ACTIVIDAD 1

En esta unidad has visto las características fundamentales de la programación orientada a objetos y estudiado los conceptos más importantes de este modelo de programación. Has aprendido a crear y manipular objetos, qué son los métodos y cómo se definen los parámetros. También has podido conocer cómo se estructura la Biblioteca de Clases de Java, viendo algunas clases importantes, como por ejemplo las que nos permiten realizar la programación de la consola. Para poder realizar la tarea de esta unidad vas a crear dos clases con una estructura básica. Las tareas a realizar se centrarán en la creación de instancias de esas clases, la creación y utilización de unos métodos básicos, y el trabajo con constructores y parámetros.

Para poder realizar la tarea de esta unidad vas a crear dos clases con una estructura básica. Las tareas a realizar se centrarán en la creación de instancias de esas clases, la implementacióny utilización de unos métodos básicos, así como el trabajo con constructores con y sin parámetros . Además de realizar entrada de datos por teclado y salida de resultados por consola.

1. Construye un proyecto en Java que cree la clase Alumno en la que se define los atributos o variables miembro que se indican a continuación:

   public class Alumno
   {
   private String nif;
   private String nombre;
   private String apellidos;
   private String direccion;
   private String poblacion;
   private String matricula;
   }

   public setNombre(String nombre)
   {
   this.nombre=nombre;
   }
   public getNombre()
   {
   return this.nombre;
   }

   Añade a la clase Alumno los métodos que faltan para poder consultar y modificar el valor de todos los atributos. Para ello observa cómo se han creado los métodos del atributo nombre y determina los parámetros y resultado de los demás atributos . El atributo nif tendrá que llevar la comprobación de la letra, para esto declararemos un método interno dentro de la clase, que se llamará desde el método **setNif** siguiendo las recomendaciones que se indican más abajo. Además deberás definir dos constructores: el constructor vacío, y el constructor con todos los valores de los atributos.

2. Crea otra clase que se llamará **EntradaAlumnos,** esta clase tendrá un método main donde se declararán 2 objetos de la clase Alumno, uno de ellos llamado alumno1 se inicializarán todos sus atributos a través de la entrada por teclado de esos datos, sin utilizar variables intermedias locales del método main, el otro objeto se instanciará y llenará a través de

variables locales que se han declarado en el método main, usando para inicializar el objeto alumno2 el constructor en el que se pasan todos los parámetros.

3. Se tendrá en cuenta también que el nif introducido sea correcto, llamado para esto al método interno de la clase Alumno, definido anteriormente.

4. Una vez que se le han dado valor a todos los atributos de los 2 objetos, se sacarán por consola los datos de estos 2 objetos, sacando su DNI,nombre...,etc

Recursos necesarios para realizar la Tarea.

- Ordenador Personal.
- Sistema operativo Windows o Linux.
- JDK y JRE de Java.
- IDE Eclipse / NetBeans.

Consejos y recomendaciones.

La letra del NIF se obtiene a partir de un algoritmo conocido como módulo 23. El algoritmo consiste en aplicar la operación aritmética de módulo 23 al número del DNI. El módulo 23 es el número entero obtenido como resto de la división entera del número del DNI entre 23. El resultado es un número comprendido entre el 0 y el 22. En base a una tabla conocida se asigna una letra. La combinación del DNI con esa letra es el NIF.

Este mismo algoritmo también puede utilizarse para el cálculo del NIE. En el caso que el NIE empiece por X, se calcula despreciando la X y utilizando los 7 dígitos, si el NIE empieza por Y, se sustituye la letra Y por el número 1, si el NIE empieza por Z, se sustituye la letra Z por el número 2 y se realiza el mismo cálculo.

El algoritmo no se aplica para obtener el Código de Identificación Fiscal (CIF), que es el "NIF" propio de las personas jurídicas, pues la letra que tiene no se basa en una fórmula, sino que identifica el tipo de entidad (p.e. B para Sociedades Limitadas; G para Asociaciones sin ánimo de lucro y otros tipos no definidos, etc.).

Tabla de asignación

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22

T R W A G M Y F P D X B N J Z S Q V H L C K E

No se utilizan las letras: I, Ñ, O, U

La I y la O se descartan para evitar confusiones con otros caracteres, como 1, l o 0.

Se usan veintitrés letras por ser este un número primo.

Para una mejor organización de la tarea, crea una carpeta para cada ejercicio llamada EjercicioXX, donde la XX es el número del ejercicio. Sitúa dentro de cada carpeta los ficheros de cada proyecto y cuando todos los ejercicios estén completos comprime esas carpetas en un sólo archivo.

ACTIVIDAD 2

A continuación, se enumeran varios supuestos que debes resolver a través de la realización de un programa para cada uno de los ejercicios propuestos.

- **Supuesto 1:** Realiza un programa que pida números enteros por teclado hasta que la suma de los 2 últimos números introducidos sea 20, diciendo al final cuantos de ellos han sido pares y cuantos impares.
- **Supuesto 2**: Realiza un programa que pida la entrada por teclado de los datos de 3 alumnos (deberemos utilizar la clase declarada en la tarea de la unidad anterior), alumno1,alumno2,alumno3. Sacando al final por consola los datos de estos alumnos ordenados por orden alfabético de sus apellidos.

  En este ejercicio debes utilizar el método de la clase String **CompareTo().**

  cadena1.**CompareTo()**(cadena2)

- **Supuesto 3:** Realiza un programa que me pida cadenas hasta que el número de palabras introducidas en total sea mayor que 30, y me diga cual de las cadenas introducidas tiene mas vocales, así como cuantas palabras de todas las introducidas han terminado en 's', teniendo en cuenta que solo debe contar una palabra independientemente de los espacios en blanco que haya entre cada palabra.

En este ejercicio debes utilizas los métodos de la clase String siguientes:

  - **charAt(posición).**
  - **length.**
  - **subString(inicio,final).**

- **Supuesto 4:** Escribe un programa que muestre un menú por consola, con las siguientes opciones:
  1. Factorial
  2. Multiplicación
  3. División
  4. Salir

Elegir opción:___

- Realizando estas operaciones de la manera que se indica en los ejemplos que se detallan a continuación:
- Ejemplo:
- Factorial: 5!→ 5 x 4 x 3 x 2 x 1
- Multiplicación mediante sumas: 4 x 5 →4 +4+4+4+4=20
- División mediante restas: 11 / 3 → 11-3=8, 8 -3=5, 5 -3=2 Resto: 2 Cociente: 3En la opción 1 se pedirá 1 número, en las opciones 2 y 3 se deberán pedir 2 números enteros y mostrar el resultado según la opción elegida, en la opción 3 se debe tener en cuenta que no se puede dividir por 0.

En todos los supuestos deberás implementar la captura de errores con la estructura

try

{

}

catch{}

Lo que debes entregar: En un único archivo, el código fuente de cada uno de los supuestos debidamente identificados (archivos *.java).

Recursos necesarios para realizar la Tarea.

- Entorno de desarrollo NetBeans / Eclipse.
- **http://docs.oracle.com/javase/7/docs/api/java/lang/String.html**. Especificación clase String con sus métodos.

Consejos y recomendaciones.

El orden de realización de los supuestos no importa.

El código fuente de cada uno de los supuestos debe estar debidamente identificado y separado de los demás.

Se recomienda que antes de enviar el código fuente hayas realizado la compilación y ejecución de los programas, de este modo evitarás posibles fallos.

Utiliza comentarios a lo largo de tu código.

Utiliza tabulaciones y estructura tu código para que sea lo más legible posible.

ACTIVIDAD 3

A lo largo de esta unidad has ido aprendiendo a crear tus propias clases así como sus distintos miembros (atributos y métodos). Has experimentando con la encapsulación y accesibilidad (modificadores de acceso a miembros), has creado miembros estáticos (de clase) y de instancia (de objeto), has escrito constructores para tus clases, has sobrecargado métodos y los has utilizado en pequeñas aplicaciones. También has tenido tu primer encuentro el concepto de herencia, que ya desarrollarás en unidades más avanzadas junto con otros conceptos avanzados de la Programación Orientada a Objetos.

Una vez finalizada la unidad se puede decir que tienes un dominio adecuado del lenguaje Java como para desarrollar tus propias clases y utilizarlas en una aplicación final que sea capaz de manipular un conjunto de datos simple. Dada esa premisa, esta tarea tendrá como objetivo escribir una pequeña aplicación en Java empleando algunos de los elementos que has aprendido a utilizar.

