PROGRAMACIÓN ORIENTADA A OBJETOS
(Laboratorio de Prácticas)

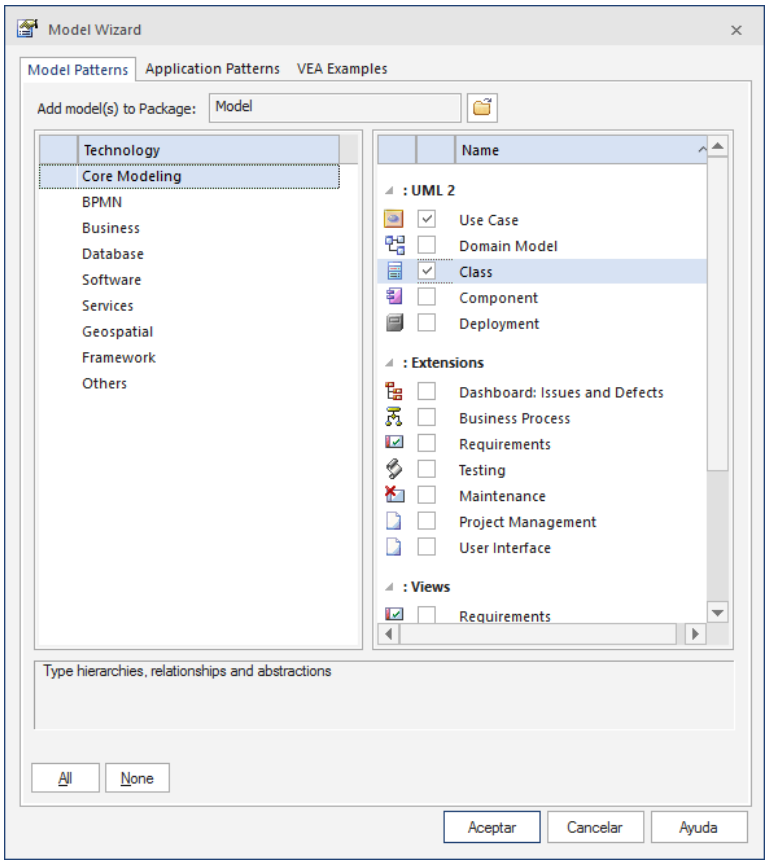
Titulaciones de Grado en Ingeniería de Computadores, Ingeniería Informática y Sistemas de Información

Sesión 5
ANÁLISIS Y DISEÑO OO CON UML

En esta sesión realizaremos análisis y diseño orientado a objetos con UML. Aprenderemos a diseñar clases con UML utilizando mecanismos proporcionados por la programación orientada a objetos con la herramienta Enterprise Architect y a generar su código en Java. Para descargar Enterprise Architect ir a la web: http://www.sparxsystems.com.au/

1. EJEMPLO BANCO

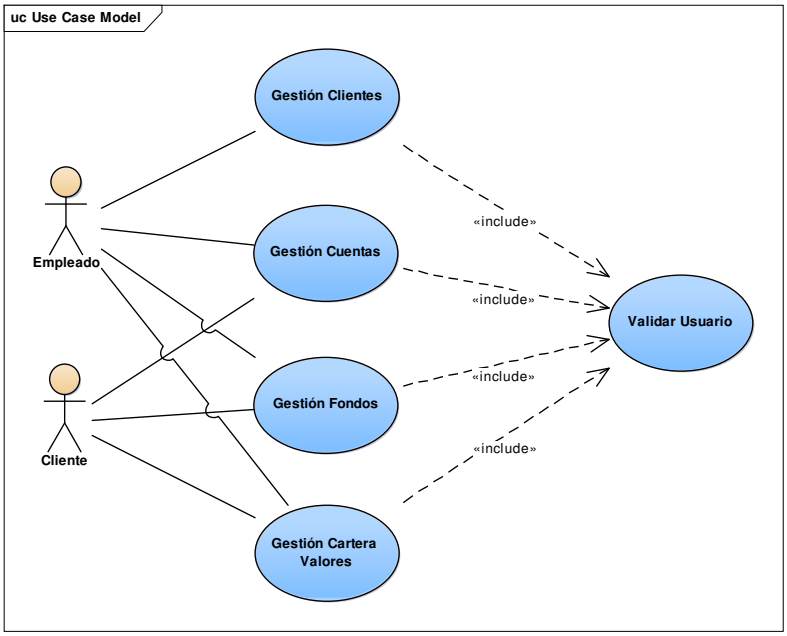
Realizaremos los diagramas de casos de uso y de clases vistos en el ejemplo de gestión bancaria del tema 4 de teoría, para ello crearemos un nuevo proyecto en Enterprise Architect seleccionando "File – New Project" o en el icono correspondiente, donde seleccionaremos los dos diagramas como se muestra en la siguiente figura:

