String getNombreEmpresa()
  {
  }


  /**
   * @param         nombreEmpresa
   */
  public void setNombreEmpresa(String nombreEmpresa)
  {
  }

  /**
```

```
   * @return       String
   */
  public String getDireccionEmpresa()
  {
  }


  /**
   * @param        direccionEmpresa
   */
  public void setDireccionEmpresa(String direccionEmpresa)
  {
  }


  /**
   * @return       String
   */
  public String getTelefonoEmpresa()
  {
  }


  /**
   * @param        telefonoEmpresa
   */
  public void setTelefonoEmpresa(String telefonoEmpresa)
  {
  }

}
```

4.1.4. *Clase Departamento*

```
import java.util.*;


/**
 * Class Departamento
 */
public class Departamento {

  //
  // Fields
  //

  private String nombreDepartamento;
  private String ubicacionDepartamento;
  private String telefonoDepartamento;

  public Vector empresaVector = new Vector();

  public Vector departamentoVector = new Vector();

  public Empresa m_empresa;
```

```
public Vector empleadoVector = new Vector();

public Departamento m_departamento;

//
// Constructors
//
public Departamento () { };

//
// Methods
//


//
// Accessor methods
//

/**
 * Set the value of nombreDepartamento
 * @param newVar the new value of nombreDepartamento
 */
public void setNombreDepartamento (String newVar) {
  nombreDepartamento = newVar;
 }

/**
 * Get the value of nombreDepartamento
 * @return the value of nombreDepartamento
 */
public String getNombreDepartamento () {
  return nombreDepartamento;
}

/**
 * Set the value of ubicacionDepartamento
 * @param newVar the new value of ubicacionDepartamento
 */
public void setUbicacionDepartamento (String newVar) {
  ubicacionDepartamento = newVar;
}

/**
 * Get the value of ubicacionDepartamento
 * @return the value of ubicacionDepartamento
 */
public String getUbicacionDepartamento () {
  return ubicacionDepartamento;
}

/**
 * Set the value of telefonoDepartamento
 * @param newVar the new value of telefonoDepartamento
 */
public void setTelefonoDepartamento (String newVar) {
```

```
    telefonoDepartamento = newVar;
  }

  /**
   * Get the value of telefonoDepartamento
   * @return the value of telefonoDepartamento
   */
  public String getTelefonoDepartamento () {
    return telefonoDepartamento;
  }


   /**
   * Add a Empresa object to the empresaVector List
   */
  public void addEmpresa (Empresa new_object) {
    empresaVector.add(new_object);
  }

  /**
   * Remove a Empresa object from empresaVector List
   */
  public void removeEmpresa (Empresa new_object)
  {
    empresaVector.remove(new_object);
   }

  /**
   * Get the List of Empresa objects held by empresaVector
   * @return List of Empresa objects held by empresaVector
   */
  public List getEmpresaList () {
    return (List) empresaVector;
  }


  /**
   * Set the value of m_empresa
   * @param newVar the new value of m_empresa
   */
  public void setEmpresa (Empresa newVar) {
    m_empresa = newVar;
  }


   /**
   * Get the value of m_empresa
   * @return the value of m_empresa
   */
  public Empresa getEmpresa () {
    return m_empresa;
  }

  /**
   * Add a Empleado object to the empleadoVector List
   */
  public void addEmpleado (Empleado new_object) {
```

```
  empleadoVector.add(new_object);
}

/**
 * Remove a Empleado object from empleadoVector List
 */
public void removeEmpleado (Empleado new_object)
{
  empleadoVector.remove(new_object);
}

/**
 * Get the List of Empleado objects held by empleadoVector
 * @return List of Empleado objects held by empleadoVector
 */
public List getEmpleadoList () {
  return (List) empleadoVector;
}

//
// Other methods
//

/**
 * @param        nombre
 * @param        ubicacion
 * @param        telefono
 */
public void Departamento(String nombre, String ubicacion, String telefono)
{
}

/**
 * @return       String
 */
public String getNombreDepartamento()
{
}


/**
 * @param        nombreDepartamento
 */
public void setNombreDepartamento(String nombreDepartamento)
{
}

