

**PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ**  
**FACULTAD DE CIENCIAS E INGENIERÍA**

**LENGUAJE DE PROGRAMACIÓN 2**

**4ta. práctica (tipo b)**  
**(Primer 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 un blog de notas).
- Está permitido el uso de apuntes de clase, diapositivas y ejercicios de clase.
- Está permitido el uso de Internet, pero está prohibida toda forma de comunicación con terceros.

**PARTE PRÁCTICA (20 puntos)**

PUEDEN UTILIZAR MATERIAL DE CONSULTA.

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 (2 puntos)**

**Importante:** Para la creación se le solicita utilizar la interfaz original ya que la nueva interfaz está generando demoras. Una vez ingresada a esta interfaz después de seleccionar MySQL se le solicita marcar la opción:

"Only enable options eligible for RDS Free Usage Tier"

☒ Only enable options eligible for RDS Free Usage Tier [Info](#)CancelNext

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

- Versión: **MySQL 8.0.28**
- Plantilla: **Capa Gratuita**
- 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.**
- Usuario administrador: **adminlp2**
- Contraseña del administrador: **lab4lp220221**
- Nombre de la base de datos (esquema): **lp2inf282**
- Tipo de acceso: **Público**
- Puerto: **3306**
- **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).

**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 02.

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 05). 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 archivo .zip a PAIDEIA que contenga toda esta nueva estructura. **Está prohibido utilizar el comodín \* para realizar las importaciones en las clases. Se descontarán puntos por importaciones innecesarias.**

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

- **com.avatar.rhnh.model:** IConsultable, Colaborador, Organizacion, Persona.
- **com.gamesoft.project.model:** Proyecto, EstadoProyecto, Producto, Videojuego, Consola, Genero, TipoConectividad.
- **com.gamesoft.main:** Principal.

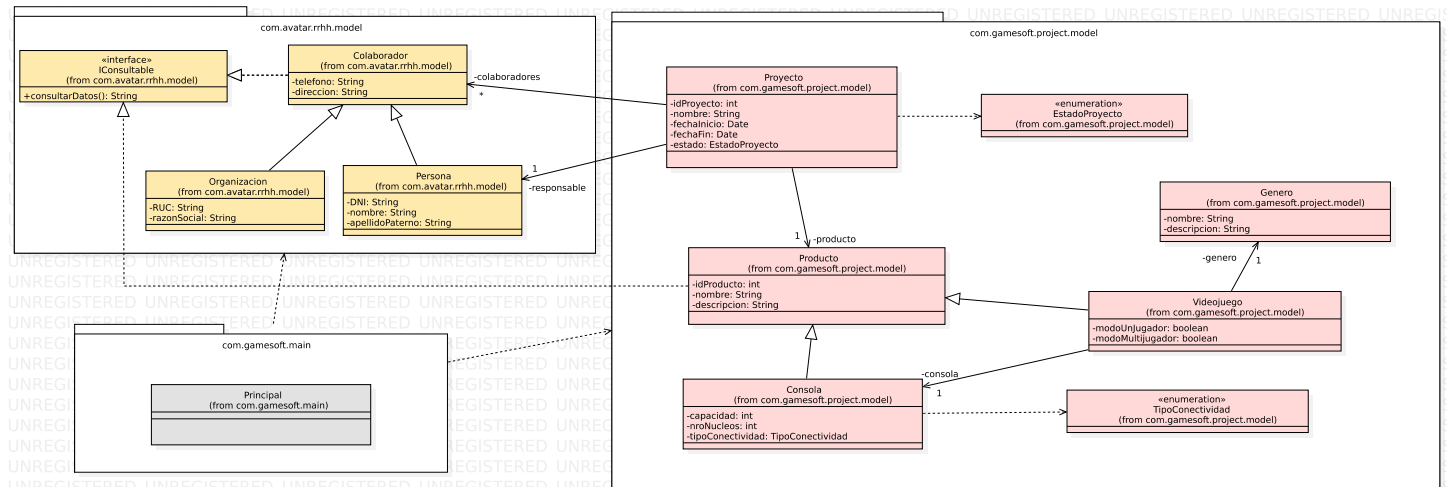


Fig. 01. Diagrama de clases – estructura de paquetes solicitada

### 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.avatar.rhnh.model** desde el directorio raíz de trabajo.
2. Genere un componente .jar llamado “**gsRRHHmodel.jar**” que contenga la estructura y clases en Bytecode contenidas en: **com.avatar.rhnh.model**.

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

3. Compile las clases contenidas en **com.gamesoft.project.model** desde el directorio raíz de trabajo utilizando el .jar previamente creado: “**gsRRHHmodel.jar**”.
4. Genere un componente .jar llamado **gsProjectmodel.jar** que contenga la estructura y clases en Bytecode contenidas en **com.gamesoft.project.model**.

