UNIVERSIDAD TECNOLÓGICA DE PANAMÁ
FACULTAD DE INGENIERIA DE SISTEMAS COMPUTACIONALES
VICEDECANATO DE INVESTIGACIÓN, POSGRADO Y EXTENSIÓN
MAESTRÍA EN INGENIERÍA DE SOFTWARE
VERANO 2025

Profesor: Danilo Domínguez Pérez, PhD
Email: danilo.dominguez1@utp.ac.pa

Laboratrorio #1 - Principios SOLID y Patrones de Diseño
Fecha entrega:

DESCRIPCIÓN GENERAL

Este laboratorio está diseñado para aplicar de manera práctica los principios SOLID, patrones de diseño y conceptos de refactorización vistos en clase. A diferencia de la asignación teórica, este laboratorio se enfoca en la práctica colaborativa de refactorización de código real, análisis crítico y diseño de soluciones.

Objetivos de Aprendizaje:

- Aplicar principios SOLID en código existente
- Identificar y refactorizar code smells y antipatrones
- Diseñar soluciones aplicando los principios apropiados
- Realizar code review colaborativo
- Evaluar trade-offs en decisiones de diseño

SECCIÓN 1: ANÁLISIS Y REFACTORIZACIÓN

Objetivo

Refactorizar un sistema existente aplicando múltiples principios SOLID para mejorar su diseño y mantenibilidad.

Contexto

Se tiene un sistema de gestión de biblioteca que maneja préstamos de libros, notificaciones y reportes. El código actual tiene varias violaciones de principios SOLID y necesita ser refactorizado.

```
import java.util.*;
import java.time.LocalDate;
import java.time.temporal.ChronoUnit;

// Sistema de Gestión de Biblioteca - Versión con problemas de diseño
public class LibrarySystem {
```

```
    private List<Book> books = new ArrayList<>();
    private List<Member> members = new ArrayList<>();
    private List<Loan> loans = new ArrayList<>();

    // Violación múltiple de SOLID
    public void processLoan(String memberId, String bookId) {
        // Validar miembro
        Member member = null;
        for (Member m : members) {
            if (m.getId().equals(memberId)) {
                member = m;
                break;
            }
        }
        if (member == null) {
            System.out.println("Miembro no encontrado");
            return;
        }

        // Validar libro
        Book book = null;
        for (Book b : books) {
            if (b.getId().equals(bookId)) {
                book = b;
                break;
            }
        }
        if (book == null) {
            System.out.println("Libro no encontrado");
            return;
        }

        // Verificar disponibilidad
        if (!book.isAvailable()) {
            System.out.println("Libro no disponible");
            return;
        }

        // Verificar límite de préstamos
        int activeLoans = 0;
        for (Loan loan : loans) {
            if (loan.getMemberId().equals(memberId) && !loan.isReturned())
```

```
{
                activeLoans++;
            }
        }
        if (activeLoans >= 5) {
            System.out.println("Límite de préstamos alcanzado");
            return;
        }

        // Crear préstamo
        Loan loan = new Loan(UUID.randomUUID().toString(), memberId,
bookId, LocalDate.now());
        loans.add(loan);
        book.setAvailable(false);

        // Enviar notificación por email
        System.out.println("Enviando email a " + member.getEmail() +
                           ": Has prestado el libro " + book.getTitle());

        // Generar reporte
        System.out.println("=== REPORTE DE PRÉSTAMO ===");
        System.out.println("Miembro: " + member.getName());
        System.out.println("Libro: " + book.getTitle());
        System.out.println("Fecha: " + LocalDate.now());
        System.out.println("Fecha de devolución: " +
LocalDate.now().plusDays(14));
    }

    public void processReturn(String loanId) {
        Loan loan = null;
        for (Loan l : loans) {
            if (l.getId().equals(loanId)) {
                loan = l;
                break;
            }
        }
        if (loan == null) {
            System.out.println("Préstamo no encontrado");
            return;
        }

        loan.setReturned(true);
```

```
        Book book = null;
        for (Book b : books) {
            if (b.getId().equals(loan.getBookId())) {
                book = b;
                break;
            }
        }
        if (book != null) {
            book.setAvailable(true);
        }

        // Calcular multa si hay retraso
        long daysOverdue = ChronoUnit.DAYS.between(loan.getDueDate(),
LocalDate.now());
        if (daysOverdue > 0) {
            double fine = daysOverdue * 0.50;
            System.out.println("Multa por retraso: $" + fine);
            // Aquí se guardaría en base de datos
        }

        // Notificación
        Member member = null;
        for (Member m : members) {
            if (m.getId().equals(loan.getMemberId())) {
                member = m;
                break;
            }
        }
        if (member != null) {
            System.out.println("Enviando email a " + member.getEmail() +
                               ": Has devuelto el libro");
        }
    }

    public void generateMonthlyReport() {
        System.out.println("=== REPORTE MENSUAL ===");
        int totalLoans = 0;
        int totalReturns = 0;
        double totalFines = 0;

        for (Loan loan : loans) {
```

```
            if (loan.getLoanDate().getMonth() ==
LocalDate.now().getMonth()) {
                totalLoans++;
                if (loan.isReturned()) {
                    totalReturns++;
                }
                if (loan.getDueDate().isBefore(LocalDate.now()) &&
!loan.isReturned()) {
                    long days = ChronoUnit.DAYS.between(loan.getDueDate(),
LocalDate.now());
                    totalFines += days * 0.50;
                }
            }
        }

        System.out.println("Préstamos: " + totalLoans);
        System.out.println("Devoluciones: " + totalReturns);
        System.out.println("Multas pendientes: $" + totalFines);
    }
}

// Clases de soporte
class Book {
    private String id;
    private String title;
    private String author;
    private boolean available;

    public Book(String id, String title, String author) {
        this.id = id;
        this.title = title;
        this.author = author;
        this.available = true;
    }

    public String getId() { return id; }
    public String getTitle() { return title; }
    public String getAuthor() { return author; }
    public boolean isAvailable() { return available; }
    public void setAvailable(boolean available) { this.available =
available; }
}
```

```
class Member {
    private String id;
    private String name;
    private String email;

    public Member(String id, String name, String email) {
        this.id = id;
        this.name = name;
        this.email = email;
    }

    public String getId() { return id; }
    public String getName() { return name; }
    public String getEmail() { return email; }
}

class Loan {
    private String id;
    private String memberId;
    private String bookId;
    private LocalDate loanDate;
    private LocalDate dueDate;
    private boolean returned;

    public Loan(String id, String memberId, String bookId, LocalDate
loanDate) {
        this.id = id;
        this.memberId = memberId;
        this.bookId = bookId;
        this.loanDate = loanDate;
        this.dueDate = loanDate.plusDays(14);
        this.returned = false;
    }

    public String getId() { return id; }
    public String getMemberId() { return memberId; }
    public String getBookId() { return bookId; }
    public LocalDate getLoanDate() { return loanDate; }
    public LocalDate getDueDate() { return dueDate; }
    public boolean isReturned() { return returned; }
    public void setReturned(boolean returned) { this.returned = returned; }
```

```
}
```

