Patrones

OO2 - Parcial 28/06/2025

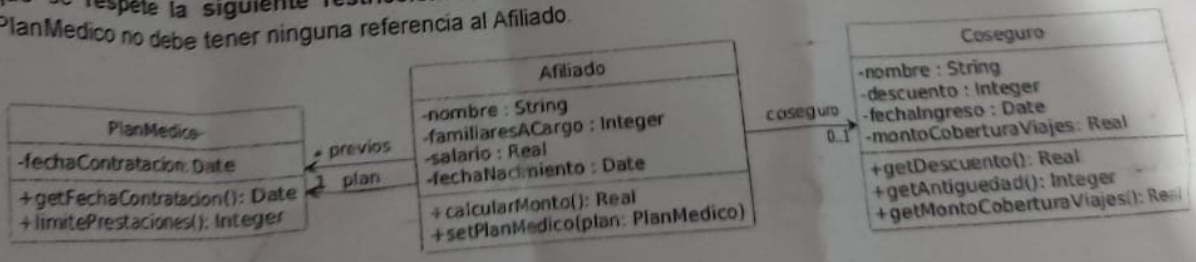
En una aplicación de seguros médicos, se registran los afiliados y el plan médico que contrataron. Existen 3 tipos de planes: plan médico obligatorio, plan integral y plan premium. Cada afiliado tiene un plan, que puede ser cambiado cuando lo solicite. Además, algunos afiliados cuentan con coseguro, que les otorga descuentos sobre el monto a pagar por el plan. El coseguro tiene un nombre, una fecha de ingreso y un porcentaje de descuento específico.

El costo mensual de un plan se calcula como la suma de un monto fijo, más un cargo por grupo familiar, más un adicional por cobertura viajera y, según el tipo de plan, un seguro adicional por internación.

Plan médico obligatorio	Plan integral	Plan premium
Monto fijo: $15.000	Monto fijo: $22.000	Monto fijo: $33.000 Si tiene coseguro, se aplica el descuento del coseguro
Cargo por grupo familiar: $3.500 por cada integrante. Si tiene coseguro, aplica el descuento del coseguro por cada integrante	Cargo por grupo familiar: $3.000 por cada integrante + 1% del salario del afiliado	Cargo por grupo familiar: Hasta 4 integrantes es sin cargo; luego, $2.800 por cada integrante adicional
Costo por cobertura viajera: 1% del salario del afiliado. Si tiene coseguro se le descuenta el monto por cobertura de viajes	Costo por cobertura viajera: 3% del salario del afiliado. Si tiene coseguro se le descuenta $10.000 pesos por año de antigüedad en el coseguro	Costo por cobertura viajera: 1% del salario del afiliado. Si tiene coseguro se le descuenta el monto por cobertura de viajes
Seguro por internación: No posee	Seguro por internación: 5% del monto fijo	Seguro por internación: 5% del monto fijo

Por ejemplo, Pedro, un afiliado con un plan médico obligatorio, de 27 años, un salario de $100.000, con 2 familiares a cargo, debe pagar por su plan médico $15.000 por el monto fijo, $7.000 por los integrantes a cargo, $1.000 por costo de cobertura viajera, siendo el total $23.000.

Usted debe brindar una solución que permita calcular el monto a cobrar por el plan médico contratado por un afiliado, utilizando el diseño propuesto. Puede ser modificado o extendido según lo considere necesario, siempre que se **respete la siguiente restricción de diseño**: la clase Afiliado debe conocer a su PlanMedico, pero PlanMedico no debe tener ninguna referencia al Afiliado.



Nota: Si utiliza constructores, debe implementarlos.
Para calcular los años entre dos fechas puede usar: `ChronoUnit.YEARS.between(diaAnterior, diaActual)`

Tareas (no deje ningún ítem en blanco):

1. Implemente su solución en Java. Utilice Template Method para solucionar el cálculo del monto a Cobrar para un Plan médico. Puede incorporar otros patrones de diseño según considere necesario.
2. Realice un diagrama de clases UML indicando claramente los roles de los patrones utilizados.
3. Escriba un test para verificar el cálculo del monto a pagar para el afiliado presentado en el ejemplo. Cambie su plan a Integral y verifique el cálculo tanto con coseguro como sin coseguro.