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

5. Compile las clases contenidas en **com.gamesoft.main** desde el directorio raíz de trabajo utilizando los .jar previamente creados: “**gsRRHHmodel.jar**” y “**gsProjectmodel.jar**”.
6. Genere un componente .jar ejecutable llamado “**gsPrincipal.jar**” que contenga la estructura y clases en Bytecode contenidas en **com.gamesoft.main**. 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 “**gsPrincipal.jar**” para que pueda funcionar como componente ejecutable.

Suba el archivo de texto a PAIDEIA. No es necesario adjuntar los archivos .jar.

### PREGUNTA 4: Conexión a Base de Datos (Con uso de IDE) (10 puntos)

Desarrolle una solución de software en lenguaje JAVA o C# en tres capas (*model*, *controller* y *main*) y utilizando procedimientos almacenados que permita realizar la inserción y el listado de objetos de tipo “Servicio”. Un servicio tiene un identificador numérico, un nombre, una descripción, un costo, una relevancia del 1 al 5, una fecha que es aquella en la que se brindará el servicio y un estado que es una letra (‘P’ de pendiente, ‘D’ en desarrollo y ‘F’ de finalizado).

A continuación, se muestran las tablas de base de datos que se han diseñado para este propósito.

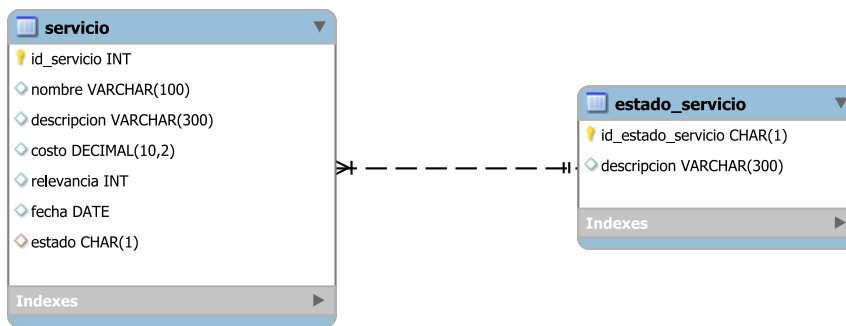


Fig. 02. Diagrama EER

Utilice la base de datos creada en la pregunta 1 y utilice el script que se encuentra en PAIDEIA (**ScriptLab04.sql**) que genera las tablas y sus relaciones. Desarrolle dos procedimientos almacenados que permitan la inserción y el listado de los servicios. Cuando programe el listar servicios debe extraer todos los atributos de un servicio.

La estructura de proyectos debe ser la siguiente:

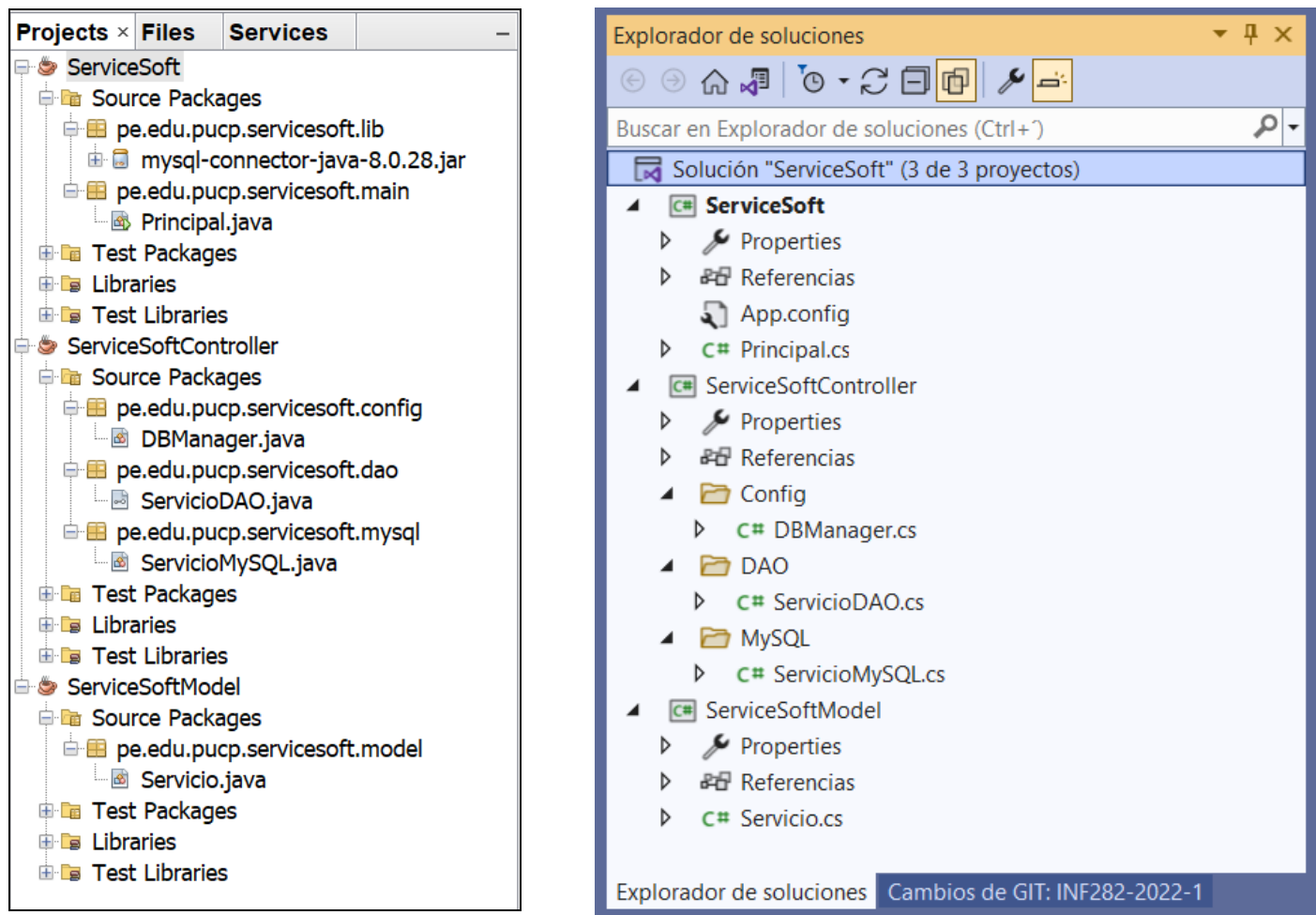
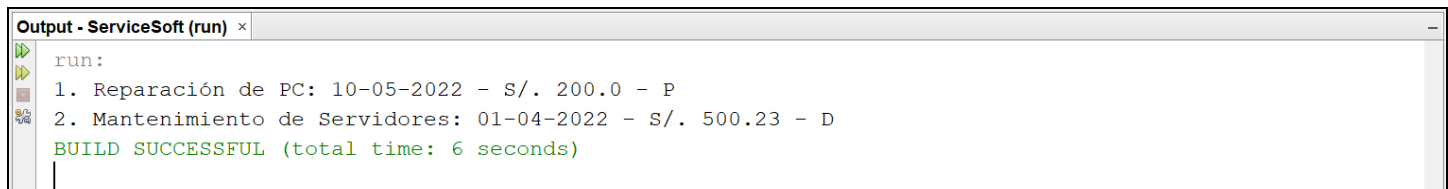


Fig. 03. Estructura de proyectos en JAVA y C#

El método **main()** en JAVA sería el siguiente:

```
public class Principal {
    public static void main(String[] args) throws Exception{
        SimpleDateFormat sdf = new SimpleDateFormat("dd-MM-yyyy");
        Servicio serv1 = new Servicio("Reparación de PC",
            "La oficina de RRHH ha solicitado la reparación de una computadora",
            200.00,5,sdf.parse("10-05-2022"),'P');
        Servicio serv2 = new Servicio("Mantenimiento de Servidores",
            "La facultad de Ciencias e Ingeniería solicitó el mes pasado un mantenimiento de servidores",
            500.23,3,sdf.parse("01-04-2022"),'D');
        ServicioDAO daoServicio = new ServicioMySQL();
        daoServicio.insertar(serv1);
        daoServicio.insertar(serv2);
        ArrayList<Servicio> servicios = daoServicio.listarTodos();
        for(Servicio serv : servicios){
            System.out.println(serv.getIdServicio() + ". " + serv.getNombre() + ": " +
                sdf.format(serv.getFecha()) + " - S/. " + serv.getCosto() + " - " +
                serv.getEstado());
        }
    }
}
```

La ejecución del programa genera la siguiente salida:



```
Output - ServiceSoft (run) x
run:
1. Reparación de PC: 10-05-2022 - S/. 200.0 - P
2. Mantenimiento de Servidores: 01-04-2022 - S/. 500.23 - D
BUILD SUCCESSFUL (total time: 6 seconds)
```

Suba en un archivo .zip todos los proyectos desarrollados a PAIDEIA, así como un archivo de texto donde se detalle el script SQL que permite la creación de los dos procedimientos almacenados.

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 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 tres capas, etc.

**Profesor del Curso:**

**Dr. Freddy Paz**

**05 de mayo del 2022**

En el caso de C# el manejo de fechas es:

```
using System;
DateTime fecha = Convert.ToDateTime("18-09-2012");
System.Console.WriteLine(fecha.ToString("dd-MM-yyyy"));
```