En la tarea crearemos cuatro clases una denominada Persona, otra denominada Profesor, otra Cursos y otra Principal. En la Clase Principal se trata de desarrollar una aplicación Java por consola que permita gestionar 3 Profesores que heredarán los datos de la clase Persona declarada anteriormente, y 3 cursos (los cuales serán los que tengan los 3 profesores. Mediante un menú se podrán realizar determinadas operaciones:

1.- Entrada de Cursos.

2.- Listado de Cursos.

3.- Entrada de datos Profesores.

4.- Consulta de un Profesor.

5.- Listado de Profesores con sus cursos

6.- Entrada de cursos a un Profesor.

7.- Salir.

En la **opción 1** entrará los siguientes datos:

Clase Curso: int cod_curso, String Denominación

En la **opción 2** se sacarán los datos de todos los cursos introducidos (máximo 3).

En la **opción 3** entrará los siguientes datos:

Clase Profesor: String Muface, int cod_curso, Persona (String NIF, String Nombre, String Dirección, String Población) . Quedando el curso a null.

En la **opción 4** se pedirá el nif de un profesor y se sacarán los datos de dicho profesor si está dado de alta, indicando la denominación del curso que se le hubiera asignado.

En la **opción 5** se sacará un listado de todos los datos de los profesores con la denominación de su curso si lo tiene ya asignado.

En la **opción 6** se pedirá el nif del profesor y se le asignará un curso mediante su código. Teniendo en cuenta que el curso que se le asigne tiene que ser uno de los introducidos, y que no se podrá repetir dos profesores para el mismo curso.

Además del programa principal de la aplicación (clase con una función main), habrá que escribir las clases indicadas anteriormente que proporcione todas las herramientas necesarias para trabajar con este tipo de información:

- Constructor (o constructores) adecuados.
- Métodos set y get.
- Obtención de información sobre el los objetos instanciados
- Aquellas herramientas auxiliares necesarias para poder trabajar cómodamente con el objeto. Algunas de esas herramientas podrán ser públicos y otras quizá no. Algunas podrán ser específicas de clase y otras podrán ser de objeto (métodos de objeto privados, métodos estáticos públicos, etc.).

En general, **deberás incluir excepciones** para controlar aquellos casos en los que el uso de un método no sea posible .

El código fuente Java de esta clase debería incluir comentarios en cada atributo (o en cada conjunto de atributos) y método (o en cada conjunto de métodos del mismo tipo) indicando su utilidad. El programa principal también debería incluir algunos comentarios explicativos sobre su funcionamiento y la utilización de objetos de la clase.

Además del programa deberás escribir también un informe con todas las consideraciones oportunas que se necesiten para entender cómo has realizado la tarea.

El proyecto deberá contener al menos 4 archivos fuente Java:

- **Programa principal** (clase con método **main:AplicacionGestionProfesores.java**).
- **La clase Persona (Persona.java).**
- **La clase Profesor(Profesor.java).**
- **La clase Curso (Curso.java)**

El documento que contendrá el informe lo elaborarás con un procesador de texto. Debe ser de tipo ".doc" (Microsoft Word) o de tipo ".odt" (OpenOffice.org).

Creación de la clase Persona..... 0,50

Creación de la clase Profesor..... 1,00

Creación de la clase Curso..... 1,00

Creación de la clase Principal.....1,50

Entrada de cursos.....1,00

Entrada de Profesores.....1,50

Consulta Curso.....1,00

Listado Profesores.....1,00

Asignación de cursos.....1,50

Criterios de puntuación. Total ….....10 puntos.

Para poder empezar a aplicar estos criterios es necesario que la aplicación compile y se ejecute correctamente en un emulador. En caso contrario la puntuación será directamente de 0,00.

ACTIVIDAD 4

Realizar un programa que declare las clases siguientes:

1.- Una **clase Objeto** con dos atributos enteros (int) **a y b**, y uno del tipo String **nombre**, la clase deberá contemplar en su cabecera la sentencia siguiente:

Class Objeto implements Comparable <Objeto>()

En otra **clase ArbolOrdenado** se declarará una estructura de árbol (estructura de conjunto **TreeSet)** donde se a**lmacenarán 5 objetos de la clase Objeto**, y otra estructura dinámica de lista (**ArrayList**) donde se copiaran los objetos almacenados en el **TreeSet** .

La entrada de objetos que se elige con la **opción 1**, llenará el árbol utilizando un método comparador por nombre definido en la clase Objeto (no podrá haber 2 objetos con el mismo nombre).

En la **opción 2** se sacará un listados de los objetos que hay en el árbol, viendo que están ordenados por nombre (no importan mayúsculas). (**Figura 1**)

En la **opción 3** se declarará otro conjunto ArrayList pasándole los 5 objetos del árbol anteriormente llenado, pero con el método de ordenación por la suma de los atributos enteros a y b de la clase Objeto, este nuevo método de comparación también estará definido en la clase Objeto . (Puedes utilizar el método sort de la clase Collections para que la entrada sea ordenada). (**Figura 2**)

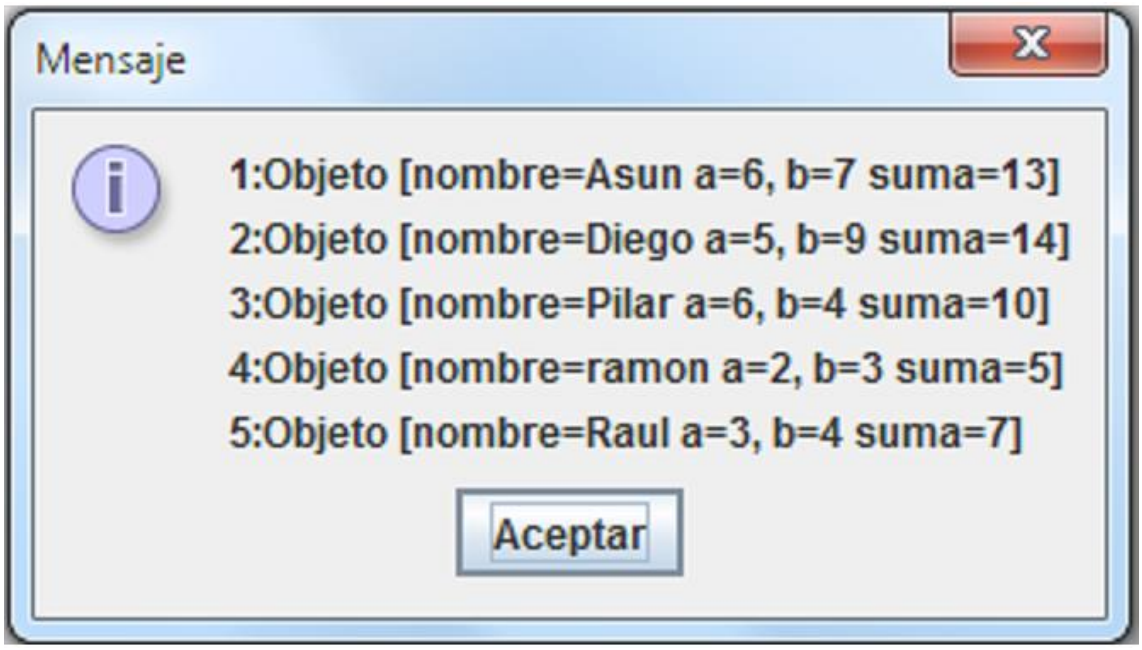
Todas las opciones en está clase se elegirán mediante un menú con las siguientes opciones:

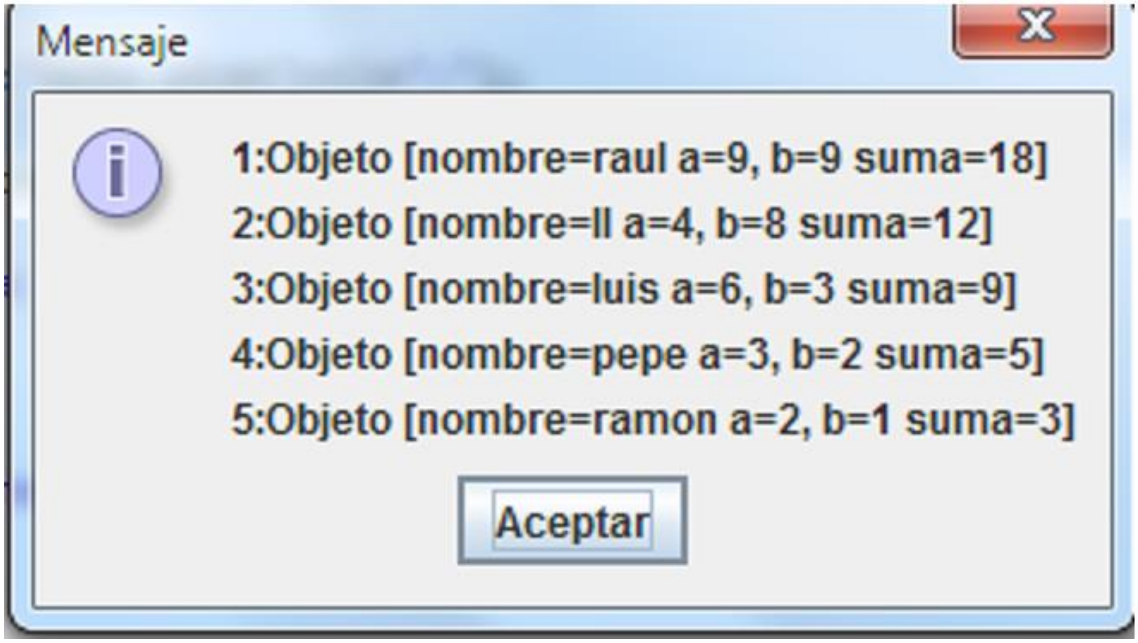
1.- Entrada de 5 Objetos.

2.- Listado ordenado nombre.

3.- Listado ordenado suma.

4.- Salir.





Recursos necesarios para realizar la Tarea.

- Ordenador personal con un procesador de textos que pueda generar archivos pdf.
- Entorno de desarrollo NetBeans / Eclipse.
- Conexión a Internet.
- Navegador Web.
- Programa de impresión en formato pdf.

Consejos y recomendaciones.

El código fuente de cada uno de los supuestos debe estar debidamente identificado y separado de los demás.

Se recomienda que antes de enviar el código fuente hayas realizado la compilación y ejecución de los programas, de este modo evitarás posibles fallos.
Utiliza comentarios a lo largo de tu código.
Utiliza tabulaciones y estructura tu código para que sea lo más legible posible.

ACTIVIDAD 5

Se trata de hacer una aplicación en Java que gestione los servicios de mesa en un restaurante. La aplicación tendrá que actualizar los datos de los siguientes ficheros serializados (Para grabar objetos serializados):

1.- Camareros.

2.- Productos.

3.- Servicios Mesa.

4.- Consumición de Mesa.

Las 4 clases de objetos para almacenar en los archivos anteriores, tendrán las siguientes variables miembros.

Los datos de **camareros** son:

- NIF. (String)
- Nombre. (String)
- Teléfono. (String)
- Dirección. (String)
- Fecha Alta en compañía. (Date).

Los datos de **productos** son:

- Cod_barra. (String)
- denominación (String)
- pvp (float)
- unidades (int)

Los datos de **serviciomesa** son:

- numeromesa (int) // las mesas posibles serán 12.
- Nif (String) // camarero
- fecha (date)
- numeroservicio (int) // Un número nuevo para cada servicio
- abierta (booleana) // si todavía se está sirviendo la mesa estará abierta

Los datos de **consumiciones** de cada mesa serán:

- cod_barra (String) // producto
- unidades (int)
- numeroservicio (int) (Es el número que tiene cada mesa)
- total (float)
  En una clase principal se realizará un menú para realizar las siguientes operaciones:

1.- **Entrada Camareros.**
2.- **Consulta Camareros.**
3.- **Entrada Productos.**
4.- **Consulta Productos.**
5.- **Abrir Servicio Mesa.**
6.- **Consumición Mesa.**
7.- **Total Cuenta de Mesa.**
8.- **Listado de los servicios de una mesa por fechas.**
9.- **Listado de los servicios que ha hecho un camarero.**
10.- **Salir.**

Las opciones del menú realizaran las siguientes operaciones:

- **Entrada Camareros**. Esta opción pedirá los datos del camarero y añadirá el registro correspondiente en el fichero.
- **Consulta Camareros**. Esta opción pedirá el NIF del camarero y sacará los datos del registro correspondiente en el fichero.
- **Entrada Productos**. Esta opción pedirá los datos del producto y añadirá el registro correspondiente en el fichero.
- **Consulta Productos**. Esta opción pedirá el Código de Barra del artículo y sacará los datos del registro correspondiente en el fichero.
- **Abrir Servicio Mesa**. Se pedirán los datos y se grabará un registro cuando se abre un nuevo servicio en esa mesa, comprobando que esa mesa está cerrada en el último servicio, además se comprobará que el número de mesa está entre la 1

y la 12, y que el nif de camarero corresponde a un camarero que está grabado en el fichero, cogiendo la fecha del sistema.

- **Consumición mesa**. Se graba un nuevo registro para cada nuevo producto o productos que se sirven en la mesa, teniendo que comprobar si esa mesa tiene el servicio abierto y que el producto está dado de alta en el fichero.
- **Total Cuenta**. Sacar el total de las consumiciones de la cuenta, se pedirá el numero de la mesa, se comprobará que la mesa esté abierta, se cerrará a continuación dicha mesa, para poder abrirla a continuación.
- **Listado de los servicios de una mesa por fechas.** Pedir el número de la mesa y la fecha de inicio y fin para sacar el total de los servicios de esa mesa entre esas fechas.
- **Listado de los servicios que ha hecho un camarero.** Pedir el nif del camarero a listar, y sacaremos el total de las mesas que ha servido dicho camarero, en una fecha determinada.

Elabora el proyecto con todas las clases necesarias, mandando todos los archivos *.java, y un documento con un procesador de texto. El documento debe ser de tipo ".doc" (Microsoft Word) o de tipo ".odt" (OpenOffice.org). En el documento escribirás un informe sobre todas las consideraciones oportunas que se necesiten para entender cómo has realizado la tarea.

Recomendaciones:

- Básate para la realización de la aplicación en la tarea de apoyo de la unidad 6 (aprovechando la clase donde se realiza la interfaz de la clase ObjectOutputStream…,MiObjectOutputStream).
- La aplicación constará de las clases: Camareros, Productos, ServicioMesa, ConsumicionMesa, Menu, y la(s) clases para tratar los ficheros.

ACTIVIDAD 6

Se trata de hacer una aplicación en Java con la librería Swing para llevar las citas médicas en una clínica. La aplicación tendrá que actualizar los datos de los siguientes ficheros serializados (Para grabar objetos serializados):

1.- Especialidades.

2.- Médicos.

3.- Pacientes.

4.- Citas Médicas.

Las 4 clases de objetos para almacenar en los archivos anteriores, tendrán las siguientes variables miembros.

Los datos de **Especialidades** son:

- **private int** id_especialidad;

- **private** String especialidad;

Los datos de **Citas Médicas** son:

- **private int** id_cita;
- **private** Date fecha;
- **private** String hora;
- **private int** id_medico;
- **private int** id_paciente;

