PRÁCTICA 1_2025.- Restaurante

En esta práctica se van **gestionar mesas de restaurantes.**

Cada restaurante tiene un número de mesas, así como un aforo máximo permitido. **Las mesas están numeradas a partir de 1.**

PASOS PARA LA REALIZACIÓN DE LA PRÁCTICA:

Descargar el proyecto de la página de la asignatura en agora.unileon.es e importarlo en el IDE utilizado:

> Los proyectos normalmente contendrán la estructura a respetar (e incluso pueden tener errores de compilación ya que todo el trabajo está por hacer).

El proyecto tiene un paquete ule.ed.service que contiene las siguientes clases:

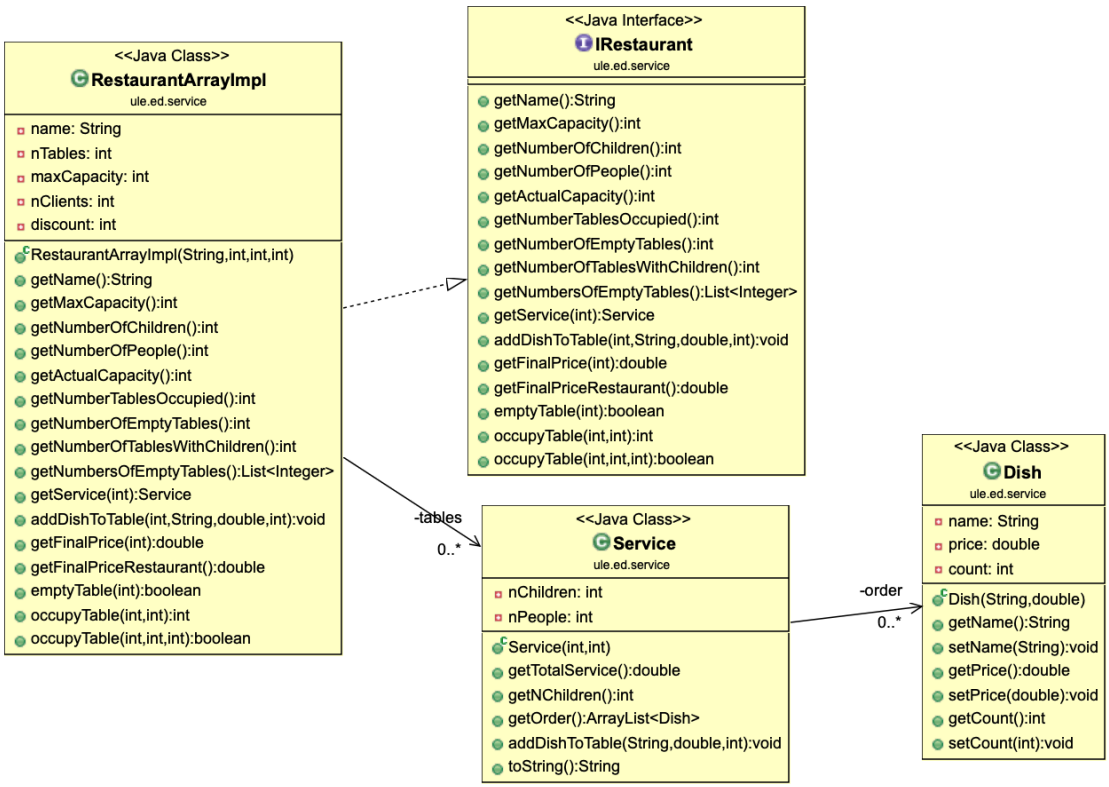
a. Un interface IRestaurante **que NO DEBE MODIFICARSE**.

Las clases que contiene este paquete **DEBEN SER COMPLETADAS (NO PUEDE MODIFICARSE LA ESTRUCTURA DE DATOS, es decir los atributos)**

b. **RestaurantArrayImpl**: Clase que **deberá implementar** un restaurante con arrays. Dispone de un array de servicios, cada posición del array contiene el servicio de una mesa concreta. **Si esa mesa está vacía el elemento correspondiente en el array será null.**

c. **Service:** clase que contiene un servicio para una mesa ocupada (comanda). Almacena el número de clientes sentados a la mesa, cuantos son niños y la lista de platos que han elegido.

d. **Dish**: clase que contiene un plato de un servicio, indica su nombre, precio y cantidad de platos de este tipo que hay en un servicio.



e. **RestaurantArrayImplTests**: Clase que **deberá contener** los tests de prueba necesarios para probar todos los métodos de la clase RestaurantArrayImpl. **SE UTILIZARÁ JUNIT4**.