/**
 * @return       String
 */
public String getUbicacionDepartamento()
{
}

/**
 * @param        ubicacionDepartamento
 */
```

```
  public void setUbicacionDepartamento(String ubicacionDepartamento)
  {
  }

  /**
   * @return       String
   */
  public String getTelefonoDepartamento()
  {
  }

  /**
   * @param        telefonoDepartamento
   */
  public void setTelefonoDepartamento(String telefonoDepartamento)
  {
  }

}
```

4.2. Código fuente corregido por clase

4.2.1. *Clase Persona*

```
/**
 * Class Persona
 */
public class Persona {

  //
  // Fields
  //

  private String nombre;
  private String apellidos;
  private String nif;

  //
  // Constructors
  //
  public Persona () { };

  //
  // Methods
  //
  /**
   * Set the value of nombre
   * @param newVar the new value of nombre
   */
  public void setNombre (String newVar) {
    nombre = newVar;
  }
```

```
  /**
   * Get the value of nombre
   * @return the value of nombre
   */
  public String getNombre () {
    return nombre;
  }

  /**
   * Set the value of apellidos
   * @param newVar the new value of apellidos
   */
  public void setApellidos (String newVar) {
    apellidos = newVar;
  }

  /**
   * Get the value of apellidos
   * @return the value of apellidos
   */
  public String getApellidos () {
    return apellidos;
  }

  /**
   * Set the value of nif
   * @param newVar the new value of nif
   */
  public void setNif (String newVar) {
    nif = newVar;
  }

  /**
   * Get the value of nif
   * @return the value of nif
   */
  public String getNif () {
    return nif;
  }
}
```

4.2.2. *Clase Empleado*

```
import java.util.*;


/**
 * Class Empleado
 */
public class Empleado extends Persona {

  //
  // Fields
  //

  private String direccionEmpleado;
```

```
private String telefonoEmpleado;
private String puesto;

public ArrayList<Empresa> empresaList = new ArrayList<Empresa>();
// asociacion entre empleado y departamento

public Departamento m_departamento;

//
// Constructors
//
public Empleado () { };

//
// Methods
//

//
// Accessor methods
//

/**
 * Set the value of direccionEmpleado
 * @param newVar the new value of direccionEmpleado
 */
public void setDireccionEmpleado (String newVar) {
  direccionEmpleado = newVar;
}

/**
 * Get the value of direccionEmpleado
 * @return the value of direccionEmpleado
 */
public String getDireccionEmpleado () {
  return direccionEmpleado;
}

/**
 * Set the value of telefonoEmpleado
 * @param newVar the new value of telefonoEmpleado
 */
public void setTelefonoEmpleado (String newVar) {
  telefonoEmpleado = newVar;
}

/**
 * Get the value of telefonoEmpleado
 * @return the value of telefonoEmpleado
 */
public String getTelefonoEmpleado () {
  return telefonoEmpleado;
}

/**
 * Set the value of puesto
 * @param newVar the new value of puesto
```

```
   */
  public void setPuesto (String newVar) {
    puesto = newVar;
  }

  /**
   * Get the value of puesto
   * @return the value of puesto
   */
  public String getPuesto () {
    return puesto;
  }


  /**
   * Add a Empresa object to the empresaList List
   */
  public void addEmpresa (Empresa new_object) {
    empresaList.add(new_object); // Usando ArrayList
  }

  /**
   * Remove a Empresa object from empresaList List
   */
  public void removeEmpresa (Empresa new_object)
  {
    empresaList.remove(new_object); // Usando ArrayList
  }

  /**
   * Get the List of Empresa objects held by empresaList
   * @return List of Empresa objects held by empresaList
   */
  public List<Empresa> getEmpresaList () {
    return empresaList;  // Usando ArrayList
  }

  /**
   * Set the value of m_departamento
   * @param newVar the new value of m_departamento
   */
  public void setDepartamento (Departamento newVar) {
    m_departamento = newVar;
  }

  /**
   * Get the value of m_departamento
   * @return the value of m_departamento
    */
  public Departamento getDepartamento () {
    return m_departamento;
  }
}
```

4.2.3. *Clase Empresa*

