PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

LENGUAJE DE PROGRAMACIÓN 2
3era. práctica (tipo b)
(Segundo Semestre 2022)

Indicaciones Generales:

- Tiempo estimado: 1h 50 minutos
- Se les recuerda que, de acuerdo al reglamento disciplinario de nuestra institución, constituye una falta grave copiar del trabajo realizado por otro estudiante o cometer plagio para el desarrollo de esta práctica.
- Puede hacer uso del entorno de desarrollo integrado de NETBEANS (a excepción de la pregunta Nro. 2 y 3, las cuales deben realizarse mediante línea de comandos y uso de editor de texto: Notepad++ o Sublime).
- Está permitido el uso de apuntes de clase, diapositivas y ejercicios de clase.
- Está permitido el uso de Internet (páginas de documentación oficial de Microsoft y Oracle).
- Está prohibida toda forma de comunicación con terceros.

PARTE PRÁCTICA (20 puntos)

PUEDE UTILIZAR MATERIAL DE CONSULTA.
Antes de comenzar el laboratorio, descargue todos los proyectos, apuntes, diapositivas que utilizará.
Se considerará en la calificación el uso de buenas prácticas de programación (aquellas vistas en clase).

PREGUNTA 1: Creación de un motor de base de datos MySQL en AWS Academy (3 puntos)

Se le solicita ingresar al servicio de AWS Academy (https://www.awsacademy.com/LMS_Login) y crear una instancia o motor de base de datos MySQL con los siguientes parámetros:

- Método de creación de base de datos: **Creación estándar.**
- Tipo de motor: **MySQL.**
- Edición: **Community (Comunidad).**
- Versión: **8.0.28**
- Plantilla: **Capa Gratuita**
- Identificador de instancias de base de datos: **lab3-lp2-2022-2**
- Nombre de usuario maestro: **administrador**
- Contraseña maestra: **20222lp2**
- Instancia de clase de BD: **db.t2.micro (1 CPU, 1 GB RAM)**
- Capacidad de disco duro para la BD: **20 GB sin posibilidad de auto-escalado.**
- Acceso Público: **SI**
- Puerto: **3306**
- Nombre de la base de datos inicial (esquema): **lab03**

- **Configure las reglas de entrada para que cualquier IP o computador tenga acceso a su base de datos.**
- **IMPORTANTE: El motor de base de datos debe ser configurado para que cualquier computador o IP pueda acceder al mismo. Si esta configuración no es realizada y no es posible conectarse a su motor de base de datos desde la computadora de los jefes de práctica o docente, entonces no se considerará puntaje alguno.**

Una vez creada su base de datos, se le solicita comprobar que funciona correctamente. Se le solicita dejar la base de datos **ACTIVA** en AWS y colocar los datos de conexión en un archivo llamado **datosConexion_códigoAlumno.txt** (en el archivo deberá colocar el *hostname*, el *username*, el *password* y el nombre de la BD o esquema).

PREGUNTA 2: Ejercicio de paquetes (Sin uso de IDE) (4 puntos)

Como parte de un proceso de reingeniería, se ha establecido modificar la estructura de paquetes inicial de las clases del software que ha desarrollado en el Laboratorio 01.
En la Fig. 01, se muestra a través de un diagrama de clases, la nueva estructura de paquetes.
Se le solicita descargar los archivos fuente que se adjuntan a este enunciado (no aquellos que se encuentran en la Semana 02). Agregue las instrucciones **package** e **import** a todas las clases, así como la estructura de carpetas para que el programa en JAVA funcione bajo la nueva estructura de paquetes solicitada. Compruebe que es posible compilar y ejecutar bajo la nueva estructura. Suba un único archivo .zip a PAIDEIA que contenga TODA esta nueva estructura (con todos los paquetes que se visualizan en el diagrama). **Está prohibido utilizar el comodín * para realizar importaciones en las clases. Para compilar sí puede utilizar el *.**

Se descontarán puntos por importaciones innecesarias.

Para un mejor entendimiento, a continuación, se detallan los paquetes y las clases que deberían contener:

- **com.librarysoft.logistica.model**: IConsultable, Biblioteca, Cubiculo, Reserva, MiembroPUCP.
- **com.librarysoft.personal.model**: Categoria, Profesor, Egresado, Estudiante.
- **com.librarysoft.main**: Principal.

