Ejercicio: Gestión de Empleados y Revisiones Salariales

Vas a desarrollar una pequeña aplicación Java que gestione empleados de una empresa usando **Hibernate** y una arquitectura en capas:

- **Capa de acceso a datos (DAO)**
- **Capa de servicios (lógica de negocio + validaciones)**
- **DTOs para comunicación con la capa de presentación**
- **Capa de presentación simulada con una clase main**

1. Tabla y entidad Hibernate: TB_EMPLEADO

Se dispone de una tabla en la base de datos llamada TB_EMPLEADO con los siguientes campos:

- ID (numérico, PK, autogenerado)
- NIF (cadena, no nulo, único)
- NOMBRE (cadena, no nulo)
- DEPARTAMENTO (cadena, no nulo; por ejemplo: "IT", "RRHH", "VENTAS")
- SALARIO (numérico con decimales, no nulo, ≥ 0)

Debes crear la entidad JPA/Hibernate correspondiente (Empleado) mapeada a esta tabla.

2. Funcionalidad a implementar (vista desde la capa de servicios)

La aplicación debe permitir las siguientes operaciones de negocio sobre empleados:

1. **Alta de empleado**
    - Recibe los datos de un empleado (NIF, nombre, departamento, salario).
    - Valida las reglas de negocio (ver apartado 3).
    - Si todo es correcto, inserta el empleado en la base de datos.
    - Devuelve la información del empleado creado.

2. **Actualización de salario**
    - Permite actualizar el salario de un empleado existente.
    - Se identificará al empleado por su NIF o por su ID (a definir en el diseño).
    - Aplica las validaciones de negocio (p. ej. salario no negativo, cambios mínimos, etc.).
3. **Búsqueda de empleado**
    - Permite buscar un empleado por NIF.
    - Si no existe, se devolverá un resultado adecuado (por ejemplo, null o lanzar una excepción de negocio).
4. **Listado de empleados por departamento**
    - Devuelve un listado de empleados pertenecientes a un determinado departamento.
    - Se usará para mostrar en la "capa de presentación" un resumen de empleados por área.

3. Reglas de negocio (validaciones en la capa de servicios)

La **capa de servicios** debe aplicar, como mínimo, estas validaciones:

1. **NIF obligatorio y único**
    - No se puede dar de alta ni actualizar un empleado con NIF nulo o vacío.
    - No se puede dar de alta un empleado si ya existe otro empleado con el mismo NIF.
2. **Nombre obligatorio**
    - El nombre no puede ser nulo ni vacío.
3. **Departamento válido**
    - El departamento no puede ser nulo ni vacío.
    - Opcional: limitar a una lista de valores válidos (ejemplo: "IT", "RRHH", "VENTAS", "FINANZAS").
      Si el departamento no está en la lista, se debe considerar error de negocio.

4. **Salario válido**
    - El salario no puede ser nulo.
    - El salario debe ser **mayor o igual que 0**.
    - Opcional: definir un salario mínimo razonable (por ejemplo, ≥ 1000) y validar también esto.

En caso de incumplir alguna de estas reglas, la capa de servicios **debe lanzar una excepción de negocio**.

4. Elementos que hay que crear (clases, interfaces, excepciones...)

4.1. Entidad Hibernate

- **Clase**: Empleado
    - Mapeada a la tabla TB_EMPLEADO.
    - Incluye los campos: id, nif, nombre, departamento, salario.
    - Con anotaciones JPA/Hibernate (@Entity, @Table, @Id, etc.).

4.2. DTO (Data Transfer Object)

- **Clase**: EmpleadoDto
  Será el objeto que se usará para comunicar la capa de presentación con la capa de servicios.

Campos recomendados:

- Long id (opcional en la presentación, puedes decidir si exponerlo o no)
- String nif
- String nombre
- String departamento
- BigDecimal salario

4.3. DAO (Data Access Object)

- **Interfaz**: EmpleadoDao
  Debe declarar, al menos, los siguientes métodos:
    - void guardar(Empleado empleado); (crear o actualizar)
    - Empleado buscarPorId(Long id);

    - Empleado buscarPorNif(String nif);
    - List<Empleado> buscarPorDepartamento(String departamento);
    - List<Empleado> buscarTodos();
- **Implementación**: EmpleadoDaoImpl
    - Implementa la interfaz EmpleadoDao usando Hibernate.
    - Gestiona las sesiones y transacciones (Session, Transaction, etc.).

4.4. Capa de servicios

- **Interfaz**: EmpleadoService
  Debe trabajar con **DTOs**, no con entidades. Métodos sugeridos:
    - EmpleadoDto altaEmpleado(EmpleadoDto empleadoDto);
    - EmpleadoDto actualizarSalario(String nif, BigDecimal nuevoSalario);
    - EmpleadoDto buscarPorNif(String nif);
    - List<EmpleadoDto> listarPorDepartamento(String departamento);
    - List<EmpleadoDto> listarTodos(); (opcional)
- **Implementación**: EmpleadoServiceImpl
    - Contiene la lógica de negocio y las validaciones descritas en el punto 3.
    - Usa el EmpleadoDao para acceder a la base de datos.
    - Convierte entre Empleado (entity) y EmpleadoDto (DTO).
    - Lanza excepciones de negocio cuando las validaciones fallen.

4.5. Excepciones

- **Clase**: BusinessException (o similar)
    - Excepción de **negocio**, lanzada desde la capa de servicios cuando se incumplen reglas como:
        - NIF duplicado
        - NIF/Nombre/Departamento inválidos
        - Salario incorrecto

Opcionalmente, puedes definir más excepciones específicas si lo ves útil.

5. Capa de presentación simulada (main)

- **Clase**: AppEmpleados (o similar)
    - Contendrá un método public static void main(String[] args).
    - Simulará la capa de presentación (por ejemplo, una aplicación de consola).
    - Debe:
        - Crear algunos EmpleadoDto y llamar a altaEmpleado.
        - BuscarPorNif y listarPorDepartamento.
        - Capturar las BusinessException y mostrar el mensaje por consola.
        - Probar actualizarSalario,