DESARROLLO DE SOFTWARE

Guía 2 – Patrones de diseño e interfaces gráficas de Java.

Presentado por:

Maribel Moreno Escobar

Presentado a:

Ing. Dilsa Enith Triana Martínez

Facultad de Ingeniería

Bogotá DC, septiembre de 2024

Tabla de contenido

Introducción

En el desarrollo de software, los patrones de diseño se han convertido en herramientas fundamentales que permiten abordar problemas comunes de diseño de forma eficiente y reutilizable. Estos patrones ofrecen soluciones probadas que mejoran la calidad del código, facilitan su mantenimiento y fomentan la reutilización. En este contexto, la Programación Orientada a Objetos (POO) en Java se presenta como una metodología clave para implementar estos patrones, aprovechando características como la encapsulación, herencia y polimorfismo.

El presente informe documenta dos actividades principales realizadas durante el desarrollo de la guía 2. La primera actividad se enfoca en la comprensión y análisis de tres patrones de diseño de software, los cuales han sido representados mediante un mapa conceptual y evaluados a través de un cuestionario en línea. La segunda actividad se centra en la implementación práctica de conceptos de POO mediante el desarrollo de dos proyectos en Java, denominados "Alcancía" y "Tienda", ambos estructurados bajo el patrón Modelo-Vista-Controlador (MVC).

Como resultado de estas actividades, se logró un entendimiento más profundo de los patrones de diseño y su importancia en el desarrollo de software escalable y mantenible. Los proyectos implementados en Java demostraron la aplicabilidad de estos conceptos en situaciones prácticas, abordando desafíos como la gestión de productos con distintos porcentajes de IVA en la tienda virtual. Las conclusiones extraídas subrayan la importancia de integrar teoría y práctica para el desarrollo efectivo de software.

Objetivos

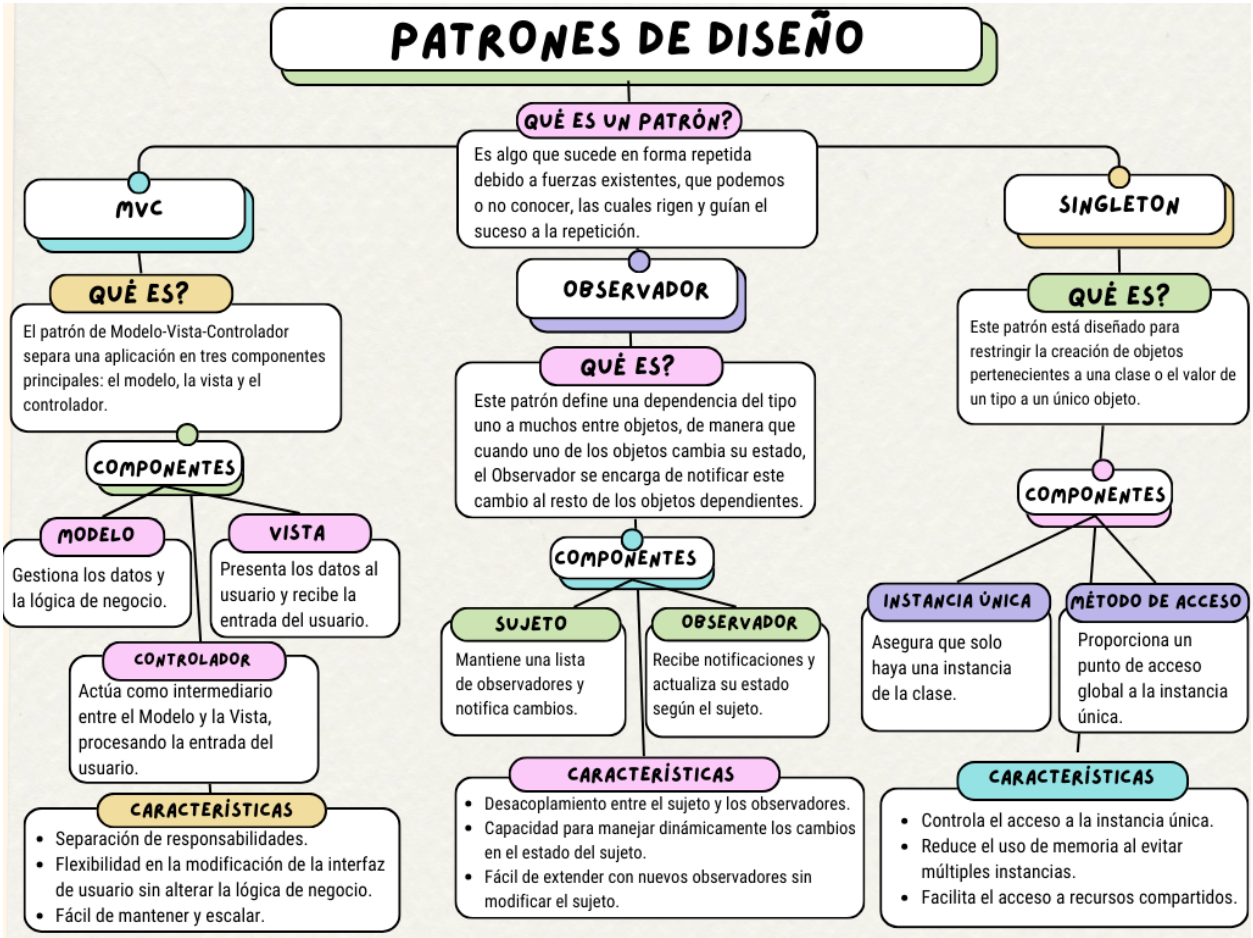
- Reconocer la utilidad que ofrecen los diversos patrones de software en la construcción de aplicaciones y logra diferenciar los diversos tipos de patrones.

