PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

LENGUAJE DE PROGRAMACIÓN 2

2da. práctica (tipo b)
(Primer Semestre 2021)

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.
- Para la sección de Librerías, no está permitido el uso de entornos de desarrollo integrados o IDEs. Deberá realizar la compilación y generación de librerías vía comandos de consola (javac/jar o csc).

A. PARTE TEÓRICA (2 puntos)

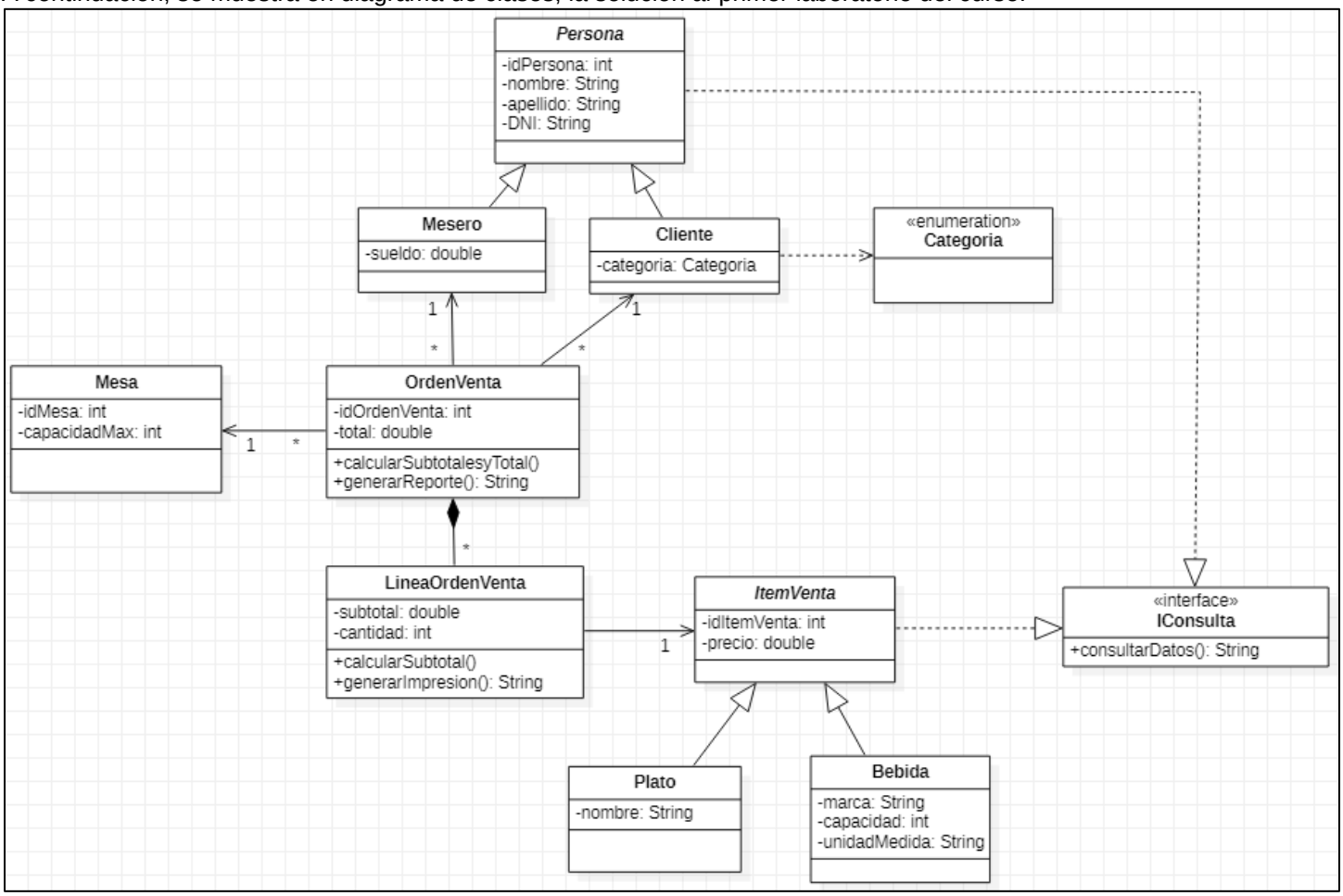
- Responda las siguientes preguntas:
  - El equivalente de la instrucción **`import`** de Java en C# es la palabra reservada **`using`**. Sin embargo, existen diferencias al momento de utilizarlas. ¿Cuáles son estas diferencias? Explique brevemente. (0.5 puntos).
  - Cuando se desean agrupar las clases en JAVA utilizamos los **paquetes** mientras que en C# utilizamos los **espacios de nombres**. ¿Cuál es la principal diferencia entre ambas cuando deseamos estructurar nuestro proyecto? Explique brevemente. (0.5 puntos).
  - ¿Hablar de paquetes es igual que hablar de librerías? Explique brevemente. (0.5 puntos).
  - Cuando compilamos un proyecto en Netbeans, ¿cuántos archivos .jar se generan? Asimismo, ¿es posible disponer de varias estructuras de paquetes en un mismo proyecto? (0.5 puntos).

