PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

LENGUAJE DE PROGRAMACIÓN 2

1era. 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.
- No está permitido el uso de entornos de desarrollo integrados o IDEs. Para este laboratorio debe utilizar un editor de texto (Notepad++, Sublime, etc.) y realizar la compilación vía comandos de consola (javac o csc).

PARTE PRÁCTICA (20 puntos)

PUEDE UTILIZAR MATERIAL DE CONSULTA.
Se considerará en la calificación el uso de buenas prácticas de programación (aquellas vistas en clase).

PREGUNTA 1 (20 puntos)

La internacional cadena de restaurantes ***Hungry Birds*** desea invertir y colocar una sucursal en el Perú. Para esto, lo ha contratado a Ud. con el objetivo de que realice la programación de un sistema de software, que le permita a la empresa gestionar las ordenes de pedidos en la atención presencial, aquellas que tienen lugar cuando los clientes acuden a las instalaciones a disfrutar de los exquisitos platillos que ofrece esta cadena de restaurantes.

Información sobre el inicio del proceso:

Cuando un cliente llega al restaurante, es recibido por el personal de recepción, quien le ofrece una breve orientación para que el cliente pueda acercarse y tomar asiento en alguna de las mesas que se encuentran disponibles en el local. Posteriormente, un mesero se acerca a la mesa a tomar la orden del cliente. El cliente indica los platos y las bebidas que desea ordenar, mientras el mesero toma nota en un cuadernillo sobre esta información.

Información sobre el funcionamiento del sistema:

Una vez que el cliente ha terminado de indicar toda la información sobre lo que desea ordenar, el mesero se acerca a un módulo dentro del local, donde estará habilitada una computadora de acceso al sistema para que proceda a ingresar la información sobre la orden de pedido. La orden de pedido siempre está asociada a un mesero, que es el encargado de ingresar la información sobre la misma. Asimismo, una orden de pedido está asociada a una mesa, de donde proviene el pedido. Finalmente, cabe mencionar que una orden de pedido también esta asociada a un cliente.

Se espera que el sistema pueda gestionar los datos tanto de los clientes como de los meseros del restaurante.

Un cliente y un mesero comparten características en común como por ejemplo un id, el nombre, el apellido y el DNI. Sin embargo, también tienen datos que los diferencian. Con respecto a los meseros, se espera que el sistema permita la gestión del valor de su sueldo. Asimismo, con respecto a los clientes se espera que el sistema gestione la categoría a la cual pertenecen. La categoría de un cliente puede tomar los siguientes valores: VIP, Platinum y Black.

También se espera que el sistema pueda gestionar la información de las mesas. Una mesa tiene datos como el id y la capacidad máxima de personas que permite.

Con respecto a los posibles ítems de venta de este restaurante, son de dos tipos: los platos y las bebidas. Ambas comparten datos en común como un id y un precio. Sin embargo, en el caso de los platos, estos tienen un nombre. Asimismo, en el caso de las bebidas, estas tienen una marca, una capacidad y una unidad de medida.

Con respecto a la orden de venta, esta tiene un id y un total. Es necesario tener en consideración que, en una misma orden, el cliente puede haber solicitado varios platos y bebidas, con diferentes cantidades, generando múltiples subtotales. El modelo de clases y relaciones debería dar soporte a este aspecto.

Finalmente, la orden de venta debe permitir el cálculo de los subtotales de las líneas de órdenes de venta asociadas a la misma y el total a través del método `void calcularSubtotalesyTotal()`, y apoyándose del método: `void calcularSubtotal()` definido en la clase LineaOrdenVenta. La clase LineaOrdenVenta también define el método `String generarImpresion()` que devuelve la información de la línea (ítem de venta incluido su precio), así como la cantidad de veces que se ha solicitado ese ítem y su subtotal.

Por último, se espera que la orden de venta implemente un método `String generarReporte()` que permita devolver toda la información sobre la misma.

Se ha realizado un análisis previo de la solución y se han determinado las siguientes clases:

Persona: que abstrae los datos de una persona.

ItemVenta: que abstrae los datos de un ítem de venta.
IConsulta: clase de tipo interface que define la obligación de consultar los datos de una persona o ítem de venta.
Categoria: clase de tipo enumerado que define las posibles categorías de un cliente.
LineaOrdenVenta: Representa una línea del detalle de la orden de venta e implementa el método:
`void calcularSubTotal()` y el método `String generarImpresion()`.

