Grupo Weboo
Universidad de La Habana

Examen Extraordinario de Programación
Curso 2014-2015

Ordenando paso a paso

NOTA: Si usted está leyendo este documento sin haber extraído el compactado que se le entregó, ciérrelo ahora, extraiga todos los archivos en el escritorio, y siga trabajando desde ahí. Es un error común trabajar en la solución dentro del compactado, lo cual provoca que los cambios **no se guarden**. Si usted comete este error y entrega una solución vacía, no tendrá oportunidad de reclamar.

Se desea desarrollar una aplicación que permita confeccionar el menú de un restaurante, así como registrar las órdenes que se realizan.

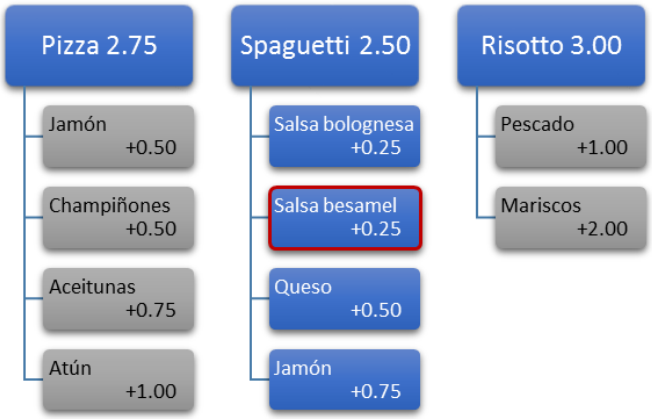
El menú de este restaurante está compuesto por diversos tipos de platos y un conjunto de ingredientes que pueden ser adicionados según el gusto del cliente. Cada aditivo es propio del tipo de plato.

Ejemplo de menú

Para registrar una orden se debe introducir en la aplicación, paso a paso, la elección realizada por el cliente. Si se desea ordenar **Spaguetti** con **Salsa besamel** y **Queso**, se debe introducir la secuencia de opciones siguiente:

Paso 1: Seleccionar Spaguetti

Paso 2: Seleccionar Salsa besamel



Paso 3: Seleccionar Queso

Como se muestra en las imágenes anteriores las opciones habilitadas en cada paso (que pueden ser seleccionadas) poseen color azul. Es decir, luego de seleccionar un tipo de plato, se habilitan sus aditivos disponibles y todos los tipos de platos. Si se selecciona un aditivo entonces se agregará al plato en preparación.

Si se selecciona un nuevo plato significa que el plato anterior está listo y se procede a preparar otro adicional del tipo especificado.

Implementación

Usted debe implementar la siguiente `interface` para simular el funcionamiento de la aplicación presentada:

```
public interface IRestaurante
{
    /// <summary>
    /// Añade un nuevo tipo de plato al menú del restaurante.
    /// </summary>
    void CreaNuevoPlato(string tipoPlato, double precioBase);

    /// <summary>
    /// Añade un aditivo disponible para el tipo de plato especificado.
    /// </summary>
    void CreaNuevoAditivo(string tipoPlato, string aditivo, double precio);

    /// <summary>
    /// Devuelve todos los tipos de platos en el restaurante con sus aditivos
disponibles.
    /// </summary>
    IEnumerable<TipoPlato> Menu { get; }

    /// <summary>
    /// Selecciona un tipo de plato o aditivo, en dependencia de la opción elegida
previamente.
    /// </summary>
    void Selecciona(string opcion);

    /// <summary>
    /// Inicia el procesamiento de una orden.
    /// </summary>
    void IniciaOrden();


    /// <summary>
    /// Finaliza el procesamiento de una orden.
    /// </summary>
    void FinalizaOrden();

    /// <summary>
    /// Reporta los costos de las órdenes atendidas en el mismo orden en que se
realizaron.
    /// </summary>
    IEnumerable<double> OrdenesAtendidas { get; }
}
```

Los métodos `CreaNuevoPlato` e `CreaNuevoAditivo`, son los métodos para confeccionar el menú del restaurante. Si se intenta crear dos tipos de platos con igual nombre, se deberá lanzar una excepción del tipo `ArgumentException`. La misma excepción debe ser lanzada si se intenta crear un aditivo de un tipo de plato inexistente o dos aditivos de un plato con igual nombre. Se le **asegura** que nunca un tipo de plato va a tener el mismo nombre que algún aditivo.

La propiedad `Menu` enumera los tipos de platos con sus respectivos aditivos disponibles.