- Construir las clases en Java que implementan una interfaz gráfica de usuario sencilla e integrarlas con las clases que implementan el modelo del mundo del problema a través del patrón de diseño MVC. (Modelo-Vista-Controlador)

Desarrollo de la guía 1.

Actividad 1. Patrones de diseño de Software

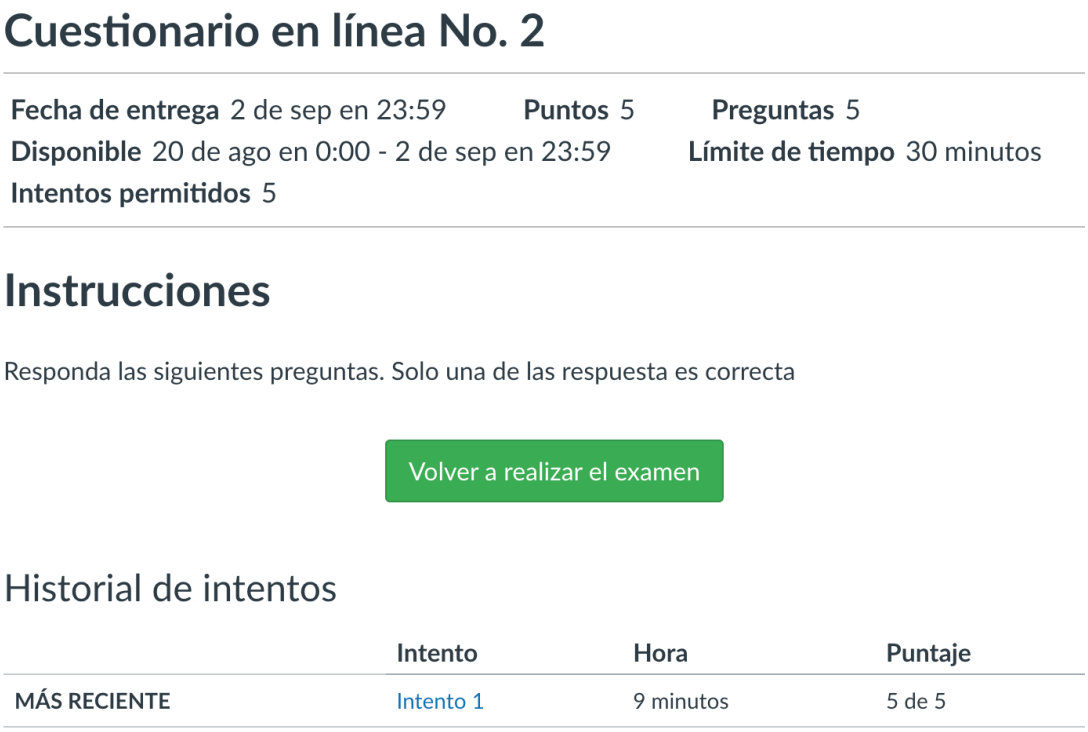
1. Mapa conceptual o mental con las características de 3 patrones de diseño de software.



Fuente: Elaboración propia.