B. PARTE PRÁCTICA (18 puntos)

PUEDE UTILIZAR MATERIAL DE CONSULTA.

B.1. PREGUNTA 1: SIN USO DE IDE

NOTA: (es posible comprobar si el JAR ha sido generado utilizando el IDE. En este caso, se invalidará su solución).
A continuación, se muestra en diagrama de clases, la solución al primer laboratorio del curso.



Se le solicita descargar los archivos fuente relacionados a la solución del primer laboratorio.

Reorganizar la clase: IConsultable bajo la siguiente estructura de paquetes y generar la siguiente librería: **interfaces.jar Agregue las instrucciones que considere necesarias de public, import y package** a la clase.
Al momento de realizar importaciones no es posible utilizar el *. (Se descontarán puntos por importaciones innecesarias).
Para compilar o generar el jar sí es posible utilizar el *.
Genere un .zip de esta nueva estructura y guarde las instrucciones de compilación y generación del jar.

```
├── interfaces
│   ├── consulta
│   │   ├── IConsulta.java
```

Reorganizar las clases: Persona, Mesero, Cliente y Categoria bajo la siguiente estructura de paquetes y generar la siguiente librería: **organizacion.jar Agregue las instrucciones que considere necesarias de public, import y package** a las clases. Al momento de realizar importaciones no es posible utilizar el *. (Se descontarán puntos por importaciones innecesarias). Para compilar o generar el jar sí es posible utilizar el *.
Genere un .zip de esta nueva estructura y guarde las instrucciones de compilación y generación del jar.

```
├── organizacion
│   ├── rrhh
│   │   ├── Persona.java
│   │   ├── empleados
│   │   │   ├── Mesero.java
│   │   ├── consumidores
│   │   │   ├── Cliente.java
│   │   │   ├── Categoria.java
```

Reorganizar las clases: Mesa, ItemVenta, Plato y Bebida bajo la siguiente estructura de paquetes y generar la siguiente librería: **logistica.jar Agregue las instrucciones que considere necesarias de public, import y package** a las clases. Al momento de realizar importaciones no es posible utilizar el *. (Se descontarán puntos por importaciones innecesarias). Para compilar o generar el jar sí es posible utilizar el *.
Genere un .zip de esta nueva estructura y guarde las instrucciones de compilación y generación del jar.

```
├── logistica
│   ├── infraestructura
│   │   ├── Mesa.java
│   ├── produtos
│   │   ├── ItemVenta.java
│   │   ├── consumibles
│   │   │   ├── Plato.java
│   │   ├── bebibles
│   │   │   ├── Bebida.java
```

