Orientación a Objetos II - 10/6/2023

Ejercicio 1 - Patrones

Se quiere implementar una aplicación para seguros de vehículos. Cada seguro estará asociado a un vehículo en específico, y una persona podrá contar con múltiples vehículos asegurados. La compañía brinda 3 tipos de seguros. Cada seguro incluye una cobertura de vida, una cobertura de daños a otros vehículos y una cobertura de daños al vehículo propio, según se muestra a continuación:

Seguro contra terceros	Seguro terceros completo	Seguro todo riesgo
Costo del seguro de vida: contempla solo el **seguro del conductor**: **100** pesos multiplicado por la edad del **asegurado**.	Costo del seguro de vida: Al **seguro del conductor** se le suma un monto de **5.000** pesos multiplicado por la cantidad máxima de ocupantes que posee el vehículo	Costo del seguro de vida: Al **seguro del conductor** se le suma un monto de **9.000** pesos multiplicado por la antigüedad la antigüedad del vehículo asegurado
Costo de la cobertura de daños a otros vehículos: **1.000** pesos más el 1% del valor del vehículo.	Costo de la cobertura de daños a otros vehículos: **4.000** pesos multiplicado por la antigüedad del auto si el auto tiene más de 4 años o **10.000** pesos si el auto tiene hasta 4 años.	Costo de la cobertura de daños a otros vehículos: **100.000** pesos dividido la edad del conductor.
Costo de la cobertura de destrucción total del vehículo propio por accidente: **0.5%** del valor del auto asegurado	Costo de la cobertura de destrucción total del vehículo propio por accidente o incendio: **0.5%** del valor del auto asegurado + **10.000** pesos	Costo de la cobertura de destrucción total o parcial del vehículo propio por accidente o incendio: **0.5%** del valor del auto asegurado + **1.000** pesos por la antigüedad del auto

La empresa cuenta con promociones disponibles para sus asegurados, pero planea agregar nuevas promociones en un futuro. Las promociones son: (i) promoción por múltiples pólizas, en donde se le descuenta al asegurado un 10% a cada una de los seguros contratados, si tiene al menos 2 pólizas, (ii) promoción por campaña excepcional, se le descuenta el 50% del costo del seguro más económico que tiene contratado y (iii) sin promoción, es decir, no se le realiza ningún descuento.

A los nuevos asegurados, la empresa asigna una promoción específica. Sin embargo, la empresa puede cambiarla en cualquier momento.

Usted debe brindar una solución para calcular el monto a abonar correspondiente a un asegurado. Este costo final es la suma de los costos de sus seguros aplicando la bonificación por la promoción que posea.

***Ayuda**: Para calcular los años entre dos fechas puede utilizar la siguiente expresión*

```
ChronoUnit.YEARS.between(fecha1, fecha2);
```

Tareas:

1. Modele una solución para el problema planteado. Si utiliza patrones, indique nombre y las ventajas que agrega en este diseño en particular
2. Implemente en Java la funcionalidad requerida.
3. Implemente un test para el siguiente escenario. Un asegurado de 50 años cumplidos, no tiene promoción, y asegura dos autos con seguro contra Terceros: un Renault 11 año 1988 valuado en 680.000 pesos, con capacidad de 4 personas y un Renault Clio año 2001 valuado en 1.200.000 pesos con capacidad de 5 personas.

Notas:

- Implemente **todos** los constructores que considere necesarios.
- Si necesita getters y setters, puede implementar uno de cada uno y asumir la existencia del resto.

Ejercicio 1 - Patrones

OO2 Parcial 3ra fecha - 15/7/2023

Se desea implementar un framework de diseño web orientado a componentes visuales que ofrezca los siguientes: Etiqueta, Botón y Contenedor.

- Cada etiqueta tiene texto y estilo actual.
- Cada botón tiene texto, una referencia a la acción que dispara y estilo actual.
- Cada contenedor tiene los elementos que incluye -los cuales pueden ser botones, etiquetas u otros contenedores.

En todos los casos, el estilo es un texto que indica las características visuales del elemento a ser dibujado, por ejemplo **"color:red; text-size:10px;"**.

En el framework se quiere incluir la funcionalidad de adecuar la apariencia de los elementos a la configuración visual que haya elegido el usuario. Para ello es necesario colaborar con instancias de la clase ConfiguracionDeEstilo, las cuales retornan strings de configuraciones de estilo (similares al ejemplo anterior) a través de estos dos mensajes:

public String getEstiloBoton()
public String getEstiloEtiqueta()

Cualquiera de los componentes visuales del framework debe responder al siguiente mensaje
public void aplicarEstilo (ConfiguracionDeEstilo estilo)

- Cuando un botón o una etiqueta recibe el mensaje, debe aplicar el estilo según corresponda.
- Cuando un contenedor recibe este mensaje, debe propagarlo a cada uno de los elementos que contiene.

public String imprimir()

- Cuando un botón o una etiqueta recibe este mensaje, se imprime individualmente.
- Cuando un contenedor recibe este mensaje, debe imprimir un encabezado y un cierre y permitir que sus elementos impriman su contenido.