```
ArrayList
import java.util.*;


/**
 * Class Empresa
 */
public class Empresa {

  //
  // Fields
  //

  private String nombreEmpresa;
  private String direccionEmpresa;
  private String telefonoEmpresa;

  // Asociación entre empresa y empleados
  public Empleado m_empleado;


  // Asociación entre empresa y departamento
  public Departamento m_departamento;

  // Lista de departamentos asociados a la empresa
  private ArrayList<Departamento> departamentoList = new ArrayList<Departamento>();

  //
  // Constructors
  //
  public Empresa () { };

  //
  // Methods
  //

  //
  // Accessor methods
  //

  /**
   * Set the value of nombreEmpresa
   * @param newVar the new value of nombreEmpresa
   */
  public void setNombreEmpresa (String newVar) {
    nombreEmpresa = newVar;
  }

  /**
   * Get the value of nombreEmpresa
   * @return the value of nombreEmpresa
   */
  public String getNombreEmpresa () {
    return nombreEmpresa;
```

```
}

/**
 * Set the value of direccionEmpresa
 * @param newVar the new value of direccionEmpresa
 */
public void setDireccionEmpresa (String newVar) {
  direccionEmpresa = newVar;
}

/**
 * Get the value of direccionEmpresa
 * @return the value of direccionEmpresa
 */
public String getDireccionEmpresa () {
  return direccionEmpresa;
}

/**
 * Set the value of telefonoEmpresa
 * @param newVar the new value of telefonoEmpresa
 */
public void setTelefonoEmpresa (String newVar) {
  telefonoEmpresa = newVar;
}

/**
 * Get the value of telefonoEmpresa
 * @return the value of telefonoEmpresa
 */
public String getTelefonoEmpresa () {
  return telefonoEmpresa;
}

/**
 * Set the value of m_departamento
 * @param newVar the new value of m_departamento
 */
public void setDepartamento (Departamento newVar) {
  m_departamento = newVar;
}

/**
 * Get the value of m_departamento
 * @return the value of m_departamento
 */
public Departamento getDepartamento () {
  return m_departamento;
}

/**
 * Add a Departamento object to the departamentoList
 */
public void addDepartamento (Departamento new_object) {
  departamentoList.add(new_object);
}
```

```
  /**
   * Remove a Departamento object from departamentoList
   */
  public void removeDepartamento (Departamento new_object)
  {
    departamentoList.remove(new_object);
  }


   /**
   * Get the List of Departamento objects held by departamentoList
   * @return List of Departamento objects held by departamentoList
   */
  public List<Departamento> getDepartamentoList() {
    return departamentoList;
  }

}
```

4.2.4. *Clase Departamento*

```
import java.util.*;

/**
 * Class Departamento
 */
public class Departamento {

  //
  // Fields
  //

  private String nombreDepartamento;
  private String ubicacionDepartamento;
  private String telefonoDepartamento;

  // Asociación entre departamento y empresa
  private Empresa m_empresa;

  // Lista de empleados en el departamento
  private ArrayList<Empleado> empleadoList = new ArrayList<Empleado>();

  // Lista de empresas asociadas al departamento
  private ArrayList<Empresa> empresaList = new ArrayList<Empresa>();

  //
  // Constructors
  //
  public Departamento () { };

  //
  // Methods
  //

  //
```

```
// Accessor methods
//

/**
 * Set the value of nombreDepartamento
 * @param newVar the new value of nombreDepartamento
 */
public void setNombreDepartamento (String newVar) {
  nombreDepartamento = newVar;
}

/**
 * Get the value of nombreDepartamento
 * @return the value of nombreDepartamento
 */
public String getNombreDepartamento () {
  return nombreDepartamento;
}

/**
 * Set the value of ubicacionDepartamento
 * @param newVar the new value of ubicacionDepartamento
 */
public void setUbicacionDepartamento (String newVar) {
  ubicacionDepartamento = newVar;
}

/**
 * Get the value of ubicacionDepartamento
 * @return the value of ubicacionDepartamento
 */
public String getUbicacionDepartamento () {
  return ubicacionDepartamento;
}

