

Ejercicio 1- Envío de paquetes

Se quiere mejorar un sistema de envíos de paquetes que permite registrar los paquetes enviados y calcular sus costos.

De los paquetes se conoce: el valor declarado, la dirección de destino, el destinatario y una descripción, como puede verse en el diagrama UML. El costo de envío es el 5% del valor declarado.

Paquete
-descripcion: String
-destinatario: String
-direccionDestino: String
-valorDeclarado: Real
+getDescripcion(): String
+getDestinatario(): String
+getDireccionDestino(): String
+getValorDeclarado(): Real
+getCostoEnvio(): Real

Se plantea la incorporación de servicios adicionales para personalizar la entrega, mejorando la seguridad, velocidad en el envío o manejo especial del paquete, sin modificar la clase Paquete mencionada anteriormente, pero permitiendo que la clase pueda implementar una interfaz en caso de ser necesario.

Los servicios adicionales que se quieren implementar son:

- **Con seguro:** Incrementa el costo de envío en un 20% del valor declarado y agrega la nota "con seguro" a la descripción.
- **Con seguimiento en tiempo real:** permite que el cliente siga el trayecto en tiempo real. Esto incrementa el costo de envío en 2000 pesos, pero no modifica la descripción.
- **Entrega exprés:** Incrementa el costo de envío en un 50% del valor declarado y agrega la nota "entrega express" a la descripción.
- **Manipulación frágil:** No incrementa el costo, pero agrega la nota "frágil" a la descripción.

Se pide que los servicios puedan combinarse libremente.

Por ejemplo, un paquete descrito como una "Caja de libros" con un valor declarado de \$20.000 que incluye los servicios adicionales de seguro (\$4.000) y entrega express (\$10.000), resulta en

- Costo de envío: 5%: \$ 1.000
- Seguro 20% de 20.000: \$ 4.000
- Entrega express 50% de 20.000: \$ 10.000
- Costo final del envío: \$ 15.000

y la descripción de "Caja de libros con seguro entrega express".

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

1. Realice un diagrama UML de clases para su solución al problema planteado.
2. Indique claramente en el diagrama UML el o los patrones de diseño que utiliza en el modelo y el rol que cada clase cumple en cada uno.
3. Implemente su solución en Java
4. Escriba un test que muestre cómo instanciar y utilizar su solución para el caso de la "Caja de libros" con los servicios detallados anteriormente.

Dado el siguiente código que implementa la generación de reportes

```

1 public class ReportGenerator {
2     private String type;
3     public ReportGenerator(String type) { this.type = type; }
4
5     public void generateReport(Document document) {
6         if ("PDF".equals(type)) {
7             // crear documento y configurar metadatos
8             DocumentFile docFile = new DocumentFile();
9             docFile.setTitle(document.getTitle());
10            String authors = "";
11            for (String author : document.getAuthors()) {
12                authors += ", " + author;
13            }
14            docFile.setAuthor(authors);
15            docFile.contentType("application/pdf");
16            docFile.setPageSize("A4");
17
18            // crear exportador y setear el contenido
19            PDFExporter exporter = new PDFExporter();
20            byte[] content = exporter.generatePDFFile(document);
21            docFile.setContent(content);
22
23            // guardar el documento
24            this.saveExportedFile(docFile);
25        } else if ("XLS".equals(type)) {
26            // crear documento y configurar metadatos
27            DocumentFile docFile = new DocumentFile();
28            docFile.setTitle(document.getTitle());
29            String authors = "";
30            for (String author : document.getAuthors()) {
31                authors += ", " + author;
32            }
33            docFile.setAuthor(authors);
34            docFile.contentType("application/vnd.ms-excel");
35            docFile.setSheetName(document.getSubtitle());
36
37            // crear exportador y setear el contenido
38            ExcelWriter writer = new ExcelWriter();
39            byte[] content = writer.generateExcelFile(document);
40            docFile.setContent(content);
41
42            // guardar el documento
43            this.saveExportedFile(docFile);
44        }
    
```

```

class ReportGeneratorTest {
    ReportGenerator generatorPDF;
    ReportGenerator generatorXLS;
    Document document;

    @BeforeEach
    void setUp() {
        document = new Document("Informe");
        document.addAuthor("Carlos");
        document.addAuthor("Ana");

        generatorPDF = new ReportGenerator("PDF");
        generatorXLS = new ReportGenerator("XLS");
    }

    @Test
    void testPDF() {
        generatorPDF.generateReport(document);

        // aca se detallan los asserts
    }

    @Test
    void testXLS() {
        generatorXLS.generateReport(document);

        // aca se detallan los asserts
    }
}
    
```

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 solucionarlos considerando que se desea incluir nuevos formatos para exportar (XLSX, CSV entre otros). Explique los pasos necesarios para realizar los refactorings elegidos, haciendo referencia al código cuando corresponda. Muestre el código final resultado luego de aplicar esos refactorings en la clase ReportGenerator y si hace falta, en la clase ReportGeneratorTest.
3. Realice el diagrama de clases del código refactorizado. Si utilizó un patrón de diseño, indíquelo en el diagrama mostrando los roles del patrón.