OPERACIONES DEL INTERFACE IRestaurant:

```
public interface IRestaurant {

	// Devuelve el nombre del restaurante
	public String getName();

	// Devuelve el aforo máximo permitido
	public int getMaxCapacity();

	//Devuelve el número de niños que hay en el restaurante ocupando alguna mesa
	public int getNumberOfChildren();

	//Devuelve el número de personas que hay en el restaurante ocupando alguna mesa
	public int getNumberOfPeople();

	// Devuelve el número de clientes totales que hay en el restaurante
	public int getActualCapacity();

	//Calcula el número de mesas ocupadas
	public int getNumberTablesOccupied();

	//Calcula el número de mesas vacías (no ocupadas)
	public int getNumberOfEmptyTables();

	//Devuelve el número de mesas ocupadas que tienen algún niño
	public int getNumberOfTablesWithChildren();

	// Devuelve una lista de enteros, con el número de las mesas vacías
	public List<Integer> getNumbersOfEmptyTables();

	// Devuelve el servicio que hay en el número de mesa indicado
	// Devuelve null si la mesa no está ocupada
	// Las mesas empiezan en el número 1
	public Service getService(int ntable);

	// añade un plato a un servicio (una comanda) de una mesa
	// Si ya existe un plato en esa comanda con ese mismo nombre y precio,
	// simplemente incrementa el número de platos
	// si no existe, lo añade a la lista de platos de la mesa.
	// count indica el número de dicho plato añadidos
	public void addDishToTable(int nTable, String name, double price, int count);

	// Devuelve el precio final del servicio de la mesa ntable después de aplicarle el descuento del día
	public double getFinalPrice(int ntable);

	// Devuelve la suma del precio final de cada servicio de las mesas ocupadas con el descuento del día
	public double getFinalPriceRestaurant();



	/**
	 * Vacía la mesa con el número indicado.
	 *
	 * Si la mesa con ese número ya está vacía, o si el número no es válido, devuelve false.
	 * Si se pudo vaciar la mesa devuelve true
	 *
	 * Los números de mesa empiezan en '1'.
	 *
	 * @param nTable
	 * @return true: si la mesa estaba ocupada
	 *              o false si la mesa no estaba ocupada o si el número de mesa no era válido.
	 *
	 */
	public boolean emptyTable(int nTable);


	/**
	 * Devuelve el número de mesa a ocupar teniendo en cuenta que :
	 * - el aforo máximo permita añadir el número de personas indicado como parámetro
	 * - buscará la mesa con el número más bajo que no esté ocupada
```

```
         *
         *
         * Las posiciones empiezan en '1'.
         *
         * @param nChildren : número de niños que habrá en la mesa
         * @param nPeople : número de personas incluyendo niños que habrá en la mesa
         * @return el número de mesa que le asigna
         *         -1 si el aforo no permite sentar a ese número de personas
         *         -2 si no se puede encontrar una mesa libre
         */
        public int occupyTable(int nPeople, int nChildren);

        /**
         * Devuelve True si puede ocupar la mesa indicada por el notable :
         * - el aforo máximo permita añadir el número de personas indicado como parámetro
         * - la mesa con el número nTable no esté ocupada
         *
         * o False en caso contrario
         *
        * @param nTable : número de la mesa a ocupar
         * @param nChildren : número de niños que habrá en la mesa
         * @param nPeople : número de personas incluyendo niños que habrá en la mesa
         * @return true o false, dependiendo de si puede ocupar la mesa o no
         *
         */

        public boolean occupyTable(int nTable, int nPeople, int nChildren);



}
```

ATRIBUTOS DE LA CLASE RestaurantArrayImpl:

```
private String name;
private int nTables;
private int maxCapacity; // máximo número de clientes totales admitidos
private int nClients; // contador de clientes actuales en el restaurante

private int discount; // Descuento a aplicar (ejemplo: 10%)
private Service[] tables; // array de servicios (cada servicio se corresponde con una
mesa)
```

ATRIBUTOS DE LA CLASE Service:

```
private ArrayList<Dish> order; //Lista de platos que se han pedido en este servicio pri-
vate int nChildren;
private int nPeople;
```

ATRIBUTOS DE LA CLASE Dish:

```
private String name; private double price; private int count;
```

CONSIDERACIONES PARA LA IMPLEMENTACIÓN DE LA PRÁCTICA:

- El constructor de la clase RestaurantArrayImpl **debe crear el array de Servicios**, pero **no debe crear los Servicios de cada posición del array**. Estos objetos Servicio se crearán cuando se ocupe una mesa con el método `int occupyTable(int nPeople, int nChildren) y`

- Es importante recordar que los arrays empiezan en 0 pero la numeración de las mesas empiezan en 1, por lo que el servicio correspondiente a la mesa n estará almacenado en el array en la posición n-1.

- Cualquier método que reciba como parámetro el no de mesa debe comprobar que es un valor válido para que **no provoque errores** en el programa debido a salirse de los límites del array.

- **IMPORTANTE**: Utilizar JUNIT 4.

FECHA LIMITE de entrega de la práctica 1: 9 de Marzo de 2025 - 23:59