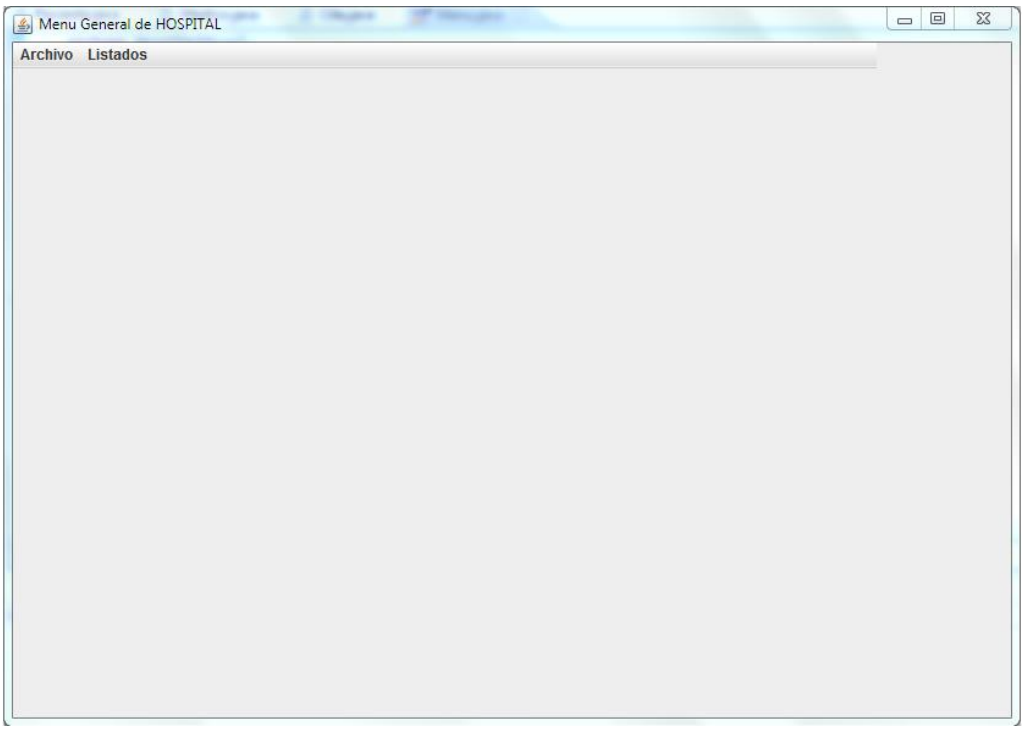
Los datos de **Médicos** son:

- **private int** id_medico;
- **private** String nombre;
- **private int** especialidad;
- **private** String puesto;

Los datos de **Paciente** son:

- **private int** id_paciente;
- **private** String ss;
- **private** String nombre;
- **private** String direccion;
- **private** String poblacion;

En una clase principal se realizará un **menú** con dos opciones generales para realizar las siguientes operaciones:

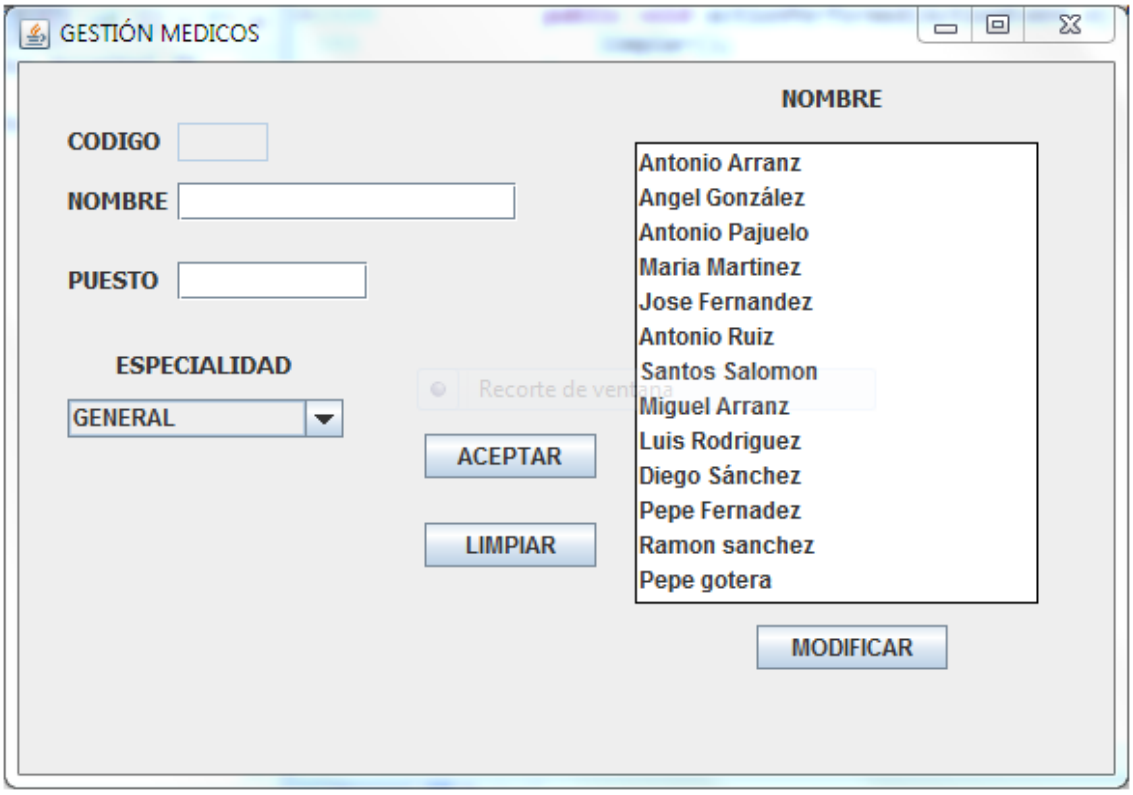


ARCHIVOS
- **Entrada/Consulta Pacientes.**
- **Entrada/Consulta Especialidades.**
- **Entrada/Consulta Médicos.**
- **Entrada Citas.**

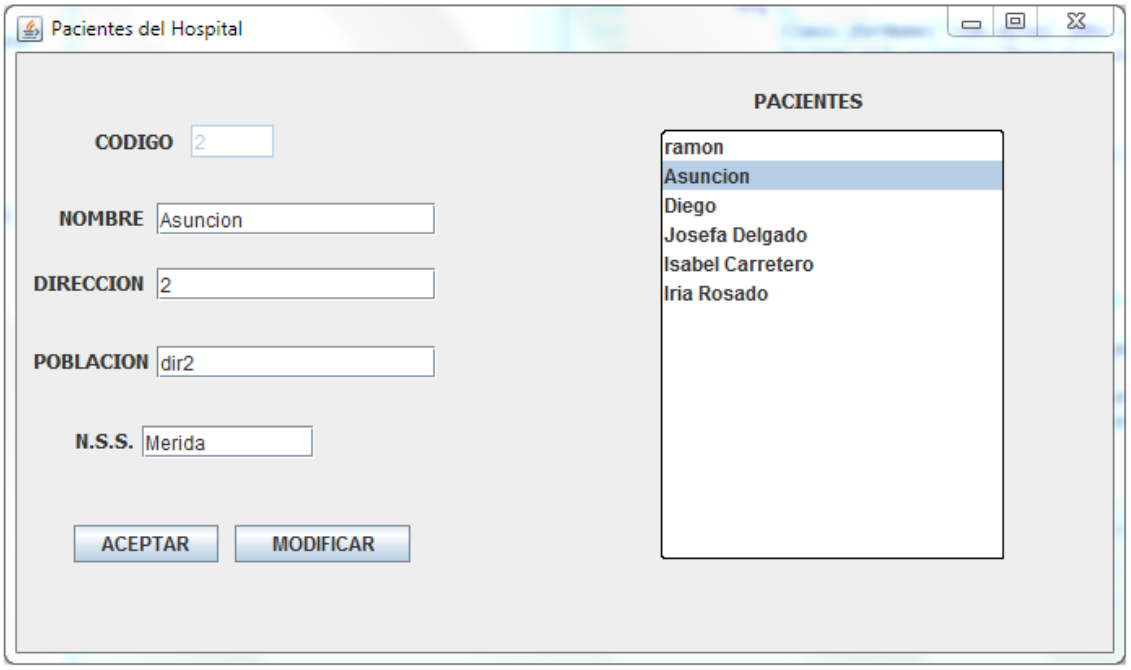
LISTADOS
- **Listado Citas de un paciente.**

Las opciones del **menú** realizarán las siguientes operaciones:

- **Entrada/Consulta Médicos.** Esta opción pedirá los datos del médico, teniendo que seleccionar la especialidad en el comboBox, que se cargará con los datos del fichero especialidad, y añadirá el registro al fichero de médicos cuando se pulse el botón **aceptar**, cuando se selecciones en la lista un médico, saldrán sus datos en las cajas de texto correspondiente.



