### Escuela Colombiana de Ingeniería

### Procesos de desarrollo de Software – PDSW

### Parcial Práctico - 1er Tercio

# Parte I. - Validar empleados

Para la primera parte del parcial, se creará una clase encargada de realizar las validaciones correspondientes para verificar si un empleado que será ingresado al sistema, cumple algunas condiciones básicas que tiene la compañía y unas regulaciones del país.

1. Arbol del proyecto del parcial

```
.
+-- pom.xml
+-- README.md
+-- README.pdf
+-- Wiki
    +-- QuickTheories.md
+-- src
    +-- main
    |  +-- java
    |  |  +-- edu
    |  |      +-- eci
    |  |          +-- pdsw
    |  |              +-- model
    |  |              |  +-- Employee.java
    |  |              |  +-- SocialSecurityType.java
    |  |              +-- servlet
    |  |              |  +-- ValidateServlet.java
    |  |              +-- validator
    |  |                  +-- EmployeeValidator.java
    |  |                  +-- ErrorType.java
    |  |                  +-- SalaryValidator.java
    |  +-- resources
    |      +-- form.html
    |      +-- result.html
    +-- test
        +-- java
            +-- edu
                +-- eci
                    +-- pdsw
                        +-- validator
```

```
+-- SalaryValidatorTest.java
```

2. El proyecto cuenta con una estructura inicial, la cual se describirá a continuación:

   - Paquete `edu.eci.pdsw.model`: Tiene las clases básicas del modelo.
     - Enumeración `SocialSecurityType`: Enumeración con 3 tipos básicos de seguridad social (Sisben, EPS, Prepagada).
     - Clase `Employee`: Clase principal del modelo, representación de un empleado en el sistema.
   - Paquete `edu.eci.pdsw.validator`: Se encuentra la lógica de negocio.
     - Enumeración `ErrorType`: Enumeración con 2 tipos de errores básicos de un empleado (error en afiliación al Sisben o a la EPS).
     - Interfaz `EmployeeValidator`: Interfaz principal para cargar la lógica de negocio para realizar las validaciones requeridas.
     - Clase `SalaryValidator`: Clase principal de la lógica de negocio, donde se realiza la validación salarial de un empleado.
   - Paquete `edu.eci.pdsw.validator` (en src/test/java): Se encuentran las pruebas de la lógica de negocio.
     - Clase `SalaryValidatorTest`: en esta clase deben estar las pruebas para `SalaryValidator`.

3. Para la validación de un registro ante el sistema de seguridad social, se tienen las siguientes condiciones:

   - Identificador:
     - Debe ser un numérico entre 1.000 y 100.000
   - Nombre:
     - Debe contener al menos 1 nombre y 1 apellido.
     - La longitud debe ser entre 10 y 30 caracteres.
   - Salario:
     - Debe ser un número positivo, entre XXXX y YYYY.
   - Seguridad Social:
     - El salario básico para Sisben es XXX
     - El salario básico para EPS es XXX
     - El salario básico para Prepagada es XXX

4. Implemente con la librería QuickTheories, un generador de empleados que me permita obtener una gran cantidad de escenarios distintos para validar empleados.

5. En caso de ser necesario, agregue más tipos de error a la enumeración, con diferentes clases de equivalencia para el problema.

6. Implemente la lógica específica en la clase `SalaryValidator` para cumplir con las restricciones de seguridad social, del punto anterior.

7. Verifique la correcta compilación, ejecución y pruebas del proyecto, por medio de la construcción con Maven.

# Parte II. - Realizar validación Web

En esta parte del parcial, se va a exponer por medio de unos servicios web, la aplicación implementada hasta el momento, de tal forma que sea posible realizar la validación de un empleado por medio de un formulario y peticiones http al servidor.

1. Se creó la siguiente estructura para el proyecto web de forma que se permita exponer la lógica a otras aplicaciones:
   - Contenido `src/main/resources`: Recursos a usar por parte de la aplicación.
     - Archivo `form.html`: Página web HTML con el formulario básico para ingreso de un empleado.
     - Archivo `result.html`: Página web HTML básica con el resultado de la validación de un empleado.
   - Paquete `edu.eci.pdsw.servlet`: Se encuentra un servlet para atender las peticiones web.
     - Clase `ValidateServlet`: Servlet con manejo a peticiones tipo *GET* y *POST* con respuestas en HTML.
2. Revise la implementación del servlet, donde se da soporte a peticiones de tipo *GET* y *POST* de la siguiente manera:
   - `GET /validate`: Recibe peticiones tipo *GET* en el endpoint `/validate` y carga el formulario de registro.
   - `POST /validate`: Recibe peticiones tipo *POST* en el endpoint `/validate` y valida el empleado con los parámetros que se envían por parámetro.
   - Revise la implementación de los métodos y agregue las funcionalidades pendientes (Tipos de contenido, creación de objeto, códigos de respuesta, etc.) de forma que se encuentren acordes al código implementado y a la funcionalidad requerida.
3. Ingrese a la página expuesta por el servlet para visualizar el formulario donde se permite el registro de empleados.
4. En la página del formulario, ingrese algunos datos para probar la implementación del método POST y la correcta respuesta ante algunas entradas.

## Entrega

- Cargar en Moodle antes de finalizar el parcial.
- Comprima todo el contenido del proyecto en un archivo .zip (excluyendo la carpeta target) y agreguela al esapcio correspondiente en Moodle.