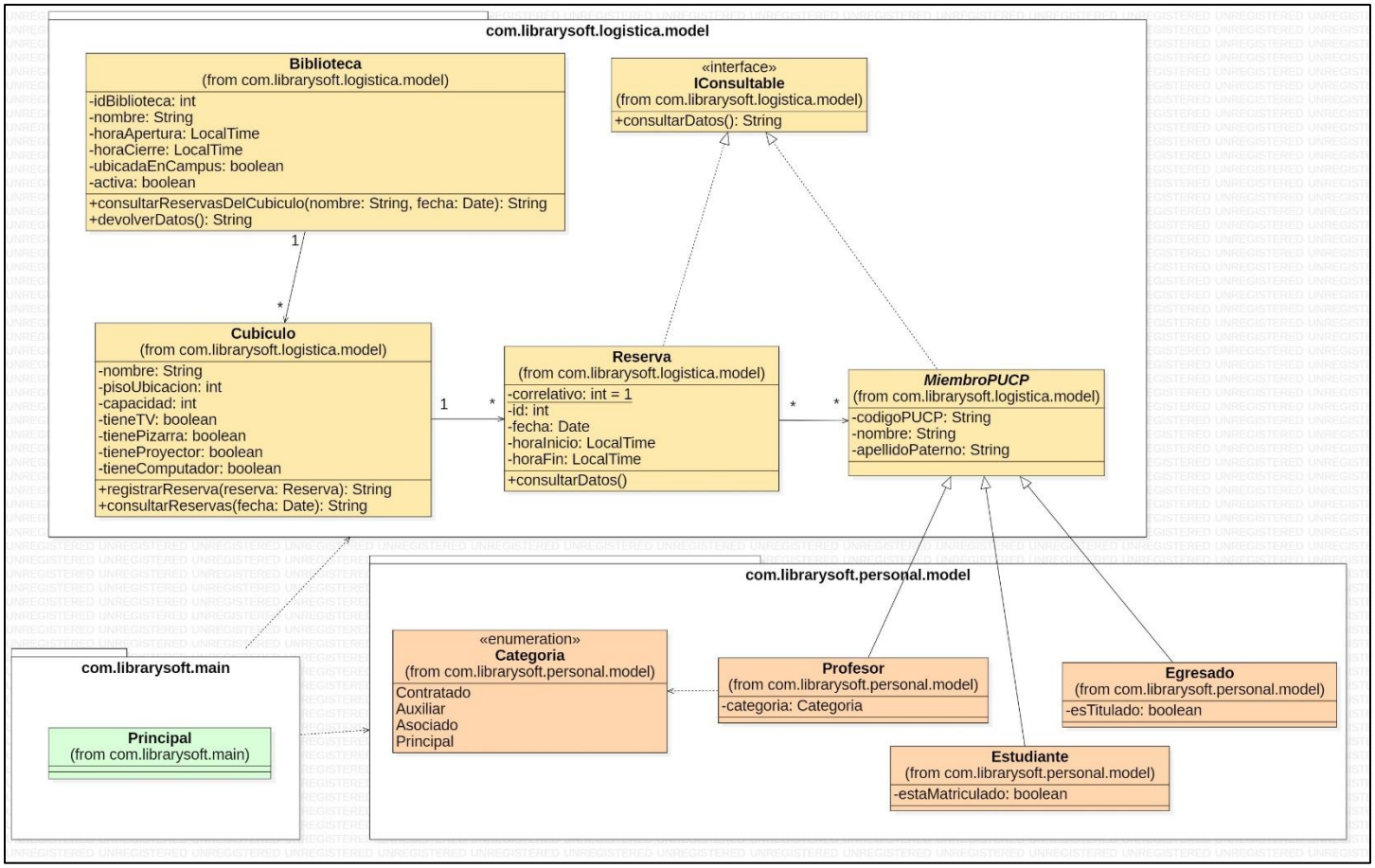

Fig. 01. Diagrama de clases – estructura de paquetes solicitada

Con las siguientes instrucciones puede corroborar que su programa se ha estructurado y funciona bajo el nuevo esquema:

```
javac com/librarysoft/logistica/model/*.java com/librarysoft/personal/model/*.java
com/librarysoft/main/*.java

java -cp com/librarysoft/logistica/model/*.class;com/librarysoft/personal/model/*.class;
com/librarysoft/main/Principal
```

PREGUNTA 3: Ejercicio de generación de librerías JAR (Sin uso de IDE) (4 puntos)

Para continuar con este ejercicio,

borre todos los archivos .class generados en el ejercicio anterior en todos los paquetes.