Tareas

Identificación de Problemas

- En grupo, identifiquen todas las violaciones de principios SOLID en el código
- Identifiquen code smells presentes
- Documenten cada problema encontrado

Refactorización

- Refactoricen el código aplicando los principios SOLID apropiados
- Apliquen patrones de diseño donde sea necesario (Repository, Strategy, etc.)
- Separen responsabilidades claramente
- Mejoren la testabilidad del código

Preparación para Presentación

- Prepárense para explicar los cambios realizados
- Identifiquen los trade-offs de sus decisiones

SECCIÓN 2: DISEÑO DESDE CERO

Objetivo

Diseñar e implementar una solución desde cero aplicando principios SOLID y patrones de diseño apropiados.

Contexto: Sistema de Notificaciones Multi-Canal

Deben diseñar un sistema de notificaciones que pueda enviar mensajes a través de múltiples canales (Email, SMS, Push, WhatsApp) con diferentes niveles de prioridad y formatos. El sistema debe ser extensible para agregar nuevos canales fácilmente.

Requisitos

1. Canales de Notificación:
    a. Email: envía notificaciones por correo electrónico
    b. SMS: envía mensajes de texto
    c. Push: envía notificaciones push a dispositivos móviles
    d. WhatsApp: envía mensajes por WhatsApp (simular)
2. Niveles de Prioridad:
    a. LOW: puede enviarse con retraso
    b. NORMAL: enviarse en el siguiente batch
    c. HIGH: enviarse inmediatament
    d. URGENT: enviarse inmediatamente y con reintentos si falla
3. Formatos de Mensaje
    a. Texto plano
    b. HTML
    c. Markdown
4. Funcionalidades:
    a. Enviar notificación a un canal específico
    b. Enviar notificación a múltiples canales
    c. Registrar intentos de envío (éxito/fallo)
    d. Obtener estadísticas de notificaciones por canal
    e. Filtrar notificaciones por prioridad
5. Extensibilidad:
    a. Debe ser fácil agregar nuevos canales
    b. Debe ser fácil agregar nuevos formatos
    c. Debe ser fácil agregar nuevas prioridades

Restricciones de Diseño

- Deben aplicar al menos 3 principios SOLID claramente
- Deben usar al menos 2 patrones de diseño (Strategy, Factory, Observer, etc.)
- El código debe ser testeable
- Deben evitar acoplamiento fuerte entre componentes

Tareas

Diseño

- Diseñen la arquitectura del sistema
- Identifiquen las clases/interfaces principale
- Decidan qué patrones usar
- Documenten sus decisiones de diseño
- Implementación

Implementación

- Implementen el código siguiendo su diseño
- Apliquen los principios SOLID
- Implementen los patrones seleccionados
- Incluyan al menos un ejemplo de uso
- Preparación para Presentación

Preparar para presentación

- Prepárense para explicar su diseño
- Identifiquen qué principios y patrones aplicaron
- Prepárense para discutir trade-offs