Por ejemplo, para un Contenedor que incluye

- un botón con texto "Aceptar" con estilo "color:red; text-size:10px;" y acción "accept()"
- una etiqueta con el texto "Parcial de Objetos" y estilo "color:blue; text-size:12px;"

el resultado del llamado a imprimir al contenedor debería ser:

```
<contenedor>
<boton estilo="color:red; text-size:10px;"  texto="Aceptar" accion="accept()"/>
<etiqueta estilo="color:blue; text-size:12px;"/> Parcial de objetos </etiqueta>
</contenedor>
```

Por otro lado, se desea integrar al framework antes descripto, un componente preexistente ya implementado con la clase ListaDesplegable, la cual no se puede modificar.
Esta clase contiene los métodos de clase y de instancia necesarios para su funcionamiento, pero en particular nos interesan los siguientes dos:

public void setStyle(String style)
public String print()

Para poder resolver su apariencia, se decide que la listaDesplegable utilice la misma configuración de estilo que utilizan las etiquetas.
Para poder imprimir una listaDesplegable dentro de un ejemplo similar al anterior, utilice el método `print()` que retorna el String esperado.

Pista: considere la impresión de cada elemento de forma independiente. Si bien la misma es similar para todos, esto no siempre será así, por lo tanto no considere abstraerla en un Template Method.

Tareas:

1. Modele una solución para el problema planteado. Si utiliza patrones, indique nombre y roles en su diseño y las ventajas que agrega en este dominio en particular.
2. Implemente en Java la funcionalidad requerida.
3. Implemente un test para verificar la funcionalidad de imprimir con el ejemplo detallado anteriormente.

Parcial de OO2 - Curso 2022 - 16/jul/2022

Ejercicio 1 - Patrones

Sea una estación meteorológica hogareña que permite conocer información de varios aspectos del clima. Esta estación está implementada con la clase HomeWeatherStation que interactúa con varios sensores para conocer fenómenos físicos. La misma implementa los siguientes métodos:

```
//retorna la temperatura en grados Fahrenheit
public double getTemperaturaFahrenheit()

//retorna la presión atmosférica en hPa
public double getPresion()

//retorna la radiación solar
public double getRadiacionSolar()

//retorna una lista con todas las temperaturas sensadas hasta el momento, en
grados Fahrenheit
public List<Double> getTemperaturasFahrenheit()
```

Esta clase se encuentra implementada por terceros y **no se puede modificar.**

Nos piden construir una aplicación que además de lo anteriormente descripto pueda obtener:

- La temperatura en grados Celsius (°C = (°F-32) ÷ 1.8).
- El promedio de las temperaturas históricas en grados Fahrenheit.

Además, la aplicación debe permitir al usuario configurar qué datos mostrar y en qué orden. Esto significa que podría querer ver la información de muchas maneras, por ejemplo:

- *Ejemplo 1: "Presión atmosférica: 1008"*
- *Ejemplo 2: "Presión atmosférica: 1008 Radiación solar: 500"*
- *Ejemplo 3: "Radiación solar: 500 Temperatura C: 28 Promedio de temperaturas C: 25"*

Para ello, usted debe proveer en algún punto de su solución, la implementación del mensaje public String displayData() que devuelva los datos elegidos en el orden configurado (dado que la app aun no cuenta con interface de usuario).

Haga uso de la clase HomeWeatherStation sin modificarla.

Tareas:
1- Modele una solución para el problema planteado. Si utiliza algún patrón, indique cuál
2- Implemente en Java
3- Implemente un test para validar la configuración del ejemplo 2, asumiendo que en el momento de la ejecución del mismo, los sensores arrojan los valores del ejemplo.

Ejercicio 1 - Patrones

OO2 Parcial 2da fecha - 1/7/2023

Consideremos una aplicación que administra tareas. Pueden existir tareas simples y tareas complejas. Estas últimas requieren a su vez otras tareas. Por ejemplo, "ir al supermercado" e "ir a la verdulería" son tareas simples. Y el siguiente es un ejemplo de una posible tarea compleja.

Ejemplo:

"preparar almuerzo"	(tarea compleja)
"realizar compras"	(tarea compleja)
"ir al supermercado"	(tarea simple) (estimado 3, comenzó 9, completada 11)
"ir a la verdulería"	(tarea simple) (estimado 1, comenzó 11, completada 12)
"cocinar"	(tarea simple) (estimado 2, comenzó 12)
"preparar la mesa"	(tarea simple) (estimado 1)

El ciclo de vida de las tareas simples se desarrolla de la siguiente manera: en primer lugar, se crean asignándoles un nombre y una duración estimada. Posteriormente, se inician y se registra el momento exacto en que comienzan. Finalmente, se completan y se registra el instante en que se completaron. No es posible completar una tarea que no se haya iniciado previamente, ni una que ya se encuentre completada. Tampoco es posible iniciar una tarea que se encuentre completada.

