**Manual Técnico del Sistema de Gestión de Envíos**

**Introducción**

Este manual proporciona información técnica detallada sobre el diseño, la implementación y el mantenimiento del Sistema de Gestión de Envíos de Paquetería.

1. **Arquitectura del Sistema**

-El sistema está desarrollado siguiendo un enfoque de programación orientada a objetos en Java.

-Se utiliza el patrón de diseño "Herencia" para modelar los diferentes tipos de paquetes.

-La lógica de cálculo del costo de envío se encapsula en la clase ServicioEnvio.

2. **Estructura de Clases**

**Paquete (clase abstracta)**

-Atributos: peso, dimensiones, direccionOrigen, direccionDestino

-Método abstracto: calcularCostoBase()

**PaqueteEstandar (clase concreta)**

-Hereda de Paquete

-Implementa calcularCostoBase() con un costo base fijo

**DocumentoUrgente (clase concreta)**

-Hereda de Paquete

-Atributo adicional: tiempoEntrega

-Implementa calcularCostoBase() con un costo base específico

**ArticuloFragil (clase concreta)**

-Hereda de Paquete

-Atributo adicional: esFragil

-Implementa calcularCostoBase() con un costo base específico

**ServicioEnvio (clase de servicios)**

-Atributos: tarifaBasePorPeso, tarifaBasePorDimensiones, tarifaFragil, tarifaUrgente

-Método principal: calcularCostoEnvio(Paquete paquete)

-Método auxiliar: sumarDimensiones(double[] dimensiones)

3. **Flujo de Ejecución**

-Se crean instancias de los diferentes tipos de paquetes (PaqueteEstandar, DocumentoUrgente, ArticuloFragil), proporcionando los valores correspondientes.

-Se crea una instancia de ServicioEnvio con las tarifas configuradas.

-Para cada paquete, se llama al método calcularCostoEnvio(Paquete paquete) de la instancia de ServicioEnvio, pasando el objeto del paquete como argumento.

-El método calcularCostoEnvio(Paquete paquete) realiza el cálculo del costo total del envío utilizando el costo base del paquete, las tarifas por peso y dimensiones, y los recargos adicionales según el tipo de paquete.

-El resultado del cálculo se devuelve y se utiliza según sea necesario en la aplicación.

6. **Extensibilidad y Mantenimiento**

-El diseño del sistema sigue los principios de la programación orientada a objetos, lo que facilita la extensibilidad y el mantenimiento.

-Para agregar un nuevo tipo de paquete, se debe crear una nueva clase que herede de Paquete e implemente el método calcularCostoBase() según las reglas de negocio específicas.

-Los cambios en las tarifas y recargos pueden realizarse actualizando los valores correspondientes en la clase ServicioEnvio.

7. **Pruebas y Depuración**

-Se recomienda implementar pruebas unitarias para cada clase, utilizando un marco de pruebas como JUnit.

-Para facilitar la depuración, se pueden agregar instrucciones de registro (System.out.println o un marco de registro como Log4j) en puntos estratégicos del código.

8. **Requisitos del Sistema**

-Versión de Java: Java 8 o superior

-Entorno de Desarrollo Integrado (IDE) recomendado: IntelliJ IDEA, Eclipse o NetBeans

-Bibliotecas adicionales: Ninguna (el sistema utiliza únicamente las bibliotecas estándar de Java)

9. **Despliegue y Configuración**

-El sistema puede ser desplegado como una aplicación de consola o integrado en una aplicación de escritorio o web más grande.

-Las tarifas y recargos pueden configurarse en el constructor de la clase ServicioEnvio o cargarse desde un archivo de configuración externo.

10. **Soporte y Mantenimiento**

- Información de contacto del equipo de desarrollo para soporte y mantenimiento.

- Procedimientos para informar problemas, solicitar nuevas funcionalidades y obtener actualizaciones.