Refactoring

OO2 - Parcial 28/06/2025

La plataforma de renta de vehículos "RentaOOs" está implementada de la siguiente manera. Actualmente ofrece tres tipos de rentas, básico, plus, y kilometraje libre, que se puede cambiar en cualquier momento durante la renta del vehículo.

```
1  public class Renta {
2    private Vehiculo vehiculo;
3    private Cliente cliente;
4    private int diasRenta;
5    private String tipoRenta;
6    private int kilometrajeInicial;
7
8    public Renta(Vehiculo vehiculo,
9                 Cliente cliente, int diasRenta) {
10   this.vehiculo = vehiculo;
11   this.cliente = cliente;
12   this.diasRenta = diasRenta;
13   this.kilometrajeInicial =
14             vehiculo.getKilometraje();
15   this.tipoRenta = "BASICO";
16 }
17
18 public void setTipoRenta(String tipoRenta){
19   this.tipoRenta = tipoRenta;  }
20
21 public double calcularTotal() {
22   if (this.tipoRenta == "BASICO") { (1) SWITCH STATEMENT
23     int kilometrosRecorridos = vehiculo.getKilometraje() - this.kilometrajeInicial; (2) PUBLIC ORD ATR
24     double precio = diasRenta * vehiculo.getPrecioPorDia()
25                 + kilometrosRecorridos * vehiculo.getPrecioPorKm(); (3)
26     double adicional = 1;
27     // los autos más viejos tienen un 15% de descuento
28     if (vehiculo.getAntiguedad() > 5) { (5)
29       adicional  = 0.85;
30     }
31     return precio * adicional;
32   }
33   else if (this.tipoRenta == "PLUS") {
34     int kilometrosRecorridos = vehiculo.getKilometraje() -  this.kilometrajeInicial; (3)
35     return kilometrosRecorridos *  vehiculo.getPrecioPorKm();
36   } else {  // si el tipo de renta es "KILOMETRAJE_LIBRE":
37     return diasRenta * vehiculo.getPrecioPorDia();
38 } } }
```

```
public class RentaTest {
 @Test
 public void testBasicoYKilometrajeLibre() {
  int precioPorDia = 100;
  int precioPorKm = 5;
  int antiguedad = 4;
  int kilometraje = 10000;
  Vehiculo vehiculo = new Vehiculo(kilometraje,
            precioPorDia, precioPorKm, antiguedad);
  Cliente cliente = new Cliente("Ana");
  Renta renta = new Renta(vehiculo, cliente, 3);

   // Simulamos uso del vehículo
  vehiculo.aumentarKilometraje(200);
  double esperadoBasico = (3 * 100 + 200 * 5);
  assertEquals(esperadoBasico, renta.calcularTotal());

  renta.setTipoRenta("KILOMETRAJE_LIBRE");
  double esperadoLibre = 3 * 100;
  assertEquals(esperadoLibre, renta.calcularTotal());
 }
}
```

```
public class Vehiculo {
 public double getPrecioPorDia() {...}
 public int getKilometraje() {...}
 public int getAntiguedad() {...}
 public double getPrecioPorKm() {...}
  public void aumentarKilometraje(int
cantidadKm) {...}  }
```

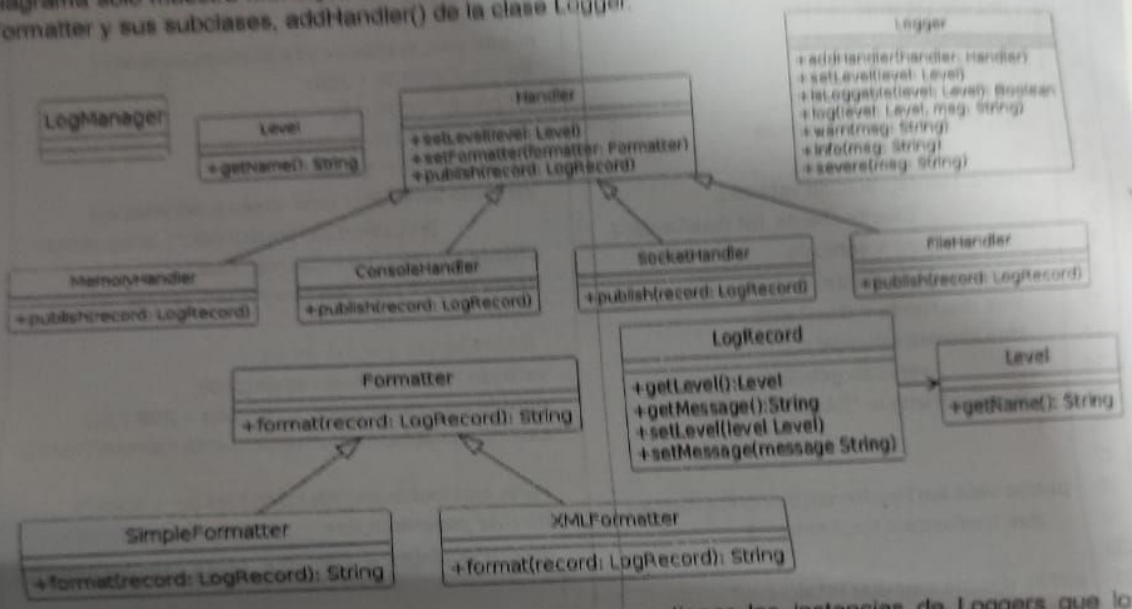
Tareas: (debe realizar los tres ítems para aprobar el tema)

1. Enumere los code smell que encuentra en el código indicando las líneas afectadas.
2. Indique que refactorings utilizará para solucionar cada uno de los smells. Explique los pasos necesarios para realizar los refactorings elegidos, haciendo referencia al código cuando corresponda. Muestre el código final resultante luego de aplicar esos refactorings en la clase Renta, y si hace falta, en la clase RentaTest.
3. Realice el diagrama de clases del código refactorizado. Si utilizó un patrón de diseño, indíquelo en el diagrama mostrando los roles del patrón.