Realice las siguientes instrucciones en estricto orden e indique en un archivo llamado “Pregunta03_codigoAlumno.txt”, las líneas de comandos utilizadas por consola para alcanzar cada actividad:

1. Compile únicamente las clases contenidas en **com.librarysoft.logistica.model** desde el directorio raíz de trabajo.
2. Genere un componente .jar llamado “**softLogisticaModel.jar**” que contenga la estructura y clases en Bytecode contenidas en: **com.librarysoft.logistica.model.**

Una vez creado el componente, borre todos los archivos .java, .class, y estructura de carpetas que corresponden al componente creado “**softLogisticaModel.jar**” **(com.librarysoft.logistica)**. Prosiga con las siguientes instrucciones:

3. Compile las clases contenidas en **com.librarysoft.personal.model** desde el directorio raíz de trabajo utilizando el .jar previamente creado: “**softLogisticaModel.jar**”.

4. Genere un componente .jar llamado **softPersonalModel.jar** que contenga la estructura y clases en Bytecode contenidas en **com.librarysoft.personal.model**.

Una vez creado el componente, borre todos los archivos .java, .class y estructura de carpetas que corresponden al componente creado: "**softPersonalModel.jar**" (**com.librarysoft.personal)**. Prosiga con las siguientes instrucciones:

5. Compile las clases contenidas en **com.librarysoft.main** desde el directorio raíz de trabajo utilizando los .jar previamente creados: "**softLogisticaModel.jar**" y "**softPersonalModel.jar**".
6. Genere un componente .jar **ejecutable** llamado "**softPrincipal.jar**" que contenga la estructura y clases en Bytecode contenidas en **com.librarysoft.main. Compruebe que funcione.** Prosiga con la siguiente instrucción:
7. Además de las instrucciones ejecutadas, incorpore en el archivo "**Pregunta03_codigoAlumno.txt**" el contenido que debería tener el **MANIFEST** de "**softPrincipal.jar**" para que pueda funcionar como componente ejecutable.

Suba el archivo de texto o documento de texto con todas las instrucciones a PAIDEIA.
Además, adjunte también los archivos JARs.

PREGUNTA 4: Conexión a Base de Datos (Con uso de IDE) (9 puntos)

Desarrolle una solución de software en lenguaje JAVA utilizando Netbeans que permita registrar y listar bibliotecas desde una base de datos. Utilice el motor de base de datos que ha creado en la pregunta 1 y utilice el script que se encuentra en PAIDEIA (**ScriptLab03.sql**) que genera la tabla "biblioteca" en la BD. **Para el desarrollo de este ejercicio, debe utilizar la clase Statement o la clase PreparedStatement.** Cuando programe el listar bibliotecas debe extraer todos los atributos de la misma.

A continuación, se muestra la tabla de base de datos que se han diseñado para este propósito.

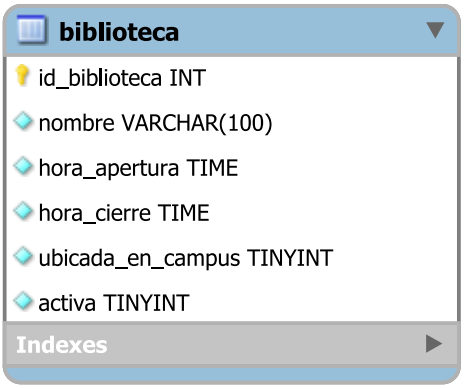

Fig. 02. Diagrama EER de la Base de Datos

Se le solicita crear 3 proyectos y utilizar el componente "**softLogisticaModel.jar**" que ha elaborado en la Pregunta 3.