Las tareas se describen como sigue:

- Todas las tareas cuentan con una descripción.
- Las tareas simples tienen una duración estimada en horas (por simplicidad consideremos un número entero mayor a cero) en que se deben completar.
- Las tareas complejas no tienen duración estimada ni horas de comienzo ni fin. La duración estimada es la suma de las duraciones estimadas de las tareas que contiene. Y si se quiere iniciar o finalizar una tarea compleja, se propaga la acción a las tareas que contiene, pero no se deja registro en la tarea compleja.

Para simplificar, consideraremos el uso de números enteros para representar los momentos de inicio y completitud de una tarea simple. Para ello, puede utilizar System.out.currentTimeMillis().

Se necesita brindar la siguiente funcionalidad sobre las tareas:

(i) estimación total de una tarea.
Para una tarea simple, es su duración estimada.
Para una tarea compleja, es la suma de las duraciones estimadas de las tareas que abarca.

(ii) tiempo utilizado en una tarea. Para una tarea simple completada es el instante de fin - el de inicio. Para cualquier otro caso de una tarea simple es cero. Para una tarea compleja es la suma de los tiempos utilizados de las tareas que contiene.

(iii) avance de una tarea. Es el cociente entre el tiempo utilizado de una tarea y su estimación total.
Para el ejemplo presentado anteriormente, sobre la tarea compleja "preparar almuerzo", tendríamos:
Estimación total: 7 horas
Tiempo utilizado: 3 hs (dado por las únicas dos tareas simples que se completaron)
Avance de la tarea: 3/7 = 0.43

(iv) iniciar una tarea. Para una tarea simple esta operación registra el instante en que se inició y a partir de este momento se la considera iniciada. Si la tarea simple ya ha sido iniciada o completada, no hace nada. Para una tarea compleja, se propaga la acción de iniciar a todas sus subtareas.

(v) completar una tarea. Para una tarea simple esta operación registra el instante en que se completó la misma y a partir de este momento se la considera completada. Si la tarea simple no ha sido iniciada, o si ya ha sido completada, no hace nada. Para una tarea compleja, propaga la acción de completar a todas sus subtareas.

Tareas:

1. Modele una solución para el problema planteado en un diagrama de clases UML. Si utiliza patrones, indique nombre y roles en su diseño y explique las ventajas que agrega en este dominio en particular.
2. Implemente en Java la funcionalidad requerida.
3. Implemente un test de la funcionalidad "avance de una tarea" para el ejemplo presentado.

Notas:

- Implemente **todos** los constructores que considere necesarios.
- Si necesita getters y setters, puede implementar uno de cada uno y asumir la existencia del resto.

Parcial de OO2 - 2022 - 8/ago/2022

Consideremos una empresa que brinda servicios y los gestiona a través de proyectos. Los proyectos tienen una fecha de inicio y de fin, un objetivo, un número de integrantes (quienes cobran un monto fijo por día) y un margen de ganancia. Durante el armado del proyecto, el mismo debe pasar por un proceso de aprobación que involucra las etapas: En construcción -> En evaluación -> Confirmada. Se desea implementar la siguiente funcionalidad:

Funcionalidad	Etapa del proyecto	Resultado esperado
Crear proyecto	-	Se crea el proyecto en etapa "En construcción" con nombre, fecha de inicio y fin, objetivo, margen de ganancia de 7%, un número de integrantes y el monto de pago por integrante por día.
Aprobar etapa	En construcción	El proyecto pasa a etapa "En evaluación" siempre y cuando su precio no sea 0 (cero). De lo contrario genera un error.
	En evaluación	El proyecto pasa a etapa "Confirmada"
	En otra situación	No produce efecto alguno en el proyecto.
Costo del proyecto	En cualquier etapa	Retorna la suma de los costos de las personas involucradas. Considerar que las personas trabajan todos los días que dura el proyecto.
Precio del proyecto	En cualquier etapa	Retorna el valor obtenido luego de aplicar el margen de ganancia al costo del proyecto.
Modificar margen de ganancia	En etapas "En construcción" y "En evaluación"	Actualiza el margen de ganancia si se encuentra en los siguientes valores: Para "En construcción" -> valores entre 8% y 10% Para "En evaluación" -> valores entre 11% y 15% Para valores fuera de los rangos permitidos no produce efecto alguno en el proyecto.
	Otra situación	No produce efecto alguno en el proyecto.
Cancelar proyecto	En cualquier etapa	Agrega "(Cancelado) " al objetivo del proyecto. Deja el proyecto cancelado.
	Si ya está Cancelado.	No produce efecto alguno en el proyecto.

Tareas:

1- Modele una solución y provea el diagrama de clases UML para el problema planteado. Si utiliza algún patrón, indique cuál.

2- Implemente en Java.

3- Implemente un test para aprobar un proyecto con las siguientes características: (i) se encuentra en evaluación, (ii) se llama "Vacaciones de invierno", (iii) tiene como objetivo "salir con amigos", y (iv) lo integran 3 personas.

Nota: para generar o levantar un error debe utilizar la expresión

```
throw new RuntimeException("Este es mi mensaje de error");
```