Consideremos una parte del framework SingleThreadTCPServer que permite construir servidores TCP en Java. El framework define el método handleClient que se encarga de leer los mensajes que envía el cliente, procesarlos y responder.

A) SingleThreadTCPServer
<pre>+startLoop(args: String[]): void #checkArguments(args: String[]): void #displayUsage(): void #displaySocketInformation(portNumber: Int): void #acceptAndDisplaySocket(serverSocket: ServerSocket): Socket #displaySocketData(clientSocket: Socket): void #displayAndExit(portNumber: int): void -handleClient(clientSocket: Socket): void +«abstract» handleMessage(message: String, out: PrintWriter): void</pre>

```

1 private final void handleClient(Socket clientSocket) {
2     try {
3         PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true);
4         BufferedReader in = new BufferedReader(
5             new InputStreamReader(clientSocket.getInputStream()));
6         String line;
7         while ((line = in.readLine()) != null) {
8             System.out.println("Received message: " + inputLine
9             + " from " + clientSocket.getInetAddress().getHostAddress()
10            + ":" + clientSocket.getPort());
11            .ozen es sb establece la ip y el puerto del cliente
12            if (line.isEmpty()) break; do asygo puden no hacerlo qdo es qmwe nati o
13            this.handleMessage(line, out);
14        }
15        System.out.println("Connection closed with " + clientSocket.getInetAddress().getHostAddress()
16        + ":" + clientSocket.getPort());
17        .ozen es etreamingshml cimeta ose, qd y qghdo cteun es obn e oce oj, oen
18    } catch (IOException e) {
19        System.err.println("Error: " + e.getMessage());
20    } finally {
21        try {
22            clientSocket.close();
23        } catch (IOException ignored) {}
24    }
25 }
```

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

- Explique brevemente cuáles son los frozen spot y hot spot del código presentado del framework.
- Modificar este framework para permitir personalizar los mensajes que se muestran en consola por defecto cuando un cliente se conecta y se desconecta.
- Explique brevemente cuáles son los frozen spot y hot spot después de la extensión realizada.
- Indique en dónde se produce la inversión de control en la extensión realizada.

OO2 - Preguntas para acceder a la Promoción

7/6/2025

Para acceder al examen de Promoción de la materia debe responder correctamente los 3 ejercicios que siguen.

Ejercicio 1.

Indique cómo se relaciona el patrón Strategy con los patrones numerados a la derecha. Para ello indique en cada uno de los aspectos que se mencionan a la izquierda los números de él/los patrones correspondientes (puede relacionarse con mas de uno).

- tiene similar propósito que 1) Template Method
- tiene similar estructura que 2) Null Object
- puede / suele usarse junto con 3) Decorator
4) State
5) Adapter
6) Composite

Ejercicio 2.

Seleccione todas las afirmaciones correctas sobre frameworks (puede ser mas de una).

- A. Un framework es una aplicación "semicompleta", "reusable", que puede ser especializada para producir aplicaciones/soluciones a medida.
- B. Framework se considera la manera menos sofisticada de reuso.
- C. Los frameworks de caja blanca son aquellos cuyas clases se encuentran en diferentes paquetes/módulos públicos.
- D. Un Hotspot es un elemento en el código que permite modificar el comportamiento/funcionalidad del framework.
- E. Cada clase en un framework resuelve un problema concreto, es independiente del contexto de uso, no espera nada de nuestro código y es generalmente independiente de otras clases en el framework.
- F. Un framework solo puede tener un frozen spot.

Ejercicio 3.

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

- () Hacer refactoring implica eliminar bugs (errores en la funcionalidad) o introducir cambios solicitados por el usuario.
- () Los code smells son indicadores de posibles problemas en el diseño del código fuente.
- () El refactoring "Extract Method" soluciona el mal olor "Envidia de atributo"
- () El refactoring "Pull Up Method" soluciona el mal olor de tener código duplicado en clases de una jerarquía.
- () Los test doubles se usan para testear valores de borde.
- () Puede ser provechoso adquirir deuda técnica para ganar feedback rápido.