La definición de la clase `TipoPlato` es:

```
public class TipoPlato
{
    public Plato(string nombre, double precioBase, IEnumerable<Aditivo> aditivos) { … }

    /// <summary>
    /// Devuelve el nombre del tipo de plato.
    /// </summary>
    public string Nombre { get; private set; }

    /// <summary>
    /// Devuelve el precio base del tipo de plato.
    /// </summary>
    public double PrecioBase { get; private set; }

    /// <summary>
    /// Devuelve los aditivos disponibles para este tipo de plato.
    /// </summary>
    public IEnumerable<Aditivo> AditivosDisponibles { get; private set; }

}
```

La definición de la clase `Aditivo` es:

```
public class Aditivo
{
    public Aditivo(string nombre, double precio) { … }

    /// <summary>
    /// Devuelve el nombre del aditivo.
    /// </summary>
    public string Nombre { get; private set; }

    /// <summary>
    /// Devuelve el precio del aditivo.
    /// </summary>
    public double Precio { get; private set; }
}
```

Si se invoca alguno de los métodos `Selecciona` o `FinalizaOrden` sin estar en procesamiento una orden se deberá lanzar una excepción del tipo `InvalidOperationException`. Lo mismo debe ocurrir si es invocado `IniciaOrden` cuando aún está en procesamiento una orden.

Si al método `Selecciona` se le pasa como argumento una opción inválida (que no está habilitada o no existe) se deberá lanzar una excepción del tipo `InvalidOperationException`.

El método `OrdenesAtendidas` es utilizado para obtener un reporte de los costos de las órdenes atendidas. Este método **no puede devolver la orden que esté en procesamiento**.

En la solución entregada usted debe completar la definición de la clase `Restaurante` que implementa `IRestaurante`. Esta biblioteca de clases tiene referenciado el ensamblado `Weboo.Utils` que contiene los tipos definidos previamente.

Ejemplo

El menú mostrado previamente se puede representar de la siguiente forma:

```
Restaurante r = new Restaurante();
r.CreaNuevoPlato("Pizza", 2.75);
r.CreaNuevoAditivo("Pizza", "Jamón", 0.50);
r.CreaNuevoAditivo("Pizza", "Champiñones", 0.50);
r.CreaNuevoAditivo("Pizza", "Aceitunas", 0.75);
r.CreaNuevoAditivo("Pizza", "Atún", 1.00);

r.CreaNuevoPlato("Spaguetti", 2.50);
r.CreaNuevoAditivo("Spaguetti", "Salsa bolognesa", 0.25);
r.CreaNuevoAditivo("Spaguetti", "Salsa besamel", 0.25);
r.CreaNuevoAditivo("Spaguetti", "Queso", 0.50);
r.CreaNuevoAditivo("Spaguetti", "Jamón", 0.75);

r.CreaNuevoPlato("Risotto", 3.00);
r.CreaNuevoAditivo("Risotto", "Pescado", 1.00);
r.CreaNuevoAditivo("Risotto", "Mariscos", 2.00);
```

Mostrando el menú:

```
foreach (var tipoPlato in r.Menu)
{
    Console.WriteLine("{0, -17} .....{1}", tipoPlato.Nombre, tipoPlato.PrecioBase);
    foreach (var aditivo in tipoPlato.AditivosDisponibles)
        Console.WriteLine("  {0, -15} .....{1}+", aditivo.Nombre, aditivo.Precio);
}
```

Si un cliente pide:

- **Rissoto** con **Mariscos**

y otro pide:

- **Pizza** sin aditivos
- **Spaguetti** con **Salsa boloñesa** y **Jamón**

Se debe realizar lo siguiente:

```
r.IniciaOrden();
r.Selecciona("Risotto");        // En preparación: "Risotto"
r.Selecciona("Mariscos");       // En preparación: "Risotto con Mariscos"
r.FinalizaOrden();              // Terminado: "Risotto con Mariscos" (5.00)

r.IniciaOrden();
r.Selecciona("Pizza");          // En preparación: "Pizza"
r.Selecciona("Spaguetti");      // Terminado: "Pizza" (2.75), En preparación: "Spaguetti"
r.Selecciona("Salsa bolognesa"); // En preparación: "Spaguetti con Salsa bolognesa"
r.Selecciona("Jamón");         // En preparación: "Spaguetti con Salsa bolognesa y Jamón"
r.FinalizaOrden();           // Terminado: "Spaguetti con Salsa bolognesa y Jamón" (3.50)
```

Para verificar los clientes que han pasado terminado su pedido:

```
foreach (var x in r.OrdenesAtendidas)
    Console.WriteLine(x);
//5
//6.25
```