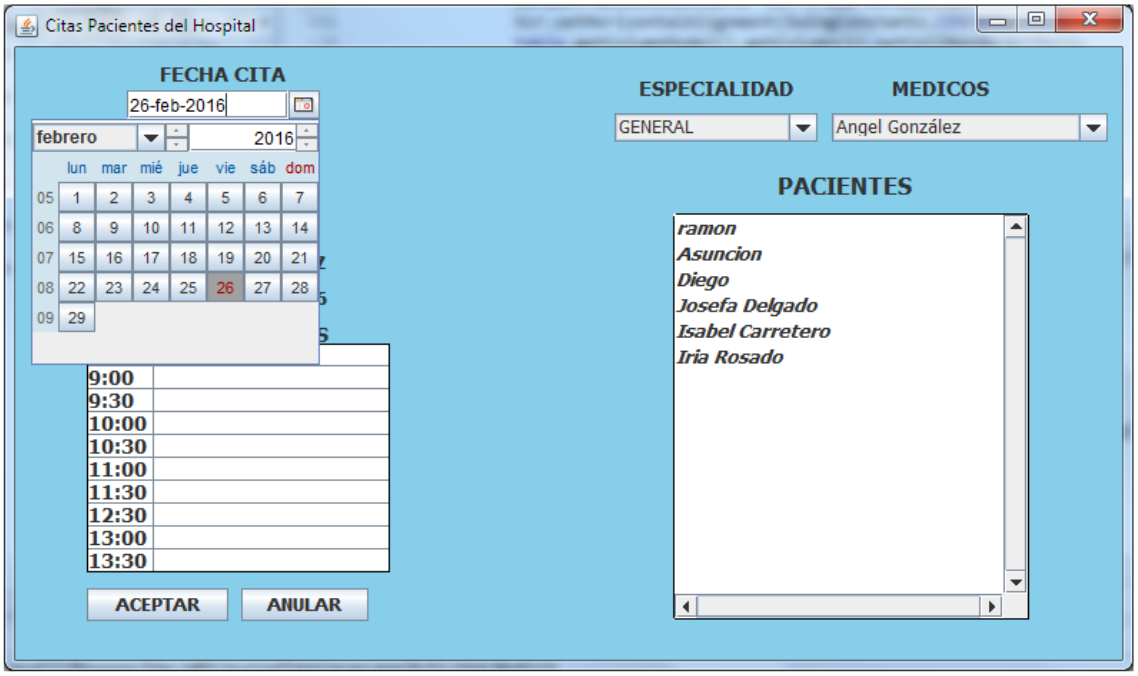
- **Entrada/Consulta Pacientes.** Esta opción pedirá los datos del paciente, y añadirá el registro correspondiente en el fichero cuando se pulse el botón **aceptar**, cuando se selecciones en la lista un paciente, saldrán sus datos en las cajas de texto correspondiente.

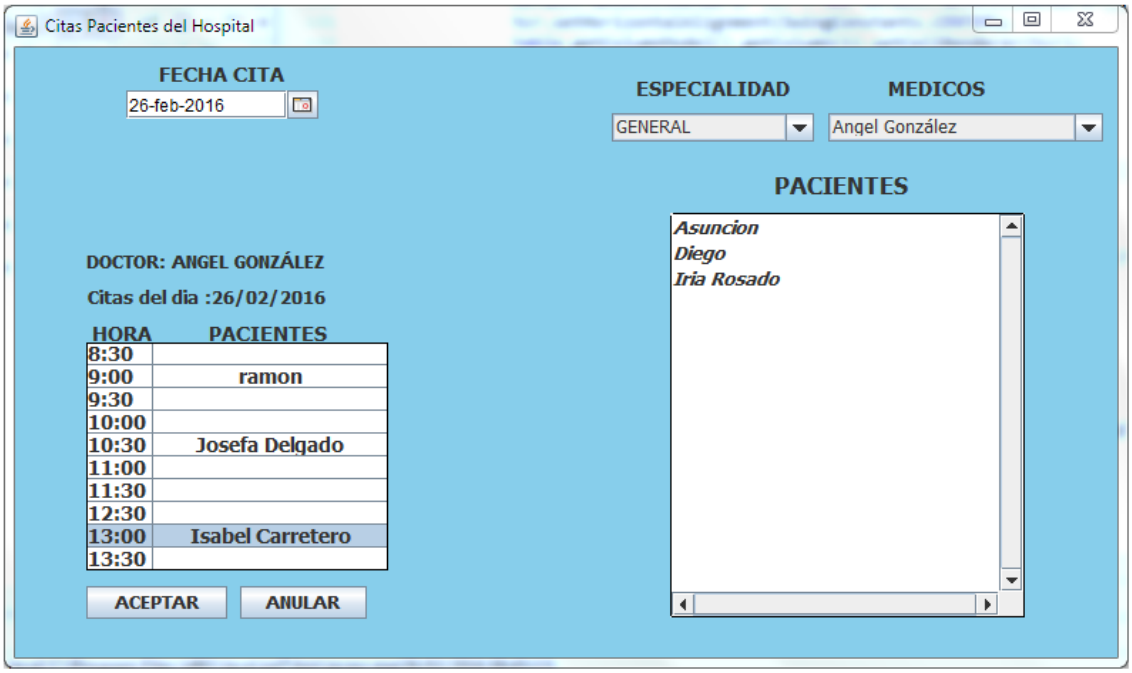


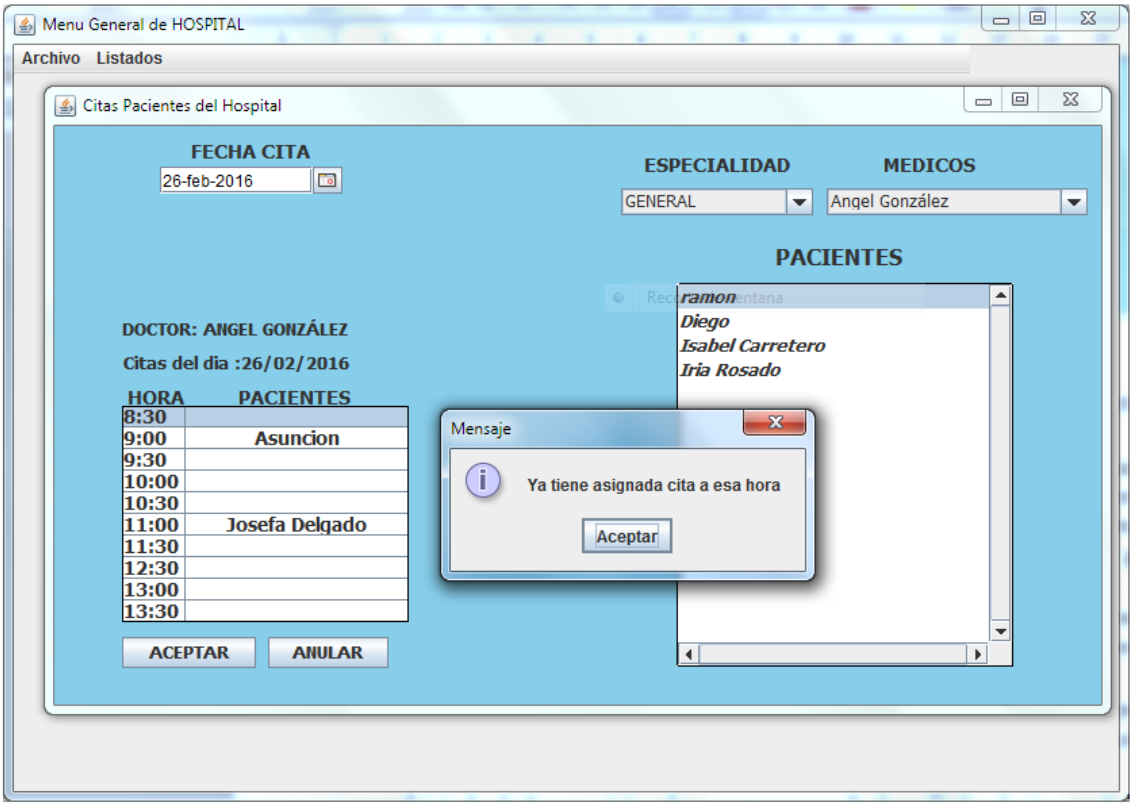
- **Entrada/Consulta Especialidades.**. Esta opción pedirá los datos de la especialidad, y añadirá el registro correspondiente en el fichero, cuando se selecciones en la lista una especialidad, saldrán sus datos en las cajas de texto correspondiente.
- **Entrada Citas.** Esta opción llevará las citas médicas por fechas de cada médico.
    - Para ello cargaremos el combo Especialidades con los datos que existen en este fichero, una vez seleccionada una especialidad se llenará el combo médicos con el nombre de los médicos que tienen esa especialidad, cuando se selecciona un médico y una fecha (puedes bajarte de internet el archivo jar jcalendar e instalarlo en tu NetBeans o Eclipse), con los datos seleccionados sacaras en la agenda del facultativo las citas ya concertadas con los nombres de los pacientes, y en la lista de pacientes todos aquellos que no tienen cita fijada ese día con dicho médico.
    - Para asignar una cita selecciona un paciente de la lista y pincha en la fila correspondiente del JTable, en ese momento se borrará al paciente de la lista para que no pueda ser de nuevo asignado, cuando asignes un paciente a una hora también se debe comprobar que dicho paciente no tiene una cita ese día y a esa hora con otro médico, si es así sacaremos un mensaje para informar de ese hecho.
    - Si **anulamos** una cita seleccionada, tendremos que discriminar si era una cita ya grabada anteriormente, o se acaba de asignar dicha cita y no está

grabada todavía en el fichero, en cualquiera de los dos casos el paciente debe aparecer de nuevo en la lista para poder asignarle a otra hora una cita con este médico.