/**
 * Set the value of telefonoDepartamento
 * @param newVar the new value of telefonoDepartamento
 */
public void setTelefonoDepartamento (String newVar) {
  telefonoDepartamento = newVar;
}

/**
 * Get the value of telefonoDepartamento
 * @return the value of telefonoDepartamento
 */
public String getTelefonoDepartamento () {
  return telefonoDepartamento;
}

/**
 * Add a Empresa object to the empresaList List
 */
public void addEmpresa (Empresa new_object) {
  empresaList.add(new_object);
}
```

```
  /**
   * Remove a Empresa object from empresaList List
   */
  public void removeEmpresa (Empresa new_object)
  {
    empresaList.remove(new_object);
  }

  /**
   * Get the List of Empresa objects held by empresaList
   * @return List of Empresa objects held by empresaList
   */
  public List<Empresa> getEmpresaList() {
    return empresaList;
  }


  /**
   * Set the value of m_empresa
   * @param newVar the new value of m_empresa
   */
  public void setEmpresa (Empresa newVar) {
    m_empresa = newVar;
  }

  /**
   * Get the value of m_empresa
   * @return the value of m_empresa
   */
  public Empresa getEmpresa () {
    return m_empresa;
  }

  /**
   * Add a Empleado object to the empleadoList List
   */
  public void addEmpleado (Empleado new_object) {
    empleadoList.add(new_object);
  }

  /**
   * Remove a Empleado object from empleadoList List
   */
  public void removeEmpleado (Empleado new_object)
  {
    empleadoList.remove(new_object);
  }

  /**
   * Get the List of Empleado objects held by empleadoList
   * @return List of Empleado objects held by empleadoList
   */
  public List<Empleado> getEmpleadoList() {
    return empleadoList;
  }
}
```

5. Explicación del Diagrama de Clases

El diagrama de clases generado para este caso práctico refleja una estructura de clases que sigue los principios fundamentales de la programación orientada a objetos, específicamente en cuanto a las relaciones de herencia, agregación y asociación entre las entidades involucradas.

5.1. Herencia:

En este diagrama, la clase *Empleado* hereda de la clase *Persona*. Este es un ejemplo clásico de la relación "es un", ya que un *Empleado* es una *Persona* con atributos adicionales. La clase *Empleado* hereda los atributos y métodos básicos de *Persona* (nombre, apellidos, y NIF), pero a su vez, agrega atributos y métodos específicos relacionados con el empleo, como la dirección, teléfono y puesto. Este tipo de relación permite reutilizar código y representa una especialización de la clase base.

5.2. Agregación:

La agregación es utilizada en este diagrama para mostrar que un *Departamento* "tiene" empleados. En este caso, un *Departamento* puede tener múltiples *Empleados*, pero estos pueden existir independientemente del *Departamento*, lo que implica una relación más flexible que la composición. El *Departamento* contiene una lista de *Empleados*, lo que indica que la vida de los *Empleados* no depende directamente de la existencia del *Departamento*.

5.3. Asociación:

Se establece también una relación de asociación entre las clases *Empleado* y *Empresa*. En este caso, un *Empleado* puede estar vinculado a múltiples *Empresas*, y una *Empresa* puede tener varios *Empleados*. Esta relación no implica una dependencia directa ni una jerarquía de objetos, sino que indica que las clases colaboran entre sí. Esta relación es bidireccional, ya que tanto *Empleado* como *Empresa* mantienen una referencia mutua entre sí.

Cada clase se documentó con sus respectivos atributos y métodos, siguiendo la notación UML estándar. Los métodos de acceso (getters y setters) son utilizados para interactuar con los atributos de las clases, y se sigue la convención de nomenclatura en *lowerCamelCase* para estos métodos. El diagrama de clases proporciona una representación visual clara de cómo se relacionan estas clases, lo que facilita la comprensión del sistema y ayuda en la generación del código a partir del modelo.

6. Correcciones al código

A lo largo del trabajo, se identificaron y corrigieron varias inconsistencias en el código original generado a partir del diagrama de clases. Estas correcciones fueron

esenciales para mejorar tanto la funcionalidad como la claridad del código, lo que resultó en un diseño más eficiente y un código más mantenible. A continuación, se detallan las correcciones realizadas y sus justificaciones:

6.1. Cambio de Vector a ArrayList

En el código original, se utilizaba la clase *Vector* para gestionar las listas de objetos, como las listas de *Empleado* y *Empresa* en las clases correspondientes. Sin embargo, *Vector* es una estructura de datos más antigua y no es la opción más eficiente en Java en la mayoría de los casos.

Justificación:

- *Vector* es una clase sincronizada, lo que implica una sobrecarga innecesaria cuando no se necesita sincronización explícita (como es el caso en este trabajo).

- *ArrayList*, por otro lado, es una implementación más moderna y eficiente para manejar listas en Java. No solo es más ligera, sino que también ofrece un mejor rendimiento, especialmente cuando no es necesario gestionar concurrencia.

- Además, *ArrayList* es más flexible y ampliamente utilizada, lo que facilita la comprensión y mantenimiento del código.

Cambio realizado: Todas las instancias de *Vector* fueron reemplazadas por *ArrayList*. Este cambio mejora la eficiencia general del código y hace que sea más fácil de entender para otros desarrolladores familiarizados con las mejores prácticas de Java.

Ejemplo:

