Manual técnico

Patrones de diseño

Un patrón de diseño de software es una solución general y reutilizable para un problema común que ocurre durante el diseño y desarrollo de software. Estos patrones ofrecen un enfoque probado y eficaz para resolver problemas recurrentes en el desarrollo de software y permiten a los desarrolladores comunicarse de manera efectiva y compartir experiencias y soluciones.

Los patrones de diseño de software se derivan de las mejores prácticas observadas en proyectos exitosos de desarrollo de software y se han documentado para que otros puedan aplicarlas de manera efectiva. Los patrones no son códigos o soluciones listas para usar, sino más bien descripciones y guías que ayudan a los desarrolladores a resolver problemas específicos de diseño en sus proyectos.

Estos patrones tienen nombres descriptivos y ayudan a abordar diferentes aspectos del diseño de software, como la estructura de clases, la comunicación entre objetos, la gestión de comportamientos y la gestión de datos, entre otros.

Estos patrones ayudan a mejorar la flexibilidad, la reusabilidad y la eficiencia del código, al tiempo que facilitan la comprensión y colaboración entre los miembros del equipo de desarrollo. Es fundamental comprender cuándo y cómo aplicar cada patrón de manera apropiada para obtener los mejores resultados en el desarrollo de software.

Patrones utilizados

Singleton

El patrón Singleton es un patrón de diseño creacional que garantiza que una clase tenga una única instancia y proporciona un punto de acceso global a esa instancia en toda la aplicación. En otras palabras, solo puede haber una única instancia de una clase determinada y se proporciona un mecanismo para acceder a esa instancia única desde cualquier parte del código.

Este patrón es útil cuando se desea tener control sobre la creación de una única instancia de una clase específica y asegurarse de que no se creen múltiples instancias idénticas. Algunos casos de uso comunes incluyen la gestión de conexiones a bases de datos, logs, configuraciones, caches y objetos que representan recursos únicos.

En este caso se utilizó para llevar control de 2 variables globales útiles para el programa, una para llevar control de los archivos JSON que se encuentran cargados y otra para llevar control de el archivo StatPy cargado

Interprete

El patrón del Intérprete (Interpreter Pattern en inglés) es un patrón de diseño de software que se clasifica dentro de la categoría de patrones de diseño comportamentales. Este patrón se utiliza para

definir una gramática para un lenguaje y proporcionar un intérprete para interpretar y ejecutar las expresiones escritas en ese lenguaje.

El patrón del Intérprete es útil cuando se tiene un problema que puede ser resuelto mediante la interpretación de un lenguaje o una gramática específica. Este patrón se utiliza en situaciones en las que se necesita representar y ejecutar expresiones complejas o reglas de negocios personalizadas, como un lenguaje de programación, un sistema de consulta de bases de datos o un motor de reglas.

En este caso se utilizó para interpretar la generación de gráficas de una archivo StatPy en el programa y para la traducción general del lenguaje statPy a Python. Facilitando grandemente la traducción e interpretación. Pues al tener clases de estructuras de StatPy se puede traducir una por una.

MVC (modelo-vista-controlador)

El patrón Modelo-Vista-Controlador (MVC, por sus siglas en inglés: Model-View-Controller) es un patrón arquitectónico ampliamente utilizado en el desarrollo de aplicaciones de software. Este patrón tiene como objetivo separar la representación de la información (el modelo), la interfaz de usuario (la vista) y la lógica de control (el controlador) en tres componentes distintos e interconectados.

Se utilizó en el control de lógica del programa en general. Se tenía el **modelo** el cual eran los archivos cargados en el sistema, la **vista** que era solamente la interfaz hecha con las herramientas del IDE, y el **controlador** que se encargaba de unir estos componentes.

Análisis de StatPy

El analizador de statPy se llevó a cabo con la herramienta de Jflex y cup. Jflex se encargó de la declaración de Tokens aceptados en el programa, y como guardarlos. Mientras que cup se encargó de llevar control de la gramática y como deberían de ser utlizados los no terminales.

Al utilizar el patrón del intérprete fue sencillo integrar el cup en el programa de traducción y análisis. Pues el cup creaba una lista de instrucciones que contenían estructuras, y así se llevaba a cabo como debería ser la traducción dentro de estas estructuras.

Si se ingresaba a una de las funciones estadísticas, estas no se traducen, pero sí se interpretan, por lo que la asignación de variables se llevaba a cabo en una tabla de símbolos. Para recuperar el valor luego y crear una gráfica.

Por motivos de simplicidad se dividieron las instrucciones de un archivo StatPy en 4 partes.

Expresiones

Básicamente todo lo que pueda devolver un valor, como llamar a una función, asignar a una variable, los tipos primitivos, etc

Gráficas

Las funciones estadísticas que sí se pueden interpretar

Sentencias

Como la asignación de variables, la reasignación de variables, imprimir, un bloque de código etc.

Estructuras

Estructuras como el Main, una estructura if, while, for, entre otras.

Análisis de JSON

De forma similar que el de StatPy, sin embargo más sencillo, pues no toma en cuenta muchas estructuras nativas de un archivo JSON. De igual forma se uitliza el patrón del intérprete para guardar la información de forma más sencilla en otro objeto.