- Cuando se pulse el botón **aceptar** recorrerá la tabla y se grabarán las citas que no estén ya grabadas, para no repetir estas.

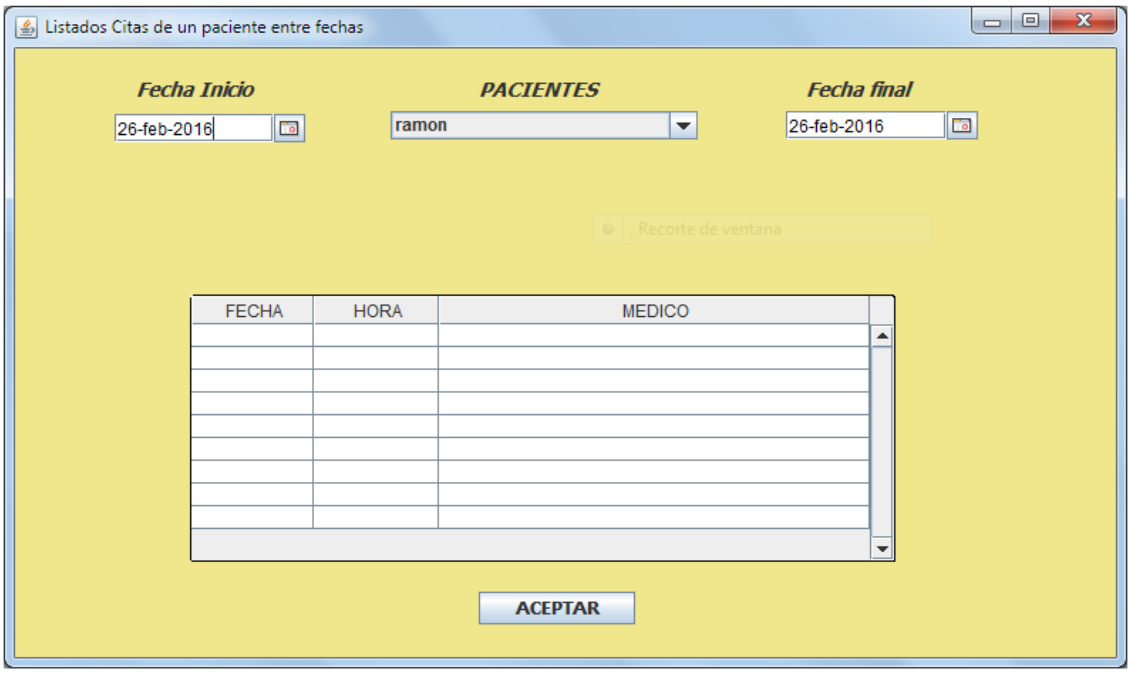


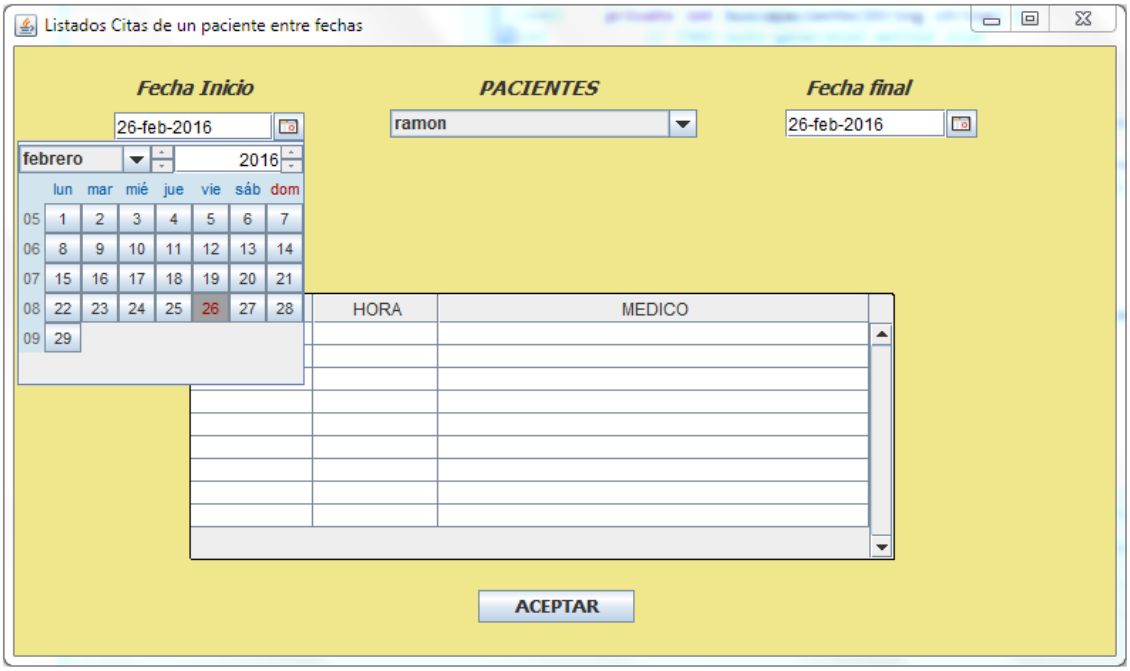




- Listado de Citas de un Paciente por fechas.

Se elegirán las fechas de inicio y final del listado, y del combo que se habrá cargado con los nombres de los pacientes, el paciente del que se quiere obtener dicho listado. Sacando los datos que se muestran en la pantalla siguiente.





Elabora el proyecto con todas las clases necesarias, mandando todos los archivos *.java, y un documento con un procesador de texto. El documento debe ser de tipo ".doc" (Microsoft Word) o de tipo ".odt" (OpenOffice.org). En el documento escribirás un informe sobre todas las consideraciones oportunas que se necesiten para entender cómo has realizado la tarea.

Recomendaciones:

- Básate para la realización de la aplicación en la tarea de apoyo de la unidad 6 (aprovechando la clase donde se realiza la interfaz de la clase ObjectOutputStream…,MiObjectOutputStream).
- La aplicación constará de las clases: Medico, Especialidad, Paciente, Citas, Menu, y una clase con su JFrame para cada una de las opciones del menú

ACTIVIDAD 7

A lo largo de esta unidad has terminado de familiarizarte con el resto de conceptos relacionados con la **Programación Orientada a Objetos** que faltaban por ver de una manera más formal y con ejemplos explícitos: **composición**; **herencia**; **clases y métodos abstractos**; **sobrescritura de métodos**; **interfaces**; **polimorfismo**; **ligadura dinámica**, etc.

Has experimentando con todos estos conceptos y los has utilizado en pequeñas aplicaciones para comprobar su funcionamiento y su utilidad.

Una vez finalizada la unidad se puede decir que tienes ya un dominio adecuado del lenguaje Java como un lenguaje que permite aplicar todas las posibilidades de la **Programación Orientada a Objetos**. Dado ese supuesto, esta tarea tendrá como objetivo escribir una pequeña aplicación en Java empleando algunas de las construcciones que has aprendido a utilizar.

