Examen 2ª evaluación

La empresa *GestoresEmpresas* te ha contratado para desarrollar un software de gestión de empresas. Se trata de una aplicación Java formada por una clase principal **Gestion**, y otras llamadas: **Empresa, Persona**, **Director**, **GerenteDep**, **Trabajador, Enumerado (Informática, Gestión, Marketing), Excepciones necesarias**.

Clase Gestion

Clase principal con función main. Encargada de interactuar con el usuario, mostrar el menú principal, dar feedback y/o mensajes de error, etc. Puedes implementar las funciones que consideres oportunas en cualquiera de las tres clases.

El programa **pedirá los datos necesarios para crear una empresa**. Si son válidos, creará la empresa y mostrará el menú principal para permitir actuar sobre la empresa. Tras cada acción se volverá a mostrar el menú.

1. **Registrar trabajador en empresa:**
    a. **Registrar director (solo puede tener uno)**
    b. **Registrar GerenteDep (máximo 3)**
    c. **Registrar Trabajador normal**
2. **Mostrar información general de la empresa, con todos los trabajadores, gerentes y director.**
3. **Mostrar el número de trabajadores actuales y el organigrama de la empresa.**
4. **Mostrar información de un departamento**
5. **Eliminar trabajador de la empresa**
6. **Agenda Director:**
    a. **Reunión**
    b. **Fuera de la oficina**
    c. **Convocar a toda la empresa(fecha) (si director está disponible) (solo los trabajadores que estén en la oficina)**

Salir de la aplicación

Clase Empresa

Una empresa tiene como datos asociados:

- Nombre de la empresa
- CIF: (Letra mayúscula y 10 dígitos)
- Fecha de fundación (LocalDate)
- Colección de para gestionar los trabajadores, gerentes y director, de forma que no pueda ver dos personas con el mismo DNI.

Cuando se crea una empresa es **obligatorio que tenga un CIF en el formato correcto (en caso contrario lanzar Excepción personalizada), un nombre, una fecha de fundación**. Se deben de crear los **constructores y propiedades necesarias** con la visibilidad adecuada.

Se deben de crear al menos los siguientes métodos:

- **registrarTrabajador:** comprobación de que no este ya registrado por el DNI y asignar un numero SS al trabajador (propiedad de clase Trabajador).
- **registrarGerente:** comprobación de que no esté ya registrado por el DNI y asignar un numero SS al trabajador (propiedad de clase Trabajador).
- **registrarDirector:** comprobación de que no esté ya registrado por el DNI y asignar un numero SS al trabajador (propiedad de clase Trabajador).
- **eliminarTrabajador**: debe eliminar el trabajador de la empresa completamente. Para buscar al Trabajador se debe de hacer por DNI.
- **Sobrescribir método toString()**: mostrar toda la información general de la empresa.
- **mostrarInformacionTrabajadores**: mostrar toda la información de todos los trabajadores registrados de la empresa.
- **reunirEmpresa()**: mostrar toda la información de los trabajadores que estén en la oficina, si el director está disponible.

Se pueden crear todos los métodos y excepciones que se necesiten o se consideren necesarias.

Clase Persona

La clase Persona deberá tener los siguientes atributos:

1. Nombre
2. FechaNacimiento
3. DNI (debe de tener el formato correcto 8 dígitos y una letra mayúscula)

4. Dirección

Se debe de comprobar que el formato del DNI sea correcto, en caso contrario lanzar una **excepción personalizada**. Además de los constructores necesarios. Las **propiedades deben de tener la visibilidad adecuada**. Se debe de **implementar el método toString()** para mostrar la información de una persona.

Clase Trabajador

La clase Trabajador deberá heredar de la clase Persona:

1. NumeroSS (10 dígitos numéricos)
2. Email de la empresa (comprobación opcional)
3. Salario
4. Departamento: Enumerado (informática, gestión, marketing)
5. Booleano estar en la oficina

Se debe de comprobar que el formato del NumeroSS sea correcto, en caso contrario lanzar una **excepción personalizada.** Además de los constructores y métodos necesarios se pueden crear tantos como se desee. Las **propiedades deben de tener la visibilidad adecuada**. Se debe de sobrescribir el método toString() del padre para mostrar la información de cada trabajador.

Clase GerenteDep

La clase GerenteDep deberá heredar de la clase Trabajador:

1. Numero trabajadores en el departamento
2. Número de teléfono (comprobación opcional)
3. Colección de trabajadores del departamento

Las **propiedades deben de tener la visibilidad adecuada**. Además, tendrá los siguientes métodos.

convocarReunionDepartamento(LocalDateTime fReunion): se mostrará por pantalla la fecha y todos los trabajadores convocados.

mostrarInfoDepartamento(): mostrar la información de todos los trabajadores y el coste total salarial.

Clase Director

La clase Director deberá heredar de la clase Trabajador:

1. Número de teléfono (comprobación opcional)
2. Booleano de estar reunido
3. Booleano de fuera de la oficina

Las **propiedades deben de tener la visibilidad adecuada**. Además, tendrá los constructores y métodos que se crean necesarios.

Se admiten y se evaluarán mejoras sustanciales en el programa o en la complejidad. Así como comprobaciones o validaciones.