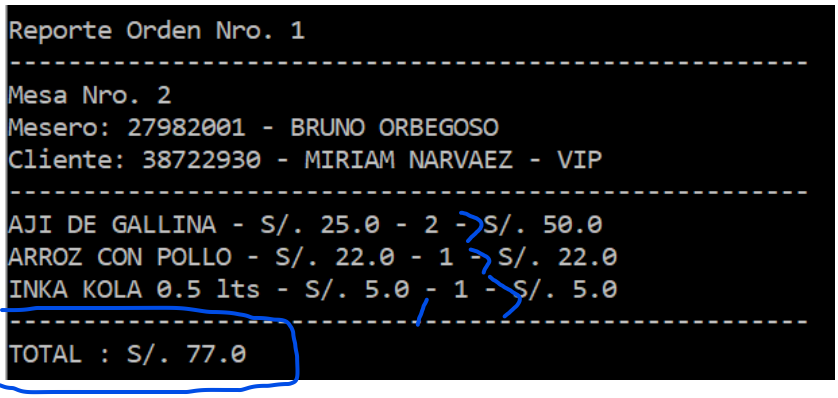
Tanto los clientes como los meseros deben permitir que se puedan consultar sus datos. En el caso de clientes se devuelve el DNI, el nombre, el apellido y la categoría. En el caso de meseros, se devuelve el DNI, el nombre, el apellido.
De igual manera, para el caso de platos que también son consultables, se devuelve el nombre, la cadena "S/." y el precio. Por último, para las bebidas, se devuelve la marca, la capacidad, la unidad de medida, la cadena "S/." y el precio.

Es indispensable que se encuentren programados todos los atributos de las clases y sus relaciones (como mínimo la programación de las relaciones que permiten la impresión del reporte). Con respecto a los constructos, getters y setters, puede programar solo aquellos que son requeridos para la salida del reporte.

Para validar el modelo de clases a realizar y como parte de las pruebas del sistema, se cuenta con la siguiente clase en JAVA:

```
class Principal{
	public static void main(String[] args){
		//Se crean tres mesas
		Mesa mesa1 = new Mesa(1, 2);
		Mesa mesa2 = new Mesa(2, 2);
		Mesa mesa3 = new Mesa(3, 4);
		//Se crea un mesero
		Mesero mesero1 = new Mesero(1, "BRUNO", "ORBEGOSO", "27982001", 1500.00);
		//Se crea un cliente
		Cliente cliente1 = new Cliente(1, "MIRIAM", "NARVAEZ", "38722930", Categoria.VIP);
		//Se crean dos platos
		Plato plato1 = new Plato(1, "AJI DE GALLINA", 25.00);
		Plato plato2 = new Plato(2, "ARROZ CON POLLO", 22.00);
		//Se crea una bebida
		Bebida bebida1 = new Bebida(3, "INKA KOLA", 0.5, "lts", 5.00);
		//Se crea una orden de venta
		OrdenVenta ov1 = new OrdenVenta(1);
		//Se asigna la mesa de la cual proviene la orden de venta
		ov1.setMesa(mesa2);
		//Se asigna el mesero a la orden de venta
		ov1.setMesero(mesero1);
		//Se asigna el cliente a la orden de venta
		ov1.setCliente(cliente1);
		//Se crean las líneas de orden de venta
		LineaOrdenVenta lov1 = new LineaOrdenVenta(plato1, 2);
		LineaOrdenVenta lov2 = new LineaOrdenVenta(plato2, 1);
		LineaOrdenVenta lov3 = new LineaOrdenVenta(bebida1, 1);
		//Se agregan las líneas a la orden de venta
		ov1.agregarLineaOrdenVenta(lov1);
		ov1.agregarLineaOrdenVenta(lov2);
		ov1.agregarLineaOrdenVenta(lov3);
		//Calculamos los subtotales y el total
		ov1.calcularSubtotalesyTotal();
		//Generamos el reporte
		String reporte = ov1.generarReporte();
		System.out.println(reporte);
	}
}
```

La ejecución del programa debería generar la siguiente salida:



Se le solicita programar en JAVA o C# todas las clases que dan soporte lo mencionado en el caso, a la lógica del negocio y a la salida del reporte. Debe subir a PAIDEIA en un archivo .zip, todo el código fuente (archivos .java o archivos .cs).

Aspectos a considerar para evitar descuento de puntos:
- Nombrar correctamente a las clases, atributos y métodos.
- Colocar a modo de comentario su nombre completo y código PUCP en la parte superior de la programación de las clases.
- Respetar el orden en la estructura de una clase.
- Emplear un archivo por clase.
- El programa debe compilar correctamente, se descontarán puntos por errores de compilación.

Rúbrica de calificación:
(1 punto) Correcta programación de la clase interface y la clase enumerate.
(8 puntos) Correcta programación de todas las clases del modelo considerando todos sus atributos.
(2 puntos) Correcta programación de los constructos, getters y setters requeridos para el reporte y el método `main()`.
(2 puntos) Correcta programación del método `consultarDatos()` en las clases donde es requerido con uso de la interface.
(3 puntos) Correcta programación de los métodos `calcularSubtotal()` `generarImpresion()` y `calcularSubtotalesyTotal()`
(2 puntos) Correcta programación del método `generarReporte()`.
(2 puntos) Correcta programación de la clase Principal y visualización del reporte.

Profesor del Curso:
Dr. Freddy Paz

15 de abril del 2021