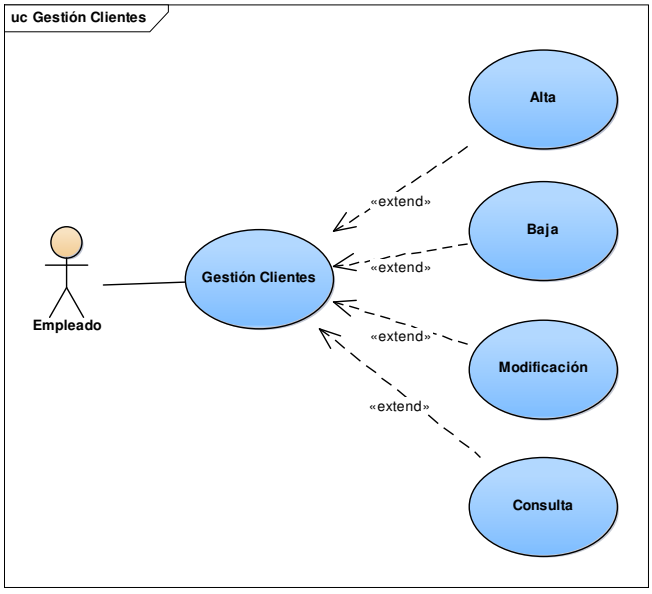


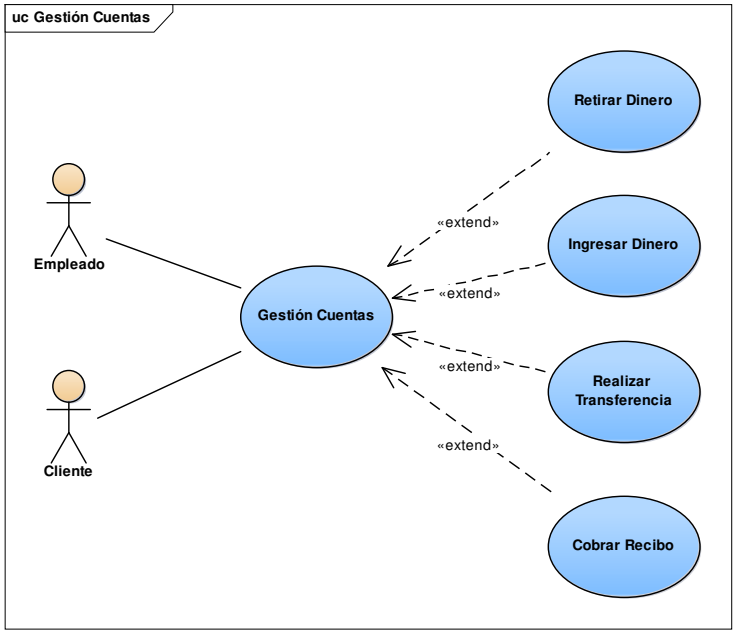
Para poder ver los distintos elementos de los dos tipos de diagramas tenemos que visualizar el Toolbox (DESING – Toolbox).

1.1 CASOS DE USO

El primer tipo de diagrama que hay que realizar son los diagramas de casos de uso presentados en el ejemplo de gestión bancaria. En estos diagramas representamos el comportamiento de un sistema desde el punto de vista de un usuario. En nuestro caso presentaremos los siguientes casos de uso principales y secundarios:







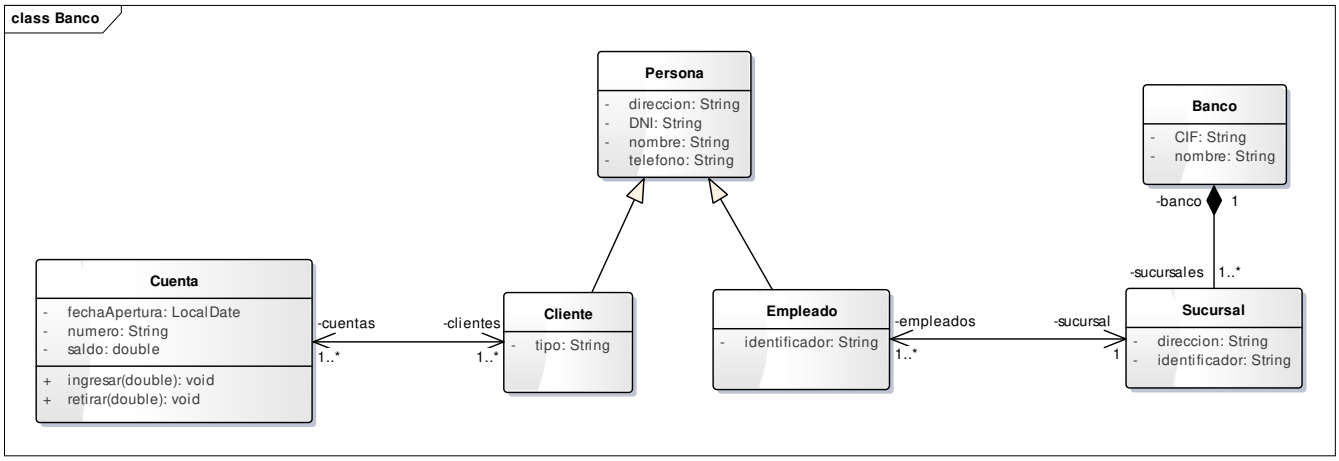
Para generar los casos de uso secundarios hay que pulsar sobre el caso de uso principal con el botón derecho del ratón y seleccionar “Add – Add Diagram – Use Case”. A continuación arrastramos el caso de uso principal y los actores siempre como “Link” y creamos los casos de uso secundarios y los relacionamos con el principal.

1.2 DIAGRAMA DE CLASES

El segundo diagrama que realizaremos será el de clases donde diseñamos las clases y las relaciones de herencia, asociación y composición que formarán parte del sistema, en nuestro caso tendremos seis clases: los clientes y empleados del banco (que heredan de la clase Persona), las cuentas, los bancos y sus sucursales. La clase Cuenta tendrá como atributos: número, saldo y fecha de apertura y como métodos *ingresar(double cantidad)* y *retirar(double cantidad)*. La clase Persona tendrá como atributos: DNI, nombre, dirección y

teléfono. La clase Cliente tendrá un único atributo: tipo. La clase Empleado tendrá un único atributo: identificador. La clase Banco tendrá los atributos nombre y CIF. La clase Sucursal tendrá los atributos: identificador y dirección. El diagrama tendrá dos relaciones de asociación entre las cuentas y clientes y entre empleados y sucursales, por último tendremos una relación de composición entre el banco y las sucursales.

Realizar el diagrama de clases como se muestra en la siguiente figura:



1.2.1 Ejercicio Tipos de Cuenta y Dirección

Realizar dos clases hijas de la clase Cuenta para distinguir dos tipos de cuentas: corrientes y a plazo. Las cuentas corrientes tendrán un atributo de tipo long que represente una tarjeta de crédito y las cuentas a plazo un atributo int que representa los meses que permanecerá abierta.

Crear la clase dirección donde se tengan los siguientes datos: calle, número, código postal, localidad y provincia. Borrar el atributo dirección de la clase Persona y Sucursal y asociarle la nueva clase dirección.

1.2.2 Generación código Java

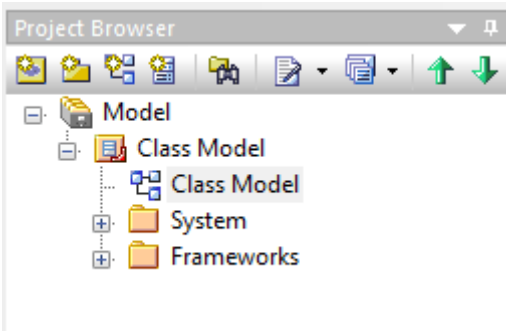
Generar el código en Java a partir de las clases creadas en los apartados anteriores. Para realizarlo se puede desde el menú:
CODE – Generate All
También se pueden seleccionar las clases que queremos generar y con el botón derecho del ratón en el menú contextual que aparece seleccionar:
Code Generation – Generate Selected Elements

Completar las clases desde Netbeans creando un nuevo proyecto e insertando las clases generadas.

