Escuela Colombiana de Ingeniería

Ciclos de Vida del Desarrollo de Software – CVDS

Parcial Práctico - 1er Tercio

Parte I. - Calcular reservas

Para la primera parte del parcial, se creará un proyecto encargado de realizar los cálculos correspondientes para verificar si una reserva en una aerolínea es posible y el costo total de la misma.

1. Arbol del proyecto del parcial

```
.
+-- pom.xml
+-- README.html
+-- README.md
+-- README.pdf
+-- Wiki
    +-- QuickTheories.md
+-- src
    +-- main
    |  +-- java
    |  |  +-- edu
    |  |      +-- eci
    |  |          +-- cvds
    |  |              +-- calculator
    |  |              |  +-- AirlineCalculator.java
    |  |              |  +-- BookingCalculator.java
    |  |              +-- model
    |  |              |  +-- BookingOutput.java
    |  |              |  +-- BookingResult.java
    |  |              |  +-- SeatCategory.java
    |  |              +-- servlet
    |  |                 +-- BookingServlet.java
    |  +-- resources
    |      +-- form.html
    |      +-- result.html
    +-- test
        +-- java
            +-- edu
                +-- eci
                    +-- cvds
                        +-- validator
                            +-- AirlineCalculatorTest.java
```

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

continuación:

- Paquete `edu.eci.cvds.calculator`: Se encuentra la lógica de negocio.
  - Interfaz `BookingCalculator`: Interfaz principal para cargar la lógica de negocio para realizar los cálculos requeridos.
  - Clase `AirlineCalculator`: Clase principal de la lógica de negocio, donde se realiza el cálculo y validaciones de la reserva.
- Paquete `edu.eci.cvds.model`: Tiene las clases básicas del modelo.
  - Clase `BookingOutput`: Clase de resultado que indica si la reserva es posible y el valor (si aplica).
  - Enumeración `BookingResult`: Enumeración con 3 tipos de resultado.
  - Enumeración `SeatCategory`: Enumeración con 3 tipos básicos de categorías para las sillas (primera clase, económica y salida de emergencia).
- Paquete `edu.eci.cvds.calculator` (en src/test/java): Se encuentran las pruebas de la lógica de negocio.
  - Clase `AirlineCalculatorTest`: En esta clase deben estar las pruebas para `AirlineCalculator`.

3. Para el cálculo del precio para una reserva en la aerolínea, se tienen las siguientes condiciones:

   - Número de sillas:
     - Debe ser un numérico entre 1 y 100
   - Categoría:
     - Primera Clase:
       * El valor por silla es $100
       * La aerolínea no permite comprar más de 15 sillas de esta categoría en una sola reserva
     - Clase Económica:
       * El valor por silla es $50
       * La aerolínea no permite comprar más de 50 sillas de esta categoría en una sola reserva
     - Salida de Emergencia:
       * El valor por silla es $50
       * Los aviones tienen solo 8 salidas de emergencia
   - Precios:
     - Aparte de las condiciones anteriormente nombradas, se tienen los siguientes descuentos:
       * 2% si se compran más de 5 sillas en la misma reserva
       * 10% si se compran al menos 10 sillas en la misma reserva
       * 20% si se compran al menos 15 sillas en la misma reserva

4. Implemente con la librería QuickTheories, un generador de reservas que me permita obtener una gran cantidad de escenarios distintos para validar si es posible registrar la reserva y calcular el valor total.

5. En caso de ser necesario, agregue más tipos de error a la enumeración `BookingResult`, con diferentes clases de equivalencia para el problema.
6. Implemente la lógica específica en la clase `AirlineCalculator` para cumplir con las restricciones de validaciones y precio, 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 una reserva y costo total de la misma 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.cvds.servlet`: Se encuentra un servlet para atender las peticiones web.
     - Clase `BookingServlet`: 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 /booking`: Recibe peticiones tipo *GET* en el endpoint `/booking` y carga el formulario de registro.
   - `POST /booking`: Recibe peticiones tipo *POST* en el endpoint `/booking` para validar la reserva y calcular el valor total con los parámetros que se envían.
   - Revise la implementación de los métodos y agregue las funcionalidades pendientes (Tipos de contenido, creación de objetos, mapeo de tipo de datos, códigos de respuesta, etc.) de forma que se encuentren acordes al código implementado y a la funcionalidad requerida. Revise los "TODO" para guiarse respecto a los cambios.
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.