Frameworks

OO2 - Parcial 28/06/2025

Consideremos el framework de Logging visto en la materia. El framework permite registrar eventos importantes, errores, etc. Estos logs son utiles para desarrolladores, administradores y usuarios. El framework permite enviar mensajes de log a consola, archivos, sockets, etc.
Como ayuda memoria se ofrece un diagrama simplificado de algunas clases del framework. El diagrama sólo muestra mensajes relevantes como publish() en Handler y sus subclases, format() en Formatter y sus subclases, addHandler() de la clase Logger.



Recuerde que el framework se encarga de crear y gestionar las instancias de Loggers que los desarrolladores de una aplicación utilizarán para loguear. La configuración de los loggers generalmente se realiza al iniciar la aplicación (y se mantiene globalmente). Esto significa que se pueden establecer niveles de log, formatos de salida, destinos de los logs y otras opciones relacionadas con la generación y gestión de logs.

Se requiere desarrollar una funcionalidad para una aplicación que utiliza el framework de logging estándar de Java. La aplicación debe generar mensajes de log en formato HTML, de la siguiente forma: `<div class="log info">Mensaje</div>`.

Tareas (no deje ningún ítem en blanco):

1. ¿Qué hotspot del framework se debe utilizar para implementar esta funcionalidad? Justifique brevemente.
2. Implemente la funcionalidad requerida.
3. Muestre un ejemplo de código que configure un Logger para utilizar el formato HTML implementado, escribiendo la salida a consola o archivo.
4. Considerando su implementación, indique las opciones correctas. Puede escribir solo la letra de cada opción elegida en la hoja del examen.
   A. La inversión de control ocurre en la clase Handler, al setear el nivel (level)
   B. La inversión de control ocurre cuando el Logger delega en el handler el manejo del mensaje a loguear.
   C. La inversión de control ocurre en el handler al momento de formatear el mensaje.
   D. En este caso no hay inversión de control

...mente los 3 ejercici

1	5/6
2	3/3
3	5/5

Responda verdadero (V) o falso (F) en cada caso:

(F) El patrón "Null Object" es uno de los patrones conocidos como "wrappers". X
(F) Los tests de unidad se programan una vez que el programa no sufrirá más cambios.
(F) TDD es una metodología cuyo objetivo es implementar "test de unidad"
(V) El refactoring "rename variable" cambia todas las ocurrencias del string que es el nombre de la variable a renombrar.
(F) Un framework de caja negra se reusa a través de la subclasificación.
(F) Los frameworks siempre tienen más frozen spots que hot spots. — [illegible]

No se cuando aplican siempre

Ejercicio 2.
Considere cada caso y seleccione el patrón que mejor aplica al problema (no justifique)

Caso 1: Se desea implementar un sistema que calcula recorridos de camiones en una ciudad. Se quiere implementar diferentes maneras de calcular el recorrido como por ejemplo: "Más rápido", "Más corto", y "Menor tránsito".

Builder X	NullObject X	Strategy ✓	Adapter X

Caso 2: Una empresa de viajes planea implementar un programa para organizar paquetes turísticos para fin de semanas largos a Carlos Paz. Los paquetes incluyen: transporte, alojamiento, comidas y dos excursiones. Se ofrecen paquetes económicos (Micro, Hotel**, 2 comidas y dos caminatas por las sierras) y VIP (Avión, Hotel****, 4 comidas, dos salidas en bote).

Builder ✓	NullObject X	Strategy X	Adapter X

Caso 3: Un banco tiene un sistema que usa una jerarquía de productos financieros tradicionales. Todos estos productos son polimórficos. Un nuevo producto crypto quiere ser integrado al sistema producto crypto fue desarrollado por otra empresa y no es polimórfico. Por razones contractuales se puede cambiar el producto crypto.

uilder X	NullObject X	Strategy X	Adapter ✓

a uno de los problemas que aparecen a la izquierda co
derecha con el que se puede asociar:

en una clase que usan
ite los datos y métodos de otra
ra realizar su trabajo

clase tiene demasiadas
abilidades.

todos en una misma clase que
gica para diferentes tipos de

efinidos en una clase son solo
sus variables de instancia.

querimientos requiere
varios métodos en el