- **LibrarySoftDBManager**: Que define la clase con los datos de conexión.
- **LibrarySoftLogisticaController:** Que define las clases de acceso a base de datos necesarias para realizar el registro y el listado de bibliotecas empleando el patrón DAO.
- **LibrarySoft:** Que contiene la clase Principal y permitirá mostrar un ejemplo de dos registros y el listado de bibliotecas en base de datos.

No es necesario implementar el modificar ni el eliminar biblioteca.
El componente "softLogisticaModel.jar" formará parte de los archivos de los proyectos "LibrarySoftLogisticaController" y "LibrarySoft". Asimismo, estará referenciado con ruta relativa.

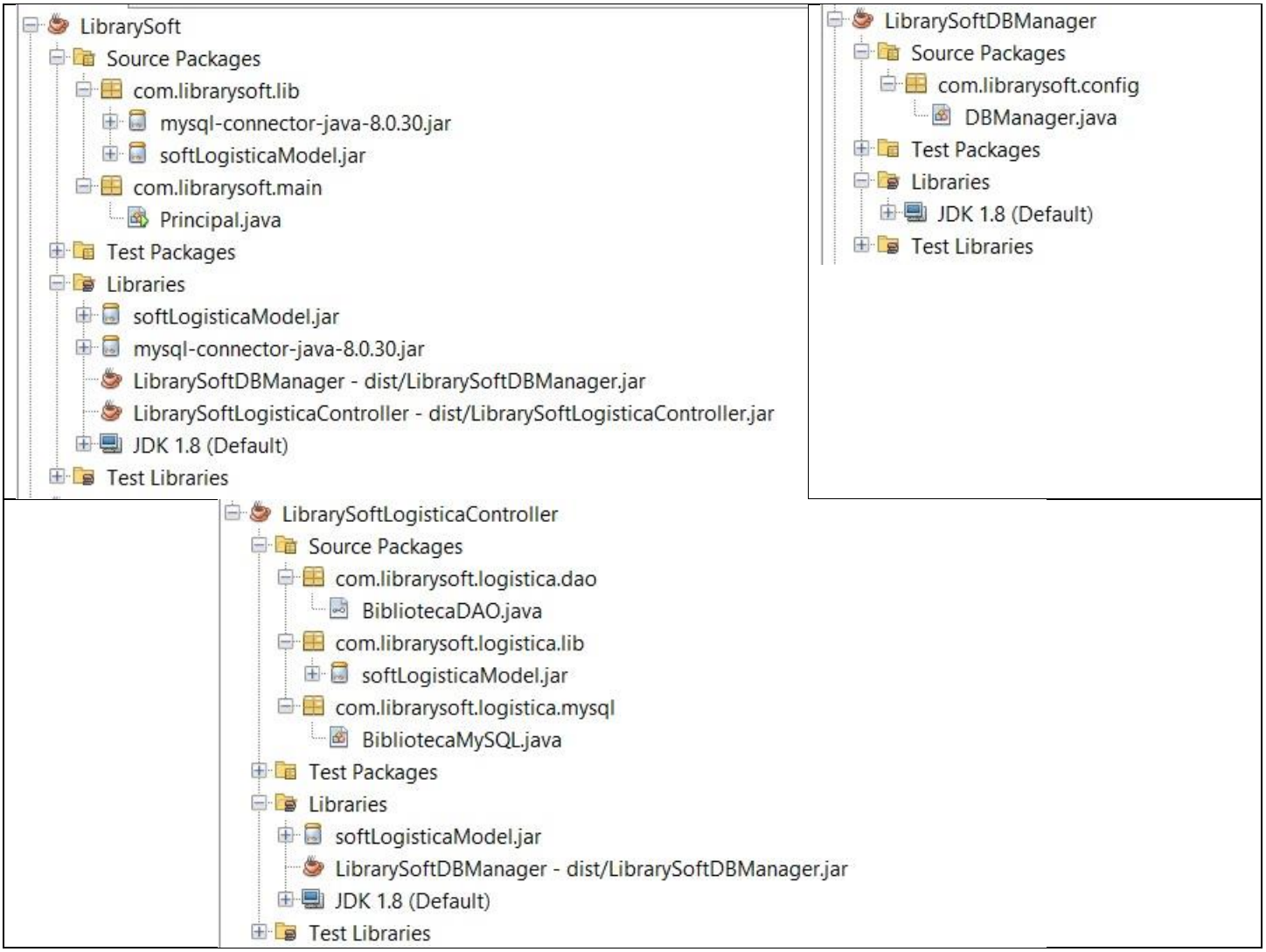

Fig. 03. Estructura de proyectos en JAVA

Para convertir de java.time.LocalTime a java.sql.Time utilice:

```
ps.setTime(2, java.sql.Time.valueOf(biblioteca.getHoraApertura()));
```