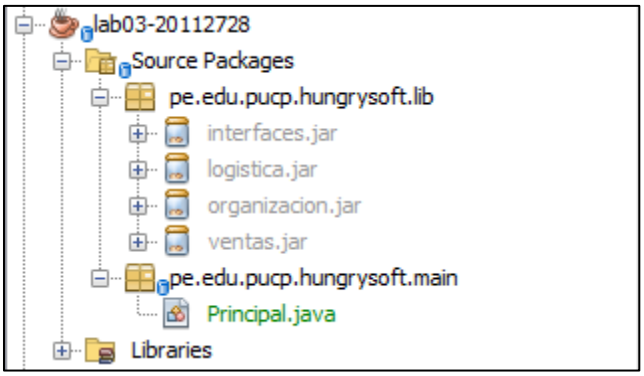
Reorganizar las clases: OrdenVenta y LineaOrdenVenta bajo la siguiente estructura de paquetes y generar la siguiente librería: **ventas.jar Agregue las instrucciones que considere necesarias de public, import y package** a las clases. Al momento de realizar importaciones no es posible utilizar el *. (Se descontarán puntos por importaciones innecesarias). Para compilar o generar el jar sí es posible utilizar el *.
Genere un .zip de esta nueva estructura y guarde las instrucciones de compilación y generación del jar.

```
├── ventas
│   ├── OrdenVenta.java
│   ├── LineaOrdenVenta.java
```

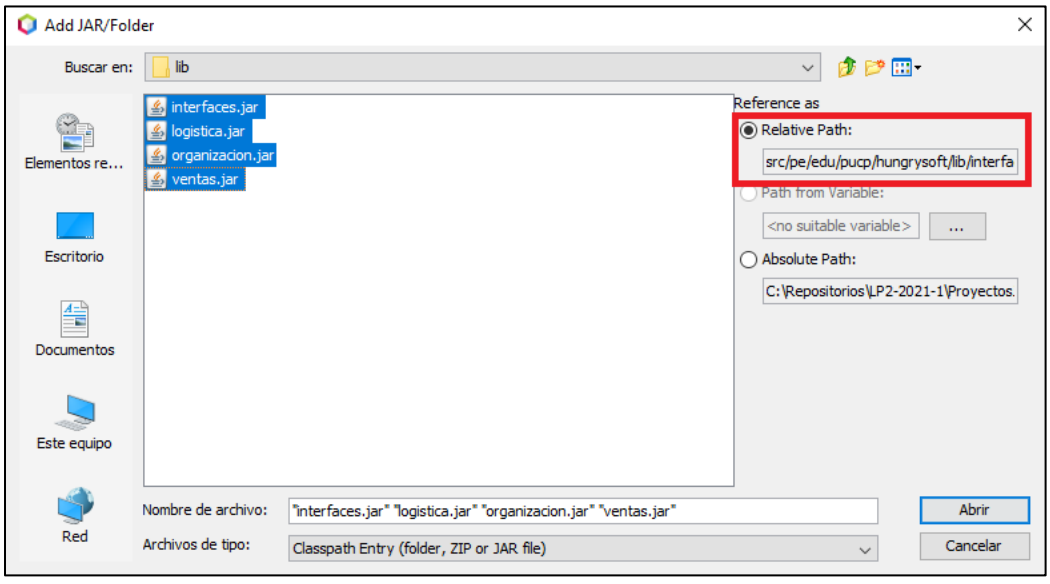
- Suba a PAIDEIA los archivos .zip que contienen las nuevas estructuras.
- Asimismo, suba los archivos jars y las instrucciones empleadas para la compilación de las clases y generación de los mismos.

B.1. PREGUNTA 2: CON USO DE IDE

- Cree un repositorio en blanco para el lenguaje JAVA en la plataforma de GitHub (https://github.com/).
  Utilice su usuario definido en:
  https://drive.google.com/file/d/1iVs_LWXwfQGxLRknZ9tit16Fu1UM3Zuh/view?usp=sharing
- Invite a los siguientes usuarios como parte de su equipo de trabajo en el repositorio que acaba de crear: fpaz19, nespreyes, aroncaln, dvillanuevab y otro34.
- Clone el repositorio remoto en su computadora utilizando Netbeans.
- Utilizando Netbeans, cree un nuevo proyecto en la carpeta asociada al repositorio clonado.
  El nombre de este proyecto será lab3 + su código de alumno.
- En este proyecto, va a crear la siguiente estructura de paquetes: **pe.edu.pucp.hungrysoft.main**
  Dentro de esa estructura va a colocar la clase **Principal.java** de la solución del primer laboratorio.
- Asimismo, va a crear la siguiente estructura de paquetes: **pe.edu.pucp.hungrysoft.lib**
  Dentro de esa estructura va a colocar las librerías **interfaces.jar, logística.jar organizacion.jar y ventas.jar**
- El proyecto quedará de la siguiente manera:

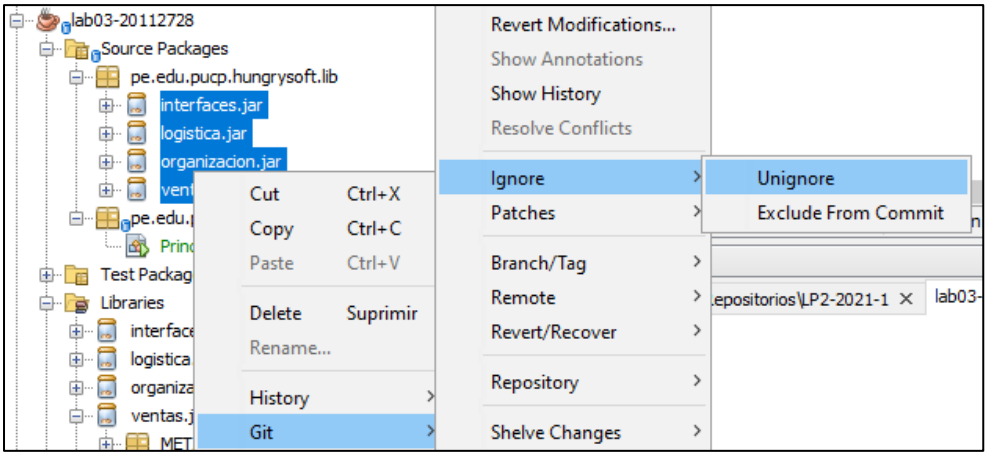


- Luego hacemos clic derecho en Libraries y hacemos clic en Agregar JAR.
- Buscamos los archivos jars dentro de la ruta del proyecto, en: src/pe/edu/pucp/hungrysoft/lib/
  Seleccionamos los cuatro jars y nos aseguramos de que este seleccionada la opción “ruta relativa”.
  Finalmente hacemos click en el botón “Abrir”.



- Luego agregaremos las instrucciones de package e import a la clase Principal a fin de que no aparezcan errores.

- Finalmente haremos click derecho las librerías que se encuentra en **pe.edu.pucp.hungrysoft.lib, ingresaremos a la opción Git -> Ignore -> Unignore**
  (Esto permitirá que los archivos jar seleccionados se envíen al repositorio remoto al momento de realizar push).



- Clic derecho al proyecto y realizar la acción **add** de Git.
- Luego realizar el **commit** para enviar los cambios al repositorio local.
- Luego realizar un **push** para enviar los cambios al repositorio remoto.

- Realice un **clean** del proyecto y suba su proyecto en un archivo .zip de igual manera a PAIDEIA. Asimismo suba en un txt la dirección https de su repositorio de github.

Rúbrica de calificación:

- (2 puntos) Código fuente bajo la nueva estructura e instrucciones que permiten una correcta la generación de la librería **interfaces.jar.**
- (3 puntos) Código fuente bajo la nueva estructura e instrucciones que permiten una correcta la generación de la librería **organizacion.jar.**
- (3 puntos) Código fuente bajo la nueva estructura e instrucciones que permiten una correcta la generación de la librería **logistica.jar.**
- (3 puntos) Código fuente bajo la nueva estructura e instrucciones que permiten una correcta la generación de la librería **ventas.jar.**
- (2 puntos) Creación de repositorio en Github y permisos a los JPs.
- (3 puntos) Creación y programación del proyecto según instrucciones.
- (2 puntos) Envío del proyecto al repositorio remoto.

Profesor del Curso:
Dr. Freddy Paz

22 de abril del 2021