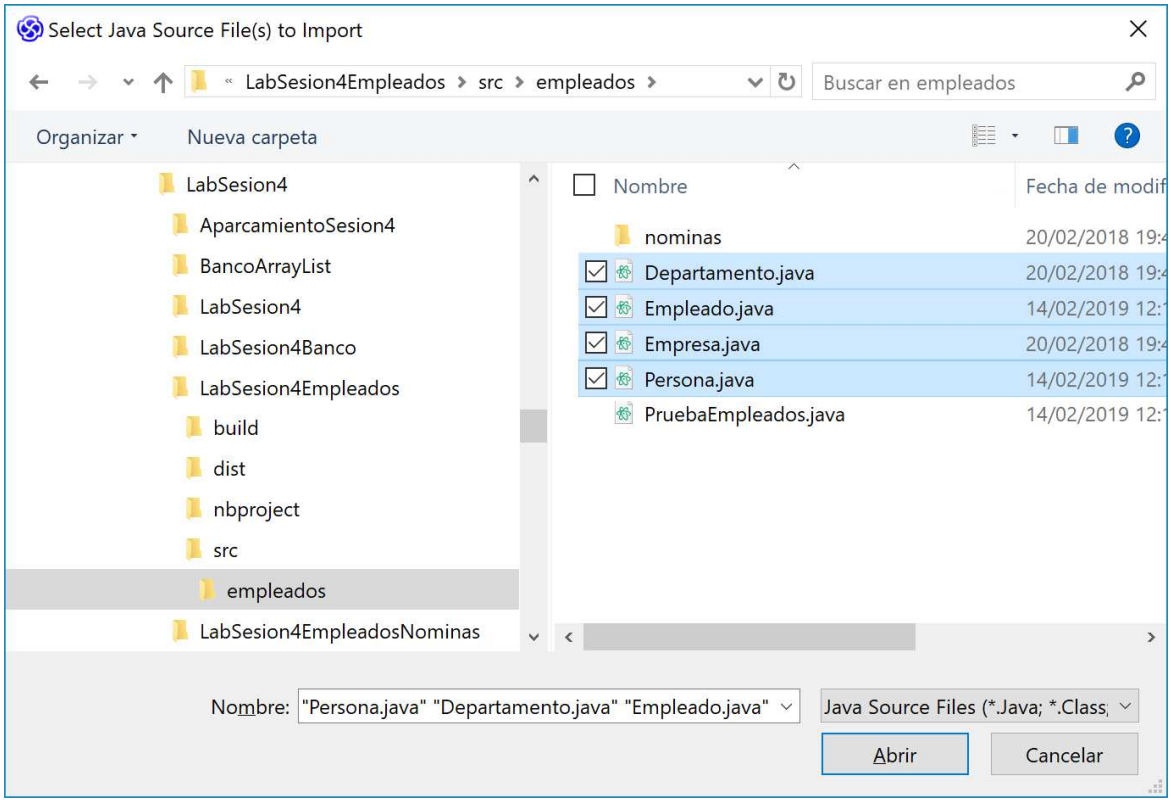
2. INGENIERÍA INVERSA

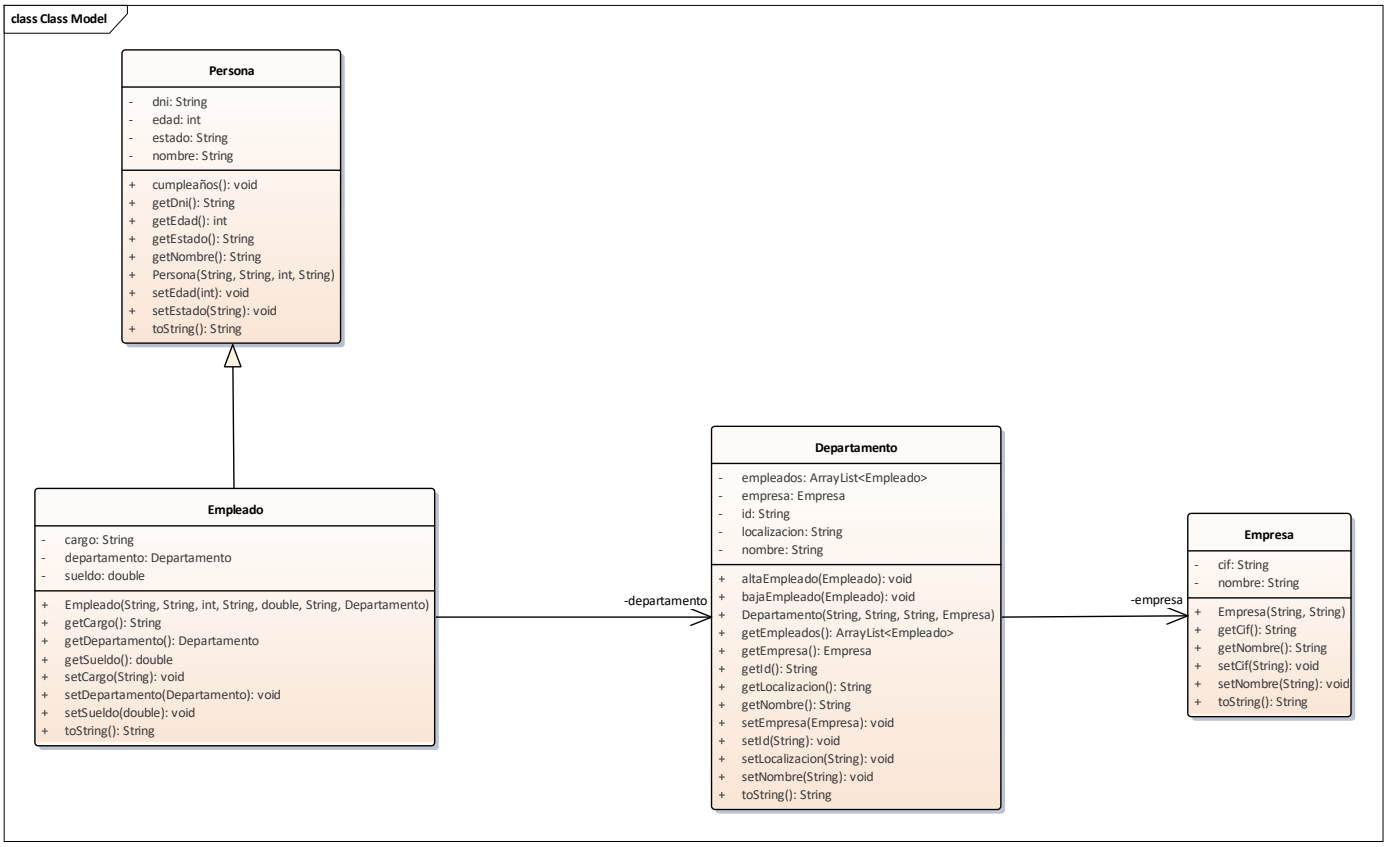
En este apartado aprenderemos a generar un diagrama de clases UML a partir de código Java. Lo primero que debemos hacer es comprobar que el código sea correcto, para ello lo primero es compilarlo y verificar que no tiene errores. Vamos a tomar como ejemplo la aplicación de Empleados que utilizamos en la sesión 4 (sin las nóminas).

Crearemos un nuevo proyecto que llamaremos “EmpleadosDC” y seleccionamos el diagrama de clases como modelo. Una vez generado el proyecto seleccionamos el “Class Model” en el “Project Browser” y borramos todo lo que aparece en el diagrama.



A continuación, seleccionamos del menú “CODE – Import Code – Java Files” y seleccionar el código fuente *.java:





El último paso es distribuir las clases de la siguiente forma (se puede utilizar: "LAYOUT – Diagram Layout"):

Tener cuidado con las ñ y acentos ya que no se representan bien. Habría que completar el diagrama con las cardinalidades de las relaciones de asociación.

3. EJERCICIO ALQUILER DE INMUEBLES

Este ejercicio consiste en el diseño con UML de una aplicación que permita gestionar los alquileres de inmuebles de una empresa.

Lo primero que hay que destacar es que este ejercicio tiene como objetivo el diseño de una aplicación mediante POO de una cierta complejidad. En este sentido, hay que tener en cuenta que la funcionalidad descrita en el enunciado constituye una simplificación de la realidad, y en algún aspecto concreto, puede diferir de la práctica habitual de las operaciones descritas.