Se trata de desarrollar una aplicación Java que permita gestionar varios tipos de **cuentas bancarias**. Mediante un menú se podrán elegir determinas operaciones:

1. **Abrir una nueva cuenta**.
2. **Ver un listado de las cuentas disponibles** (código de cuenta, titular y saldo actual).
3. **Obtener los datos de una cuenta concreta**.
4. **Realizar un ingreso en una cuenta**.
5. **Retirar efectivo de una cuenta**.
6. **Consultar el saldo actual de una cuenta**.
7. **Salir de la aplicación.**

Las cuentas se irán almacenando en alguna estructura en memoria según vayan siendo creadas. Cada cuenta será un objeto de una clase que contendrá la siguiente información:

- **Titular** de la cuenta (un objeto de la clase **Persona**, la cual contendrá información sobre el titular: **nombre**, **apellidos**, **fecha de nacimiento**).
- **Saldo** actual de la cuenta (número real).
- **Número de cuenta** (**CCC - Código Cuenta Cliente**).
- **Tipo de interés** anual (si se trata de una **cuenta de ahorro**).
- **Lista de entidades** autorizadas para cobrar recibos de la cuenta (si se trata de una **cuenta corriente**).
- **Comisión de mantenimiento** (para el caso de una **cuenta corriente personal**).
- **Tipo de interés por descubierto** (si es una **cuenta corriente de empresa**).

- **Máximo descubierto permitido** (si se trata de una **cuenta corriente de empresa**)

Las **cuentas bancarias** pueden ser de dos tipos: **cuentas de ahorro** o bien **cuentas corrientes**. Las **cuentas de ahorro** son **remuneradas** y tienen un determinado **tipo de interés**. Las **cuentas corrientes** no son remuneradas, pero tienen asociada una **lista de entidades autorizadas** para cobrar **recibos domiciliados** en la cuenta.

Dentro de las **cuentas corrientes** podemos encontrar a su vez otros dos tipos: las **cuentas corrientes personales**, que tienen una **comisión de mantenimiento** (una cantidad fija anual) y las **cuentas corrientes de empresa**, que no la tienen. Además, las **cuentas de empresa** permiten tener una cierta cantidad de **descubierto** (**máximo descubierto permitido**) y por tanto un **tipo de interés por descubierto** y una **comisión fija** por **cada descubierto** que se tenga. Es el único tipo de cuenta que permite tener **descubiertos**.

Cuando se vaya a abrir una nueva **cuenta bancaria**, el **usuario de la aplicación** (**empleado del banco**) tendrá que solicitar al **cliente**:

- **Datos personales**: **nombre**, **apellidos**, **fecha de nacimiento**.
- **Tipo de cuenta** que desea abrir: **cuenta de ahorro**, **cuenta corriente personal** o **cuenta corriente de empresa**.
- **Saldo inicial**.

Además de esa información, el **usuario de la aplicación** deberá también incluir:

- **Número de cuenta** (**CCC**) de la nueva cuenta. Debe ser válido (habrá que comprobarlo).
- **Tipo de interés** de remuneración, si se trata de una **cuenta de ahorro**.
- **de mantenimiento**, si es una **cuenta corriente personal**.
- **Máximo descubierto permitido**, si se trata de una **cuenta corriente de empresa**.
- **Tipo de interés por descubierto**, en el caso de una **cuenta corriente de empresa**.
- **Comisión fija por cada descubierto**, también para el caso de una **cuenta corriente de empresa**.

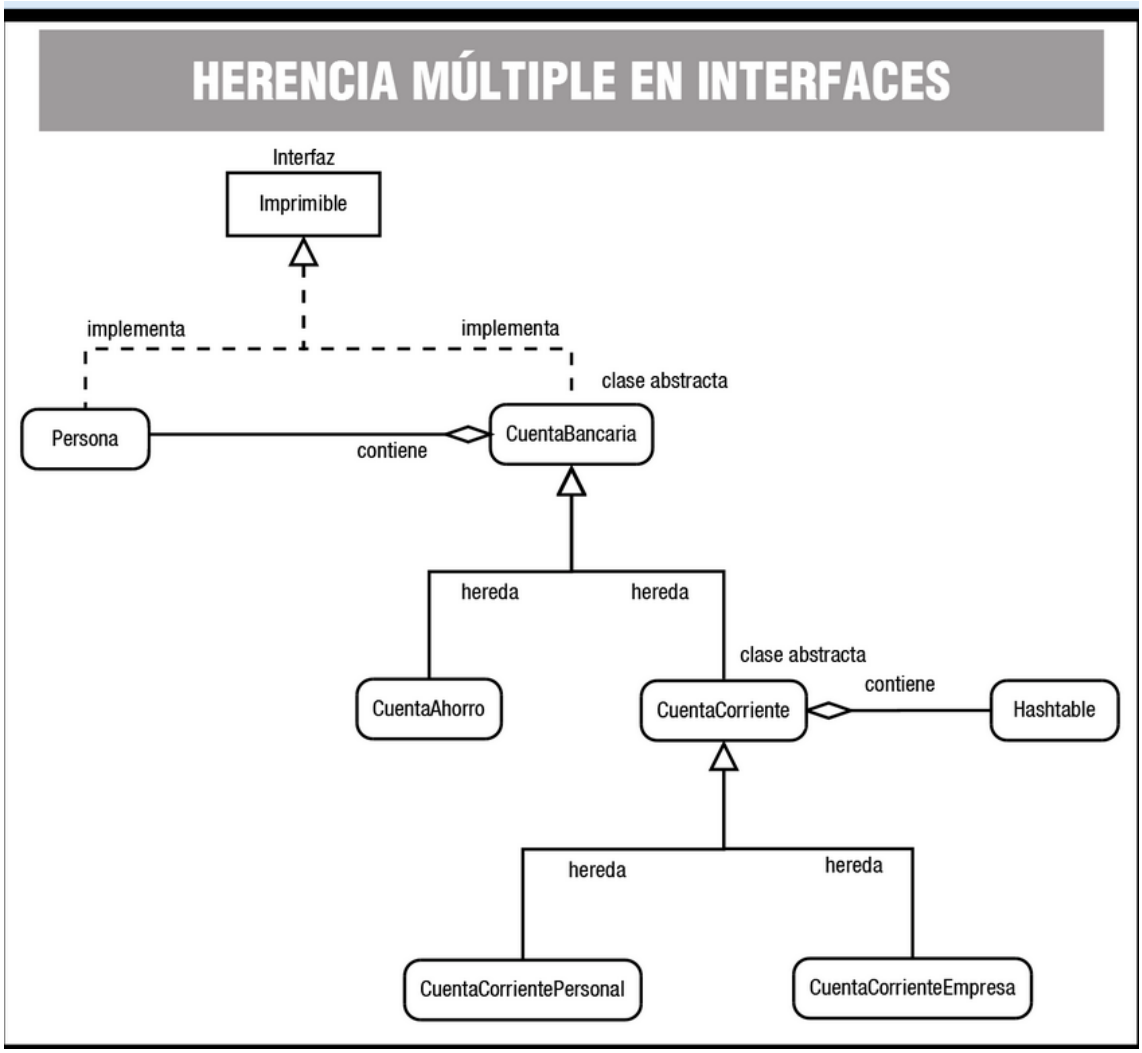
La aplicación deberá asegurarse que la información introducida sea **válida y coherente** (CCC válido; saldos, comisiones y tipos de interés positivos, etc.).

El programa que escribas debe cumplir al menos los siguientes requisitos:

- Para almacenar los objetos de tipo cuenta podrás utilizar cualquier estructura de almacenamiento que consideres oportuna (`ArrayList, Hashtable`, etc.).
- Para trabajar con los datos personales, debes utilizar una clase **Persona** que contenga la información sobre los datos personales básicos del cliente (**nombre**, **apellidos**, **fecha de nacimiento**).
- Para trabajar con el número de cuenta debes utilizar el modelo de **Código Cuenta Cliente** (**CCC**), que es posible que también la ya hayas usado en otras unidades.

- Para guardar las entidades autorizadas a cobrar recibos debes utilizar una `Hashtable` que contenga pares de tipo (código de entidad (`String`), máxima cantidad autorizada para un recibo).

Aquí tienes un ejemplo de una posible estructura de clases para llevar a cabo la aplicación:



El **código fuente** Java de **cada clase** debería incluir **comentarios** en cada **atributo** (o en cada conjunto de atributos) y **método** (o en cada conjunto de métodos del mismo tipo) indicando su **utilidad**. El **programa principal** (**clase principal**) también debería incluir algunos comentarios explicativos sobre su funcionamiento y la utilización de objetos de las distintas clases utilizadas.