Se adjunta link web: https://www.canva.com/design/DAGPhW-Mgbw/DFF1jekPcpW3zgRRU3Ga6Q/view?utm_content=DAGPhW-Mgbw&utm_campaign=designshare&utm_medium=link&utm_source=editor

2. Resolver el cuestionario en línea No.2.



Cuestionario en línea No. 2

Fecha de entrega 2 de sep en 23:59 **Puntos** 5 **Preguntas** 5
Disponible 20 de ago en 0:00 - 2 de sep en 23:59 **Límite de tiempo** 30 minutos
Intentos permitidos 5

Instrucciones

Responda las siguientes preguntas. Solo una de las respuesta es correcta

Volver a realizar el examen

Historial de intentos

	Intento	Hora	Puntaje
MÁS RECIENTE	Intento 1	9 minutos	5 de 5

Actividad 2. Programación Orientada por Objetos en Java

1. Proyecto “Alcancía”:

   Descripción: El proyecto "Alcancía" tiene como objetivo principal crear un programa que permita la simulación del uso de una alcancía, facilitando la administración del ahorro mediante una interfaz de usuario simple y funcional. La aplicación se divide en dos componentes principales: la lógica de la alcancía y la interfaz gráfica.

   Funcionalidades Principales:

- **Agregar Monedas:** El usuario puede añadir diferentes cantidades de monedas a la alcancía, especificando su valor.
- **Mostrar Contenido:** Permite visualizar el total de dinero ahorrado en la alcancía.
- **Interfaz Gráfica:** Implementada en Java, la interfaz permite una interacción intuitiva y amigable con el usuario.

  Tecnologías Utilizadas:

- **Lenguaje de Programación:** Java

- **Entorno de Desarrollo:** NetBeans 22
- **Patrones de Diseño:** MVC (Modelo-Vista-Controlador)

  Desafíos y Soluciones: Durante el desarrollo del proyecto, enfrenté dificultades al implementar la arquitectura MVC, especialmente en la organización de los archivos y la separación de responsabilidades. Sin embargo, tras investigar y revisar ejemplos, logré estructurar el proyecto correctamente y asegurar una correcta comunicación entre las capas.

2. Proyecto “Tienda”:

   Descripción: El "Tienda" es una aplicación diseñada para simular la operación de una tienda que maneja diferentes tipos de productos, incluyendo papelería, droguería y supermercado. El sistema permite gestionar inventarios, realizar ventas y calcular el IVA dependiendo del tipo de producto.

   Funcionalidades Principales:

- **Gestión de Productos:** Los productos se pueden crear, modificar y consultar según sus características como tipo, nombre, cantidad en bodega, y cantidad mínima requerida.
- **Cálculo de IVA:** Implementa una lógica para calcular el IVA específico para cada tipo de producto: 4% para supermercado, 12% para droguería, y 16% para papelería.
- **Registro de Ventas:** Permite registrar ventas y actualizar el inventario, incluyendo el cálculo automático del dinero en caja.

  Tecnologías Utilizadas:

- **Lenguaje de Programación:** Java
- **Entorno de Desarrollo:** NetBeans 22
- **Patrones de Diseño:** MVC (Modelo-Vista-Controlador)

  Desafíos y Soluciones: Uno de los principales desafíos fue implementar la lógica de cálculo de IVA en función del tipo de producto, y asegurar que el sistema manejara correctamente las diferentes tasas, adicionalmente, se identificó que la aplicación no estaba tomando el cambio de imagen cuando se cambiaba de producto. Estos retos se superaron mediante la creación de métodos específicos en la clase Producto, lo que permitió una correcta segregación de responsabilidades y facilitó el mantenimiento del código.

Durante el desarrollo de la actividad 2, se utilizó la práctica de dejar los comentarios "TODO" en aquellas secciones del código donde se requería implementar funcionalidades adicionales. Estos comentarios actuaron como recordatorios para identificar y completar las partes del código que requerían desarrollo posterior. Una vez revisados y entendidos los requisitos específicos de cada funcionalidad, el código correspondiente fue implementado, asegurando que se cumplieran las especificaciones solicitadas.

Se adjunta link del repositorio: https://github.com/marym9/curso-DSW-EAN/tree/7a653c8e97031a5fcb8f2b583a6336b64684951c/guia%202

Conclusión

En el desarrollo de software, la aplicación de patrones de diseño juega un papel crucial en la creación de soluciones robustas y escalables. En este informe, se ha abordado la importancia de los patrones de diseño y se ha incluido un mapa conceptual para tres patrones clave: **Modelo-Vista-Controlador (MVC)**, **Observador** y **Singleton**.

Patrón MVC: Este patrón se ha implementado en los proyectos de *Alcancía* y *Tienda*. Su capacidad para separar las responsabilidades en modelos, vistas y controladores facilita el mantenimiento y la escalabilidad del software. En el proyecto de *Alcancía*, el patrón MVC permitió una clara separación entre la lógica de negocio (Modelo), la interfaz de usuario (Vista) y la interacción del usuario (Controlador). Este enfoque promovió una mejor organización del código y simplificó la implementación de nuevas funcionalidades.

