PRUEBAS SOFTWARE, TIPOS, PATRONES DE DESARROLLO GOF

Jose Daniel Vanegas: 2220231063

Daniel Sebastian Naranjo: 2220231021

Juan Camilo Santamaría: 2220231066

Introducción al Desarrollo de Software

Daniel David Leal Lara

Uniagustiniana

Mayo 20 de 2023

1. **Programación Orientada a Objetos (POO):**

La Programación Orientada a Objetos (POO) es un paradigma de programación que se basa en la utilización de objetos para representar entidades del mundo real. En POO, los objetos son instancias de clases, que definen las propiedades y comportamientos comunes de un conjunto de objetos. Los principales conceptos de la POO son:

Clases: Una clase es una plantilla o molde que define la estructura y comportamiento de los objetos. Describe las propiedades (atributos) y acciones (métodos) que los objetos de esa clase pueden realizar.

Objetos: Son las instancias concretas de una clase. Representan entidades individuales y tienen un estado interno y un conjunto de comportamientos asociados.

Encapsulación: Es el principio que establece que los detalles internos de un objeto deben estar ocultos y solo se deben acceder a través de una interfaz pública. Esto proporciona modularidad y facilita el mantenimiento y la reutilización del código.

Herencia: Es el mecanismo que permite que una clase herede las propiedades y comportamientos de otra clase. Permite crear jerarquías de clases y promueve la reutilización de código.

Polimorfismo: Es la capacidad de un objeto de tomar diferentes formas y comportarse de diferentes maneras en función del contexto. Permite tratar objetos de diferentes clases de manera uniforme a través de interfaces comunes.

2. **Patrones de Desarrollo GoF:**

Los patrones de desarrollo GoF son ampliamente utilizados en el desarrollo de software para mejorar la modularidad, la flexibilidad y la reutilización del código.

Los patrones de desarrollo GoF (Gang of Four) son un conjunto de patrones de diseño de software propuestos por Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides en su libro "Design Patterns: Elements of Reusable Object-Oriented Software". Estos patrones

proporcionan soluciones probadas y prácticas para problemas comunes en el diseño de software. Los patrones de desarrollo GoF se dividen en las siguientes categorías:

Patrones de creación: Se centran en la creación de objetos de manera flexible y reutilizable. Ejemplos de patrones de creación incluyen el patrón Singleton, el patrón Factory Method y el patrón Abstract Factory.

Patrones de estructura: Se enfocan en la composición de clases y objetos para formar estructuras más grandes y flexibles. Ejemplos de patrones de estructura incluyen el patrón Adapter, el patrón Composite y el patrón Proxy.

Patrones de comportamiento: Se ocupan de la comunicación y la interacción entre objetos. Estos patrones ayudan a definir cómo los objetos colaboran entre sí para lograr un comportamiento deseado. Ejemplos de patrones de comportamiento incluyen el patrón Observer, el patrón Strategy y el patrón Template Method.

Patrones de responsabilidad: Se centran en cómo los objetos se comunican y delegan responsabilidades entre sí. Ejemplos de patrones de responsabilidad incluyen el patrón Chain of Responsibility, el patrón Command y el patrón Mediator.

3. Testing en el desarrollo de software:

El testing, o pruebas de software, es un proceso fundamental en el desarrollo de software que tiene como objetivo verificar si un sistema cumple con los requisitos establecidos y si funciona correctamente. El testing se realiza para identificar y corregir errores (bugs) y garantizar la calidad del software antes de su lanzamiento.

El proceso de testing implica la ejecución de diferentes tipos de pruebas en el software. Estas pruebas pueden ser manuales o automatizadas y se realizan en diferentes etapas del ciclo de vida del desarrollo de software. Algunos de los principales tipos de pruebas incluyen:

Pruebas de caja negra y caja blanca:

Pruebas de caja negra:

En las pruebas de caja negra, el tester evalúa el software sin tener conocimiento de su estructura interna. Se centra en probar la funcionalidad del software desde una perspectiva externa, tratando de identificar errores en la entrada y salida del sistema. Estas pruebas se basan en los requisitos y especificaciones del software y no requieren conocimiento detallado de su implementación.

Pruebas de caja blanca:

Las pruebas de caja blanca, también conocidas como pruebas estructurales, se basan en el conocimiento detallado de la estructura interna del software. El tester examina y prueba el código fuente y las estructuras de control para evaluar la cobertura y la corrección del software. Estas pruebas se enfocan en asegurar que todas las rutas lógicas y estructuras del software sean probadas exhaustivamente.

4. **Tipos de Pruebas:**

Pruebas unitarias: Las pruebas unitarias se centran en verificar el correcto funcionamiento de componentes o módulos individuales del software. Estas pruebas se realizan a nivel de código y se centran en probar funciones, métodos o clases específicas de manera aislada.

Pruebas de interfaz: Las pruebas de interfaz se enfocan en verificar la interacción y comunicación entre diferentes componentes o módulos del software. Estas pruebas se realizan para garantizar que los diferentes componentes se integren correctamente y que los datos se transfieran y procesen adecuadamente entre ellos.

Pruebas de componentes: Las pruebas de componentes se centran en probar la funcionalidad de componentes o módulos completos del software. Estas pruebas evalúan el comportamiento y el rendimiento de componentes individuales para asegurar su correcto funcionamiento dentro del sistema global.