Además del programa deberás escribir también un **informe** con todas las consideraciones oportunas que se necesiten para entender cómo has realizado la tarea.

El proyecto deberá contener al menos los siguientes archivos fuente Java:

- **Programa principal** (clase con método `main`: **AplicacionCuentaBancaria.java**).
- Un archivo por cada clase o interfaz que hayas implementado.

El documento que contendrá el informe lo elaborarás con un procesador de texto. Debe ser de tipo ".doc" (Microsoft Word) o de tipo ".odt" (OpenOffice.org). Debe tener tamaño de página A4, estilo de letra Times New Roman, tamaño 12 e interlineado normal.

Criterios de puntuación. Total 10 puntos.

Para poder empezar a aplicar estos criterios es necesario que la aplicación compile y se ejecute correctamente en un ordenador. En caso contrario la puntuación será directamente de 0,00.

Criterios de puntuación.	
Existe una clase `CuentaBancaria` base que proporciona los **atributos** y **métodos** necesarios para cualquier tipo de cuenta bancaria genérica. La clase funciona correctamente.	2,00
Existe una clase `CuentaAhorro` que proporciona los **atributos** y **métodos** necesarios para trabajar con una **cuenta de ahorro**. La clase funciona correctamente.	2,00
Existe una clase `CuentaCorrientePersonal` que proporciona los **atributos** y **métodos** necesarios para trabajar con una **cuenta de corriente personal**. La clase funciona correctamente.	2,00
Existe una clase `CuentaCorrienteEmpresa` que proporciona los **atributos** y **métodos** necesarios para trabajar con una **cuenta de corriente de empresa**. La clase funciona correctamente.	2,00
Las clases de tipo **cuenta corriente** disponen de una **colección** que contiene la **lista de entidades autorizadas** para cobrar recibos en esa cuenta.	1,00
Se utiliza algún tipo de **colección** (`Hashtable`, `ArrayList`, etc.) para manipular las cuentas que se van creando durante la ejecución del programa.	1,00
La **información de un titular** no es almacenada en objetos de la clase **Persona** que a su vez son almacenados dentro de los objetos de cuenta bancaria.	-1,00
Las clases no son capaces de **validar** y **gestionar correctamente un CCC**.	-1,00
Los métodos de las clases no son capaces de lanzar **excepciones** si se produce alguna situación anómala.	-1,00
Se utiliza el polimorfismo y la ligadura dinámica para trabajar con las cuentas bancarias y funciona correctamente.	2,00
No se han incluido **comentarios** en las clases tal y como se ha pedido en el enunciado.	-1,00
No se han incluido **comentarios** apropiados en el **programa principal** describiendo el funcionamiento de éste.	-1,00

No se ha entregado el informe explicativo.	-2,00
El programa principal **no** es capaz de crear objetos de alguno de los tres tipos de cuentas bancarias solicitados (**cuenta de ahorro, cuenta corriente personal** o **cuenta corriente de empresa**).	-5,00
Alguna de las opciones de menú pedidas en el enunciado (menú del programa principal) **no** funciona correctamente.	-1,00 por cada opción
Total (máximo)	**10,00**

Dado que algunos criterios de puntuación son negativos, podría suceder que el balance final fuera negativo. En tal caso la puntuación final será simplemente de 0,00. Si el balance final es positivo y mayor que 10,00, la puntuación final quedará como 10,00.

Recursos necesarios para realizar la Tarea.

- Ordenador personal.
- JDK y JRE de Java SE.
- Entorno de desarrollo NetBeans con las funcionalidades necesarias para desarrollar y emular aplicaciones Java.

Consejos y recomendaciones.

Para realizar la aplicación te sugerimos las siguientes recomendaciones:

- Básate en los diferentes ejemplos que has tenido que probar durante el estudio de esta unidad y de unidades anteriores. Algunos de ellos te podrán servir de mucha ayuda, así que aprovéchalos.
- Puedes obtener información sobre del funcionamiento del CCC y cómo calcular los dígitos de control del siguiente artículo de Wikipedia:

  Wikipedia: Código Cuenta Cliente.

- Puedes generar cuentas bancarias válidas (o comprobarlas) para hacer pruebas en tu programa desde el siguiente enlace:

  Generador/validador de cuentas bancarias.

ACTIVIDAD 8

Detalles de la tarea de esta unidad.

Enunciado.

Se trata de hacer una aplicación en Java que acceda a una base de datos MySQL de una notaría y mediante un menú pueda realizar determinadas operaciones. La base de datos a la que accederemos, por comodidad, será la correspondiente al esquema test, que ya viene por defecto en la instalación de MySQL, y ahí crearemos las tablas necesarias.

Puedes utilizar un `JOptionPane` para presentar el menú con las opciones que permita realizar el programa:

- Crear tabla `Clientes`.
  - Que contendrá los campos: `Cod_Cliente`, `Nombre` y `Teléfono del mismo`.
- Crear tabla `Escrituras`. Que contendrá los siguiente campos:
  - `Código`, que representa el código único de escritura.
  - `Tipo`, representa el tipo de escritura, almacenándose valores tales como: "TEST" para testamento, "CPVE" para compraventa, etc.
  - `Nom_fich`, corresponde al nombre del fichero donde se guarda el texto íntegro de la escritura, de modo que tomará valores como: "`compraventa_AntonioBJ_10_12_2010.doc`".
  - `Num_interv`, que representa el número de intervinientes, o sea, de las personas que involucra la escritura, como 2 compradores y un vendedor, por tanto, serían 3 los intervinientes.
- Crear la tabla `EscCli`, que sirve para relacionar las escrituras, con los intervinientes que participan en ella, por lo que contendrá los campos:
  - `codCli`
  - `codEsc`
- Insertar datos en la tabla `Clientes`.
  - Inserta al menos tres clientes en la tabla de `Clientes`.
- Insertar datos en la tabla `Escrituras` y `EscCli`.
  - Inserta algunos datos para que haya clientes con escrituras.
- Recuperar datos de la tabla `Clientes`.
  - Recupera los datos de la tabla `Clientes` y los muestra al usuario.
- Recuperar datos de la tabla `Escrituras`.
  - Recupera los datos de la tabla `Escrituras` y los muestra al usuario.
- Actualizar en tabla `Clientes`.
  - Realizar una consulta de actualización, "`Update`", sobre un registro previamente insertado, por ejemplo cambiar el teléfono a un cliente, o el nombre.
- Listar el nombre de los clientes que hayan realizado una compraventa (tipo de escritura CPVE).

Elabora el programa y si además de esas opciones que se piden, realizas alguna o algunas otras, entonces elabora también un documento con un procesador de texto. El documento debe ser de tipo ".doc" (Microsoft Word) o de tipo ".odt" (OpenOffice.org). El documento debe tener tamaño de página A4, estilo de letra Times New Roman, tamaño 12 e interlineado normal.

En el documento escribirás un informe sobre las opciones adicionales que hayas incluido con las consideraciones oportunas que se necesiten para entender su utilidad.

Además deberás entregar el archivo SQL que hayas exportador de la Bases de Datos, para que pueda ejecutar el programa con tu base de datos.

Una vez que tengas terminados el programa y el documento explicativo (si lo has elaborado) y, el archivo sql , comprime todas las cosas en un único fichero. El nombre del fichero debe ser del siguiente estilo:

"`PROG_T11_Apellido1Apellido2Nombre.zip`". Por ejemplo para el alumno Juan Cruz García, el fichero se llamará:

```
PROG_T11_CruzGarciaJuan_E1.zip
```

No utilices espacios en blanco, la letra ñ o tildes en el nombre de archivo. Envía el fichero al buzón de tareas para que la corrija tu tutor del módulo.

Criterios de puntuación. Total 10 puntos.

La tarea es la actividad más importante de la unidad, por tanto debemos transmitir al alumnado la trascendencia de responder y elaborar dicha actividad con el máximo de rigor e interés ya que esta es el principal mecanismo de consolidación de su aprendizaje.

La tarea se evalúa valorándola con 10 puntos.

Recursos necesarios para realizar la Tarea.

Es suficiente con los contenidos y los enlaces que se proporcionan en la propia unidad. Ningún recurso adicional es necesario.
Aunque puedes utilizar Xampp o WampServer para gestionar el servidor local y la base de datos.