Patrón Observador: Este patrón permite una comunicación eficiente entre los componentes del software, manteniendo el desacoplamiento entre el sujeto y los observadores. Esto es especialmente útil para aplicaciones donde los cambios en un componente deben reflejarse automáticamente en otros, sin la necesidad de un acoplamiento rígido.

Patrón Singleton: El patrón Singleton asegura que una clase tenga una única instancia y proporciona un punto de acceso global a ella. Esto es crucial para gestionar recursos compartidos y evitar la creación de múltiples instancias que podrían llevar a inconsistencias.

Los proyectos de “Alcancía” y “Tienda” permitieron profundizar en el desarrollo de software utilizando Java y el patrón de diseño MVC. Además, aprender a enfrentar y resolver problemas relacionados con la organización del código y la implementación de funcionalidades específicas como el cálculo de IVA y la gestión de inventarios, lo que logró fortalecer habilidades en programación orientada a objetos y en el uso de entornos de desarrollo como NetBeans.

Para finalizar se resalta que, el uso de patrones de diseño no solo ha optimizado el proceso de desarrollo, sino que también ha proporcionado una base sólida para la construcción de aplicaciones escalables y mantenibles. La comprensión y aplicación de estos patrones son esenciales para el desarrollo de software de alta calidad y para enfrentar desafíos en el diseño y la arquitectura del software.

Referencias

Aguilar, J. M. (15 de octubre de 2019). *campus MVP.es*. Obtenido de ¿Qué es el patrón MVC en programación y por qué es útil?: https://www.campusmvp.es/recursos/post/que-es-el-patron-mvc-en-programacion-y-por-que-es-util.aspx

ASP.NET Core. (19 de marzo de 2024). *Microsoft*. Obtenido de Información general sobre ASP.NET MVC: https://learn.microsoft.com/es-es/aspnet/mvc/overview/older-versions-1/overview/asp-net-mvc-overview

Deitel, P., Deitel, H., Romero Elizondo, A. V., & Fuenlabrada Velázquez, S. (2016). ***Cómo programar en Java (10a edición).*** Capítulos 8, 9 y 10. México, México: Pearson Educación. Recuperado de https://bit.ly/39yCOVd.

Esther. (10 de septiembre de 2009). *arantxa.ii.uam.es.* Obtenido de 09 Observer: chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/http://arantxa.ii.uam.es/~eguerra/docencia/0809/09%20Observer.pdf

Junta de Andalucia. (S.F). *Junta de Andalucia*. Obtenido de Patrón Modelo Vista Controlador: https://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/122

Junta de Andalucia. (S.F). *Junta de Andalucia*. Obtenido de Observador: https://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/196

Junta de Andalucia. (S.F). *Junta de Andalucia*. Obtenido de Singleton: https://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/202#:~:text=El%20patr%C3%B3n%20de%20dise%C3%B1o%20Singleton,de%20acceso%20global%20a%20ella.

JAVARUSH. (27 de febrero de 2021). *JAVARUSH*. Obtenido de Patrones de diseño: Singleton: https://javarush.com/es/groups/posts/es.2365.patrones-de-diseo-singleton

MitoCode. (26 de mayo de 2018). *Curso de Patrones de diseño*. Obtenido de Youtube: https://www.youtube.com/watch?v=cwfuydUHZ7o&list=PLvimn1Ins-41Uiugt1WbpyFo1XT1WOquL&index=1

Refactoring Guru. (S.F). *Refactoring Guru*. Obtenido de Observer: https://refactoring.guru/es/design-patterns/observer

Refactoring Guru. (S.F). *Refactoring Guru*. Obtenido de Singleton: https://refactoring.guru/es/design-patterns/singleton

Valdecantos, H. A. (Agosto de 2010). *ResearchGate.* Obtenido de Principios y patrones de diseño de software en torno al patrón compuesto Modelo Vista Controlador para una

arquitectura de aplicaciones interactivas.:https://www.researchgate.net/profile/Hector-Valdecantos/publication/308314622_Principios_y_patrones_de_diseno_de_software_en_torno_al_patron_compuesto_Modelo_Vista_Controlador_para_una_arquitectura_de_ap

Villalobos S., J. A., & Casallas, R. (2006). *Fundamentos de programación: aprendizaje activo basado en casos: un enfoque moderno usando Java, UML, Objetos y Eclipse*. Capítulos 1, 2, 3 y 4. Bogotá, Colombia: Pearson Educación. Recuperado de https://bit.ly/36aHRc2.