EJERCICIOS BBDD

Ejercicio1. Base de datos MySQL <world>

MySQL ofrece algunas bases de datos de prueba. En nuestro caso, vamos a crear un programa para manejar la base de datos <world>, que contiene información sobre ciudades y países del mundo.

- Descargar la base de datos desde (https://dev.mysql.com/doc/index-other.html) y instalarla en local.
- Descargar el código fuente desde el Aula Virtual.

El programa esta formado por dos clases, la clase BD_Word y el main. La primera contiene la conexión a la base de datos y métodos para ejecutar las consultas. Hemos hecho todos los métodos de tipo estático para poder llamarlos desde cualquier punto. Podemos hacer esto pues solo vamos a tener una instancia de dicha clase y no vamos a necesitar crear varias objetos de este tipo. Y ganamos en sencillez de uso.

El programa tiene el siguiente menú:

1. Habitantes de paises
2. Habitantes de ciudades
3. Paises con un mínimo de X habitantes
4. Ciudades con un mínimo de X habitantes
5. Salir

Tarea1: probar el programa y ver como está programado

Tarea2: añadir una opción al menú para mostrar los habitantes de las ciudades ordenador de mayor a menor número de habitantes. Incluir el nombre del país en la salida. Además, pedir por teclado si queremos limitar la salida de la consulta. Es decir, si decimos limite=10, mostrara los diez primeros. Si decimos limite=0, mostrará todo.

Ejemplo: `Indica numero de filas a mostrar [0 para todas]: 5`

name	population	Name
Mumbai (Bombay)	10500000	India
Seoul	9981619	South Korea
São Paulo	9968485	Brazil
Shanghai	9696300	China
Jakarta	9604900	Indonesia

Ejercicio2. Agenda con base de datos MySQL <BD Agenda>

Hacer una agenda de contactos usando una base de datos relacional local (es decir, en tu ordenador) para almacenar los contactos. Crear una base de datos llamada *Agenda* y una tabla que sea C*ontactos* con los campos Nombre, Apellidos, email y fecha de nacimiento (este último de tipo Date).

La base de datos y la tabla las debes crear tu. La fecha de nacimiento la pedirás en formato DD/MM/AAAA. No vamos a definir ninguna clase en el programa, por lo que trabajaremos directamente con la base de datos como en el ejercicio anterior.

Hacer un menú con las siguientes opciones y con la lista completa al principio:

```
**************************** AGENDA ************************************
1    Mortadelo   Mortadelo   mortadelo@kk.com    01/01/1900
2    Filemon     Filemon     filemon@kk.com      05/01/1900
3    Juan        Perez Cano  juan@kk.com         08/11/1979
4    Celia       Perez Cano  celia@kk.com        21/01/1986
************************************************************************
```

1. Insertar contacto
2. Borrar contacto
3. Listar contactos por letra comienzo
4. Listar contactos por mes
5. Salir

NOTA: Para mostrar la lista de contactos y la fecha, por defecto se mostrara en formato AAAA-MM-DD. Usaremos la función de MySQL `DATE_FORMAT(…)`

```
select nombre,apellidos,email, DATE_FORMAT(fecha_nac,'%d/%m/%Y')
as FechaNac from contactos
```

NOTA2: MySQL necesita meter los *dates* en formato AAAA-MM-DD. Casualmente, cuando imprimimos una fecha de tipo LocalDate sin usar ningún formato específico (dtf), se imprime en el formato AAAA-MM-DD, por lo que MySQL la va a aceptar.

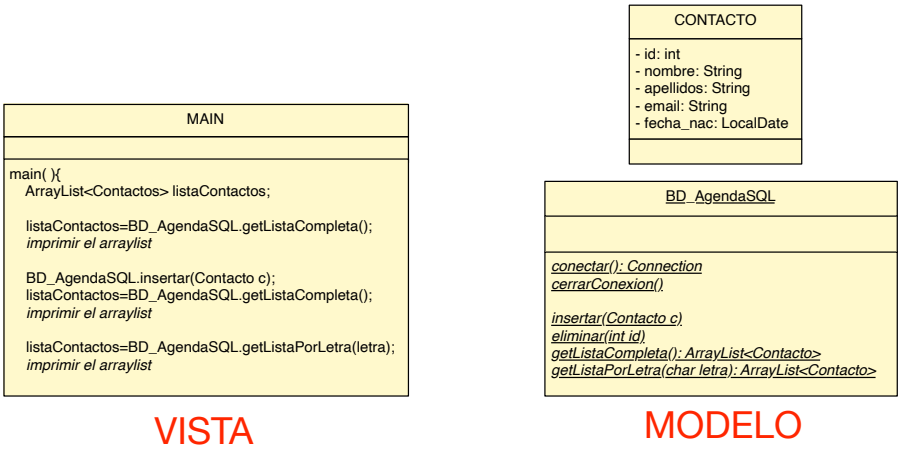
NOTA3: Para hacer el apartado 3 y 4 tenemos dos opciones:

- opción1: pedir la lista completa de contactos (select * from contactos) y luego con JAVA programar los que empiezan por una letra o los que son de un mes escogido.
- opción2 (la buena): usar la fuerza de SQL y lanzar la consulta directamente que te muestra solo los contactos que cuyo nombre empieza por una letra o cuyo mes es el dado. Esta opción es más eficiente pues la base de datos nos va a devolver solamente las filas que nos interesan.

Ejercicio3. Agenda con base de datos MySQL <BD Agenda> y clases

En el ejercicio anterior, no usamos ninguna clase, pero en un programa más serio y extenso, los datos se guardan en la base de datos pero se manejan por medio de objetos. Por ejemplo, a la hora de pedir los contactos a la base de datos, esta me debería devolver un arraylist de *Contactos* y yo posteriormente los imprimiría (en el ejercicio anterior, el método donde hacemos el SELECT directamente imprime los contactos, algo que no es lo normal. El método me debe devolver los contactos y yo ya los imprimo).

Vamos a repetir el ejercicio, cambiando ligeramente el menú, pero esta vez añadiendo la clase Contacto. El resultado aparentemente será el mismo, solo que está diseñado de forma diferente. En este ejercicio, la presentación de los datos se hace en el main. Estamos separando la *vista* de los datos del control de los mismos.



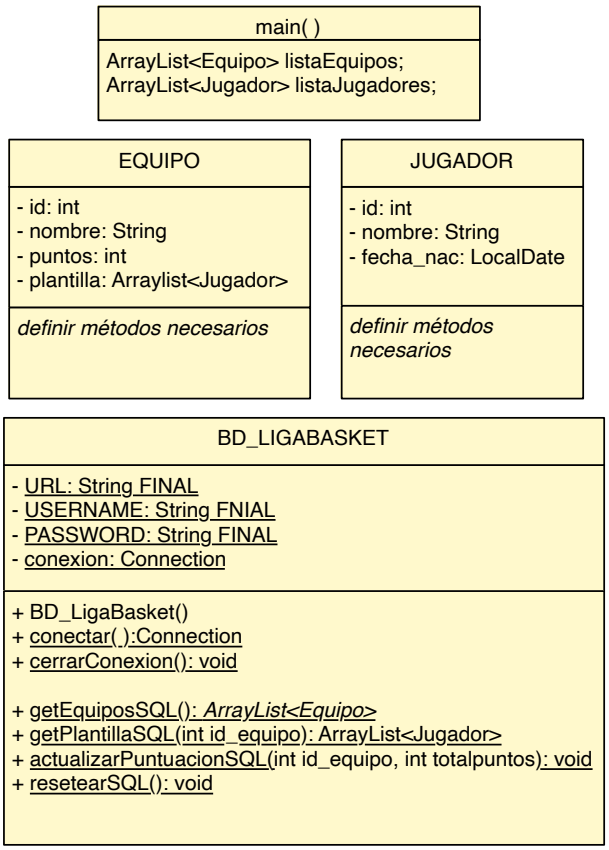
Eliminaremos la opción del menú de listar por mes, dejando solo la de filtrar por letra. Mostraremos antes del menú la lista de contactos, bien la completa o bien la filtrada por letra.

```
==================== LISTA DE CONTACTOS ======================
  1 - Mortadelo  Mortadelo       mortadelo@kk.com     01/01/1900
  3 - Juan       Perez Cano      juan@kk.com          08/11/1979
  4 - Celia      Perez Cano      celia@kk.com         21/01/1986
 14 - Maria      Alarcon         reyes@mail.com       22/12/1956
 16 - Javier     Lopez Cano      javi@gmail.com       23/02/2000
 17 - Josefa     Ruiz Ruiz       josefa@arrakis.es    03/05/1976
================================================================
1. Insertar Contacto
2. Borrar Contacto
3. Lista completa
4. Filtrar por letra comienzo
5. Salir
Escoge Opcion >
```

```
================= LISTA FILTRADA (Letra: j) ==================
  3 - Juan       Perez Cano      juan@kk.com          08/11/1979
 16 - Javier     Lopez Cano      javi@gmail.com       23/02/2000
 17 - Josefa     Ruiz Ruiz       josefa@arrakis.es    03/05/1976
================================================================
1. Insertar Contacto
2. Borrar Contacto
3. Lista completa
4. Filtrar por letra comienzo
5. Salir
Escoge Opcion >
```

