Una pequeña empresa de restauración solicita nuestros servicios para desarrollar una aplicación que nos permita gestionar a sus empleados.

De todos los empleados se desea guardar la siguiente información: *código de empleado*, *nombre*, *años de experiencia previa en el sector* (antes de su contratación) *y año de ingreso en la empresa*.

En la empresa podemos tener los siguientes tipos de empleados: **cocinero**, **encargado** y **camarero**. Los camareros, a su vez, pueden ser **fijos** o **eventuales**. Los camareros eventuales serán requeridos en aquellos eventos en los que se necesite mayor afluencia de personal, de ellos almacenaremos además un *número de teléfono* para localizarlos en dichas ocasiones. Los camareros eventuales deberán implementar una interfaz llamada **ILlamable** que define un único método **llamar**. Dicho método mostrará un mensaje pantalla con el texto "Llamando al número [*número de teléfono del empleado*]…",

Todos los empleados implementarán el método **toString** que nos devolverá la información del empleado indicando su cargo (Encargado, Cocinero o Camarero), código de empleado, nombre, años de experiencia previa y año de ingreso en la empresa. Los camareros eventuales mostrarán también el teléfono. El formato será como en el siguiente ejemplo:

CAMARERO
Código: 0302
Nombre: Fulanito Gómez
Experiencia previa: 10
Año de ingreso: 2016
Teléfono: 678789144

Los códigos de empleado estarán formados por 4 dígitos y serán asignados automáticamente por el sistema de forma que los códigos de los encargados empiecen siempre por 01, los de los cocineros por 02 y los de los camareros por 03, no pudiendo haber códigos repetidos. Para ello, la clase Empleado definirá un método abstracto llamado **generarCodigo** que será implementado por las clases correspondientes y nos asignará un código para el empleado. EL método generarCodigo() será llamado desde el constructor de cada tipo de empleado.

Nuestro programa tendrá además una clase llamada **Gestion** que tendrá una constante llamada **CUPO** que indicará el número máximo de empleados que podemos contratar y a la que daremos un valor de 20 y como atributo, un array en el que podremos almacenar tantos empleados como indique la constante CUPO como máximo, aunque podríamos tener menos empleados. Las posiciones del array en la que no tengamos empleados almacenados pueden estar vacías.

La clase Gestion tendrá al menos los siguientes métodos:

contratar(): preguntará por el tipo de empleado a contratar, pedirá el nombre, la experiencia previa, el año de ingreso y, si es necesario, el teléfono y añadirá el empleado al array. En el caso de que ya se tenga cubierto el número máximo de empleados indicado en la constante CUPO nos mostrará un mensaje indicando que el cupo de empleados está completo y no creará ningún empleado nuevo.

despedir(): pedirá el código del empleado que se desea despedir y, de existir, lo eliminará del array. Si no existe, sacará un mensaje indicándolo.

listar(): mostrará un listado de todos los empleados contratados

llamar(): pedirá el código del empleado a llamar. Si el empleado existe y es un camarero eventual, llamará al método llamar del mismo, si existe pero no es un camarero eventual, mostrará un mensaje diciendo que no es un camarero eventual, si no existe, mostrará un mensaje indicándolo.

La clase Gestion incluirá además un método main que mostrará el siguiente menú:

1. Contratar
2. Despedir
3. Listar
4. Llamar
5. Salir

Dicho menú llamará a los correspondientes métodos de la clase y se volverá a mostrar hasta que el usuario seleccione la opción 6. Salir

Se pide:

- Diseñar el **diagrama de clases** para este sistema, agrupando elementos (atributos y métodos) comunes evitando duplicar elementos. Para ello se podrá añadir clases intermedias o abstractas si se considera oportuno. Se deberá entregar la representación de dicho diagrama por escrito.
- **Implementar las clases descritas** ajustándose al esquema diseñado en el apartado anterior. Se implementarán los métodos que se consideren necesarios y se valorará el correcto uso de los modificadores.
- **Parte extra**: adicionalmente, el alumno podrá implementar otro método para la clase Gestion llamado estadísticas() cuya función será la siguiente:
    - **estadísticas():** Sacará por pantalla información sobre la plantilla indicando el número de empleados contratados y el porcentaje de empleados que hay de cada tipo.

  A esto añadirá una nueva opción en el menú que será la opción 5. Estadísticas, pasando la opción Salir a ser la número 6. La realización de esta parte extra podrá sumar hasta 1 punto a la nota total del examen.

NOMBRE: _____

DIAGRAMA DE CLASES

Criterios de calificación			
Apartado	**Criterio**	**Max**	**Puntos**
Compilación	**Si el programa no compila la nota máxima del examen será de 2 puntos**		
Diseño del diagrama de clases	Elección adecuada de las clases añadiendo aquellas necesarias para evitar duplicidad	1,5	
Implementación del diagrama de clases	Correspondencia del diagrama con su implementación	0,5	
	Implementación de los métodos necesarios en cada una de las clases	0,5	
	Correcto uso de los modificadores	0,5	
	Uso correcto de clases abstractas e interfaces	0,5	
	Método generarCódigo()	1	
	Método toString()	0,5	
Programa principal: Gestor	Definición correcta de la clase según el enunciado	0,5	
	Método contratar()	1,5	
	Método despedir()	1	
	Método listar()	0,5	
	Método llamar()	1	
	Funcionamiento del menú	0,5	
Extra	Método estadísticas()	1	
Total		10+1	