Una empresa gestiona el alquiler de un conjunto de inmuebles que administra en calidad de propietaria. Cada inmueble puede ser bien un piso, un local comercial o una oficina. En el caso de las oficinas se puede alquilar un edificio de oficinas completo o de forma parcial dividiendo las oficinas por plantas entre las empresas interesadas. Los pisos pueden ser alquilados por particulares (inquilinos), los locales comerciales y las oficinas solo por empresas.

Cada inmueble tiene los siguientes datos: identificador (único dentro de la aplicación) y dirección (compuesto por: calle, número, código postal, localidad y provincia). En el caso de los pisos se especificará su planta y puerta y el precio mensual del alquiler. En el caso de los locales comerciales se especificará la superficie que tiene expresada en metros cuadrados y el precio por metro cuadrado. En el caso de los edificios de oficinas se especificarán las plantas que tiene y la superficie de cada planta también en metros cuadrados así como su precio por planta.

Para que un particular pueda alquilar un piso es necesario que presente una nómina actualizada donde se refleje que el salario que recibe es mayor que el precio del alquiler del inmueble. Además de su nómina debe proporcionar los siguientes datos: NIF, nombre, teléfono, fecha de nacimiento y correo electrónico.

Para que una empresa pueda alquilar un local comercial o una oficina es necesario saber que tiene suficiente solvencia financiera para afrontar el pago del alquiler, para ello debe presentar su balance general donde se vean reflejados los fondos propios, los activos y los pasivos. Además deberá proporcionar los datos propios de la empresa como su nombre, el CIF, dirección, teléfono y correo electrónico.

Cuando se alquila un inmueble determinado no podrá ser alquilado de nuevo hasta que no quede vacío, ya que no se puede alquilar a más de un inquilino o empresa. Cuando se realiza el alquiler se debe formalizar un contrato de alquiler, para el cual se debe registrar la fecha de realización, la duración de éste (en meses) y los datos del inquilino o empresa.

Por otra parte la aplicación debe permitir la generación de los recibos mensuales de los inmuebles alquilados de forma que se refleje el número de recibo (único para cada inmueble y que no varía a lo largo del tiempo), la cantidad a pagar (desglosado en base, IVA y total), la fecha de emisión, el agua y la luz. Además para cada recibo se debe saber si está cobrado o no. Para facilitar la emisión de los recibos cada mes la aplicación permitirá la generación de recibos idénticos a los del mes anterior pero con el cambio de la fecha. También se tiene que permitir la modificación de recibos emitidos en meses anteriores al actual. La aplicación también debe permitir la impresión de los recibos en diversos formatos para su envío al cliente.

También se desea controlar los movimientos bancarios que se producen para cada inmueble. Un movimiento bancario siempre está asociado a un banco y a una cuenta de ese banco. En esa cuenta existirá un saldo (positivo o negativo) que aumentará o disminuirá con cada movimiento. Para cada movimiento se desea saber también su descripción y la fecha en la que se ha realizado. Un movimiento bancario puede ser de dos tipos o un ingreso o un gasto. Si el movimiento bancario es un gasto estará asociado a un inmueble determinado y se indicará el tipo de gasto (ejemplos: reparación del ascensor, de la caldera, sueldo personal limpieza, etc.). En el caso de los ingresos corresponderán a los recibos cobrados de los inmuebles. A partir de los movimientos bancarios la aplicación será capaz de realizar informes económicos personalizados.

Por último la aplicación será capaz de generar listados a partir de los datos almacenados, por ejemplo, listado de todos los inmuebles, listado de inquilinos o empresas ordenados por fecha, listado de inquilinos o empresas que han pagado o no sus recibos, listado de todos los recibos pendientes de pago en un determinado periodo etc.

- Realizar diagramas de casos de uso que engloben las siguientes acciones (se deben generar casos de uso principales y secundarios) que asignaremos a dos tipos de actores: gestor de la empresa de alquiler y clientes (inquilino y empresa):
  1. Gestor: Gestión de pisos, locales, oficinas y clientes (altas, bajas, modificaciones y consultas).
  2. Cliente: Alquilar, desalquilar, modificación y consulta de sus datos. El cliente debe identificarse en el sistema antes de realizar estas operaciones.
  3. Gestor: Generación de recibos, gestión de movimientos bancarios y generación de listados.

- Realizar un diagrama de clases donde se identifiquen las clases necesarias para realizar la aplicación encargada de la gestión de los alquileres.

- A partir del diagrama de clases generar el código de la aplicación en Java.