```
// Antes
public Vector<Empresa> empresaVector = new Vector<>();

// Después
public ArrayList<Empresa> empresaList = new ArrayList<Empresa>();
```

6.2. Eliminación de métodos duplicados

En la clase *Empleado*, se identificaron métodos duplicados, como getDepartamento() y setDepartamento(), que aparecían repetidos sin ninguna razón funcional. Tener métodos duplicados puede generar confusión y errores en el código, además de hacer que el mantenimiento sea más difícil.

Justificación:

- La duplicación de métodos puede resultar en un código redundante y propenso a errores. Por ejemplo, si se modifica uno de estos métodos, pero no el otro, el código podría comportarse de manera inconsistente.

- También se incrementa la complejidad del código, ya que el desarrollador debe estar al tanto de la existencia de estos métodos redundantes y de qué versión debe usarse.

Cambio realizado: Se eliminó la duplicación de los métodos getDepartamento() y setDepartamento() en la clase *Empleado*, dejando solo una versión de cada uno, lo cual simplifica el código y mejora su legibilidad.

Ejemplo:

```
// Antes
public Departamento getDepartamento() { ... }
public Departamento getDepartamento() { ... } // Método duplicado

// Después
public Departamento getDepartamento() { ... } // Solo una versión del método
```

6.3. Eliminación de código innecesario

El código generado contenía algunos métodos y constructores vacíos que no tenían ninguna función en el programa. Estos métodos eran redundantes y no cumplían con ninguna necesidad del sistema, por lo que su presencia solo añadía complejidad innecesaria.

Justificación:

- Los métodos vacíos ocupan espacio en el código y pueden confundir a otros desarrolladores que puedan pensar que tienen algún propósito.

- Al eliminar estos métodos innecesarios, se mejora la claridad del código, lo que facilita su comprensión y mantenimiento.

Cambio realizado: Se eliminaron los métodos vacíos que no aportaban nada al sistema, como los constructores sin implementación o métodos get y set redundantes que no eran utilizados por el sistema.

Ejemplo:

```
// Antes
public void setNombre(String nombre) { }

// Después
(Método eliminado ya que no tenía ninguna implementación ni utilidad)
```

6.4. Uso adecuado de la nomenclatura y convenciones de Java

Se revisaron los nombres de las variables y métodos para asegurarse de que seguían las convenciones de nomenclatura de Java, como el uso de *lowerCamelCase* para métodos y variables. Estas convenciones son importantes porque garantizan que el código sea coherente con las mejores prácticas de la comunidad de desarrolladores de Java.

Justificación:

- Las convenciones de nomenclatura facilitan la lectura del código y permiten que otros desarrolladores comprendan rápidamente el propósito de las variables y métodos.

- El uso de convenciones coherentes mejora la mantenibilidad del código, especialmente cuando el proyecto crece o cuando otros desarrolladores deben trabajar con él.

Cambio realizado: Se ajustaron los nombres de las variables y métodos para seguir las convenciones adecuadas, como cambiar nombreEmbresa a nombreEmpresa y usar *lowerCamelCase* en los métodos.

6.5. Optimización de las relaciones entre las clases

La estructura de las relaciones entre las clases *Empleado*, *Departamento* y *Empresa* fue revisada para asegurar que los objetos estuvieran correctamente relacionados de manera eficiente y con el mínimo de dependencias posibles.

Justificación:

- Las relaciones entre las clases deben reflejar las necesidades del sistema y permitir una interacción eficiente entre las entidades. En este caso, las relaciones de agregación y asociación se estructuraron de manera que las clases pudieran colaborar sin crear dependencias innecesarias.

- El diseño adecuado de las relaciones también permite que el sistema sea más fácil de extender y mantener en el futuro.

Cambio realizado: Se ajustaron las relaciones entre las clases para reflejar de manera más precisa las interacciones entre los objetos, utilizando listas como *ArrayList* para almacenar las relaciones entre *Empleado* y *Empresa* o *Departamento* y *Empresa*, y asegurando que no hubiera relaciones redundantes.

6.6. Conclusión de la mejora del código

Las correcciones realizadas en el código fueron fundamentales para mejorar tanto el rendimiento como la legibilidad del sistema. El cambio de *Vector* a *ArrayList* optimizó el uso de memoria y el rendimiento de las listas, mientras que la eliminación de métodos duplicados y la simplificación de las relaciones entre clases contribuyó a un código más claro y fácil de mantener. Además, la correcta aplicación de las convenciones de nomenclatura garantizó que el código fuera coherente y fácil de entender para cualquier desarrollador que trabajara con él en el futuro.

Estas correcciones son un reflejo de las mejores prácticas en desarrollo de software, y muestran cómo un pequeño ajuste puede tener un gran impacto en la calidad y la eficiencia del código. A través de este proceso, no solo se mejora el código en sí, sino que también se adquiere una comprensión más profunda de cómo gestionar las relaciones entre las clases de manera eficiente en sistemas orientados a objetos.

7. Conclusiones y Reflexión Final

El trabajo realizado ha sido exitoso en su conjunto. La instalación y configuración de Umbrello se llevaron a cabo sin problemas, y la herramienta resultó ser muy útil para crear un diagrama de clases detallado que refleja las relaciones entre las clases del sistema. Este tipo de modelado visual es esencial para la comprensión y desarrollo de sistemas orientados a objetos, y facilitó en gran medida la creación del código correspondiente.

El diagrama de clases generado no solo fue útil para visualizar las relaciones de herencia, agregación y asociación entre las clases *Empleado*, *Departamento* y *Empresa*, sino que también proporcionó una base sólida para generar el código fuente correspondiente en Java. Aunque la generación de código no es completamente automática en Umbrello (por ejemplo, los constructores no fueron generados), el proceso permitió que el código fuera generado rápidamente y que las relaciones entre las clases fueran claras desde el principio.

Las correcciones realizadas en el código también fueron esenciales para mejorar su calidad. El cambio a *ArrayList* y la eliminación de métodos duplicados optimizaron el rendimiento y la legibilidad, asegurando que el código sea más eficiente y fácil de mantener.

En general, el uso de Umbrello para la creación de diagramas de clases y la generación de código ha demostrado ser una práctica efectiva para estructurar sistemas y facilitar su desarrollo. Este proceso proporciona una comprensión más clara de las relaciones entre las clases y permite a los desarrolladores trabajar de manera más eficiente..

8. Referencias

- Material de la unidad
- https://jbgarcia.webs.uvigo.es/asignaturas/TO/usoArgoUML/index.html
- https://docs.kde.org/trunk5/es/umbrello/umbrello/
- https://www.youtube.com/watch?v=p0WtDSwwBfI&ab_channel=GeorgeRice
- https://drive.google.com/file/d/1ZCZmH5_mDiLEdoYaI92ZKyt7UNO3forS/view?usp=drive_link para ver la imágen del diagrama correctamente.