Ejercicio4. Base de datos MySQL <ligabasket>

Hacer una aplicación para gestionar los puntos de una competición de baloncesto. En dicha competición hay 3 equipos y cada equipo tiene una plantilla de 2 jugadores ya asignados. Se proporciona una base de datos con la lista de equipos y con los jugadores fichados.




```
     LIGA BASKET
*******************************
[  1] Lorca        0 puntos
[  2] Aguilas      0 puntos
[  3] Pulpi        0 puntos
*******************************

1. Mostrar plantillas
2. Añadir puntos a equipo
3. Resetear puntos
4. Salir
Elige una opcion >
```

```
     LIGA BASKET
*******************************
[  3] Pulpi       10 puntos
[  1] Lorca        5 puntos
[  2] Aguilas      3 puntos
*******************************
```

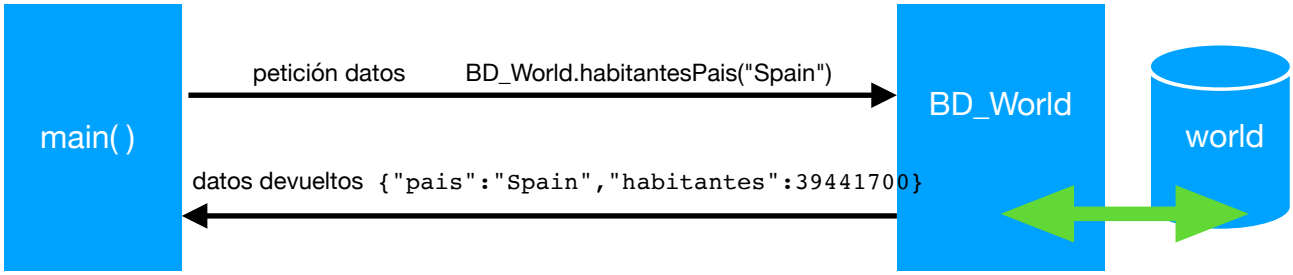
SE LIBRE DE AÑADIR MÉTODOS QUE NECESITES

```
Plantilla del Lorca
================================
     Marc Gasol  37 años (29/01/1985)
      Pau Gasol  41 años (06/07/1980)

Plantilla del Aguilas
================================
   Lebron James  37 años (30/12/1984)
 Michael Jordan  59 años (17/02/1963)

Plantilla del Pulpi
================================
  Magin Johnson  62 años (14/08/1959)
Saquille O_Neal  50 años (06/03/1972)
```

- Mostraremos las plantillas de todos los equipos indicando nombre, edad y fecha de nacimiento de cada jugador de la plantilla (usar *ChronoUnit* para calcular edad).
- La lista de equipos aparecerá en la cabecera siempre ordenada por puntuación.
- Resetar puntos pone a 0 los puntos de todos los equipos. Antes de hacer el reseteo, pedir confirmación del mismo, por seguridad.

Ejercicio5. Base de datos MySQL <world> (ejercicio1) usando JSON

Vamos a modificar el primer ejercicio usando JSON. La idea es en lugar de que la clase BD_World imprima los datos, que los devuelva en formato JSON.



Cada opción del menú, como devuelve datos con campos diferentes, devolverá un tipo json diferente.

1. Habitantes de paises →`{"pais":value, "habitantes":value}`
2. Habitantes de ciudades →`{"ciudad":value,"pais":value, "habitantes":value}`

…y así con el resto de opciones del menú.

MEJORA

A la hora de mostrar los resultados, tenemos que programar por cada opción del menú un algoritmo para imprimir los datos. Vamos a crear una función que te imprima un JSONObject sin saber de antemano los campos que tiene, y poder llamarla en cada opción del menú.

`imprimirJSONObject(JSONObject):void`

Al hacerlo, no te preocupes si salen los campos desordenados, el objetivo es conseguir imprimir toda la información del objeto JSON

PISTA: busca algún método de la clase JSONObject que te devuelva el conjunto de campos.