Para convertir de java.sql.Time a java.time.LocalTime utilice:

```
biblioteca.setHoraApertura(rs.getTime("hora_apertura").toLocalTime());
```

La clase **Principal** sería la siguiente:

```
package com.librarysoft.main;
import com.librarysoft.logistica.dao.BibliotecaDAO;
import com.librarysoft.logistica.model.Biblioteca;
import com.librarysoft.logistica.mysql.BibliotecaMySQL;
import java.time.LocalTime;
import java.util.ArrayList;
public class Principal {
    public static void main(String[] args){
        //Se generan los objetos
        Biblioteca bib01 = new Biblioteca("Biblioteca Central Luis Jaime
Cisneros",LocalTime.of(9,00),LocalTime.of(20,00),true);
        Biblioteca bib02 = new Biblioteca("Biblioteca del Instituto Riva-
Agüero",LocalTime.of(7,00),LocalTime.of(17,00),false);
        //Se crea el DAO de conexión
        BibliotecaDAO daoBiblioteca = new BibliotecaMySQL();
```

```
        //Se registran las bibliotecas
        int resultado;
        resultado = daoBiblioteca.insertar(bib01);
        System.out.println(evaluarResultado(resultado,bib01.getNombre()));
        resultado = daoBiblioteca.insertar(bib02);
        System.out.println(evaluarResultado(resultado,bib02.getNombre()));
        //Se consultan las bibliotecas desde base de datos
        ArrayList<Biblioteca> bibliotecas = daoBiblioteca.listarTodas();
        //Se imprimen
        for(Biblioteca bib : bibliotecas){
            System.out.println(bib.devolverDatos());
        }
    }

    public static String evaluarResultado(int resultado, String nombre){
        String cadena;
        if(resultado == 1)
            cadena = "Se ha registrado con éxito la biblioteca: " + nombre;
        else
            cadena = "Ha fallado el registro de la biblioteca: " + nombre;
        return cadena;
    }
}
```

La ejecución del programa genera la siguiente salida:

```
Output - LibrarySoft (run)
run:
Se ha registrado con éxito la biblioteca: Biblioteca Central Luis Jaime Cisneros
Se ha registrado con éxito la biblioteca: Biblioteca del Instituto Riva-Agüero
1. Biblioteca del Complejo de Innovación Académica, Hora Apertura: 08:00, Hora Cierre: 20:00 - Ubicada en Campus: SI
2. Biblioteca del Instituto Confucio, Hora Apertura: 10:00, Hora Cierre: 17:00 - Ubicada en Campus: NO
3. Biblioteca Central Luis Jaime Cisneros, Hora Apertura: 09:00, Hora Cierre: 20:00 - Ubicada en Campus: SI
4. Biblioteca del Instituto Riva-Agüero, Hora Apertura: 07:00, Hora Cierre: 17:00 - Ubicada en Campus: NO
BUILD SUCCESSFUL (total time: 12 seconds)
```

Suba en un archivo .zip todos los proyectos desarrollados a PAIDEIA.

Los siguientes aspectos pueden conllevar a un descuento significativo en su propuesta de solución:

- No seguir las instrucciones establecidas en el enunciado.
- Que los programas no compilen.
- Que los programas presenten errores en tiempo de ejecución.
- Que los proyectos tengan referencias innecesarias.
- Que el *driver* de conexión a base de datos no se haya adjuntado al proyecto.
- Que el *driver* de conexión a base de datos no se encuentre referenciado con ruta relativa en el proyecto.
- No seguir las prácticas vistas en clase: patrón DAO, desarrollo por componentes en capas, etc.

Profesor del Curso:
Dr. Freddy Paz

08 de septiembre del 2022