PROYECTO No. 2

Carnet 201730511 – Edy Rolando Rojas González

Resumen

La comprensión de los tipos de datos abstractos proporcionan grandes ventajas en el desarrollo de software, tales como enfocarse en las partes más importantes y obviar los detalles, enfocándonos en en la interfaz y no en la implementación.
Los TAD están directamente relacionadas con la Programación Orientada a Objetos, paradigma de programación altamente utilizado en la industria debido a que proporciona grandes ventajas como la reutilización de código, legibilidad, mantenimiento y por ende sistemas más robustos en el tiempo.
Un lenguaje como Python aporta las herramientas necesarias otorgando un aprendizaje fácil del lenguaje debido a una sintaxis clara y legible, también es necesario mencionar que sus que tiene una amplia gama de Librerías como las que brindan el manejo de archivos XML y Graphviz que ayuda a crear y renderizar grafos.
La asociación de estos conceptos y la utilización de este lenguaje de programación brinda al estudiante las herramientas necesarias para ser una gran oferta ante la industria de la tecnología.

Palabras clave

Tipo de dato abstracto, Listas enlazadas en varios sentidos, Archivos XML, Programación Orientada a Objetos, Graphviz

Abstract

The understanding of abstract data types provides great advantages in software development, such as focusing on the most important parts and avoiding the details, focusing on the interface and not on the implementation.
TADs are directly related to Object Oriented Programming, a programming paradigm highly used in the industry because it provides great advantages such as code reuse, readability, maintainability and therefore more robust systems over time.
A language such as Python provides the necessary tools for easy learning of the language due to a clear and readable syntax, it is also necessary to mention that it has a wide range of libraries such as those that provide the management of XML files and Graphviz that helps to create and render graphs. The association of these concepts and the use of this programming language gives the student the necessary tools to be a great offer to the technology industry.

Keywords

Abstract data type, Lists linked in several directions, XML Files, Object Oriented Programming, Graphviz

Introducción

En base a los requerimientos, restricciones y objetivos planteados en el proyecto 2, se desarrolló una solución utilizando el paradigma de programación Orientada a Objetos, el programa desarrollado se divide en 5 módulos principales que abstraen lo más importante del problema, el manejo de archivos, manejo de archivos XML y TDA específicos para este problema. También se utilizó una librería para facilitar la creación del grafo en Graphviz, aprovechando la versatilidad del lenguaje de Python.

Desarrollo del tema

En el análisis inicial del problema se optó por desarrollar utilizando el patrón de MVC sobre la premisa que se manejan distintos controladores y servicios, y dos modelos principales que atribuyen sus bases en un TDA orientada a las listas enlazadas.

Se presentan los siguientes requerimientos dictaminados por el enunciado del Proyecto 2:

- Utilizar POO.
- Desarrollar las propias estructuras de almacenamiento.
- Visualizar las "Maquetas" a través de Graphviz.
- El programa debe de ser capaz de manipular archivos XML

Para poder aplicar un uso adecuado de POO se dividió el desarrollo en 5 módulos principales que se describen a continuación.

- ADT, contiene las propias estructuras de almacenamiento y la clase Nodo, el Nodo está dispuesto de tal manera que contempla 4 referencias; arriba, abajo, siguiente y previo con el fin de poder usarlo para recorrer la maqueta y hallar los objetivos en el orden estipulado de acuerdo a la descrito en el enunciado.
- Controllers, almacena los controladores de las entidades/modelos.
- Models, las plantillas u objetos que se abstrayeron.
- GUI, este módulo tiene una única clase que se encarga de mostrar las vistas al usuario

Se desarrollaron dos tipos de listas, una clásica lista enlazada con sus principales implementaciones tales como agregar al último e imprimir la lista. El otro tipo de lista un poco más compleja se definió como lista Matricial enlazada en la que las implementaciones fueron un poco más complejas debido a distintas consideraciones que se fueron tomando en cuando se desarrollaba como por ejemplo poder enlazar los nodos con su anterior su siguiente, su superior y el que se encuentra debajo, como también tomar en cuenta que existe una cantidad de filas y columnas que ya se encuentran definidas por medio de la maqueta.

Para el procesamiento de los archivos XML y la creación de las entidades abstraídas se inició considerando que dentro del archivo de entrada, podrían haber más de una maqueta por lo que se consideró que una maqueta debe almacenar información como nombre, filas, columnas, objetivos entre otros. El manejar la clase "Maqueta" de esta manera permite la implementación de ordenar la lista de manera ordenada y mostrarla al usuario.

Entendiendo que es sumamente importante abstraer objetos para poder realizar comparaciones y validaciones nos encamina a pensar que es

importante identificar el tipo de casilla y su representación por lo cual se abstrayeron 4 tipos de casilla que heredarán de una clase abstracta, al utilizarse el paradigma POO y analizando la abstracción realizada se pudo observar que partiendo de la idea que hay casillas especiales que se colocan en partes específicas de la matriz estas casillas pueden considerarse como indexadas resultando en una clase abstracta de la cual también heredar.

La anterior abstracción también presenta varias ventajas como poder indicarle a graphviz la representación gráfica de cada casilla y mostrar la maqueta de tal manera que sea fácil identificar cuáles son “caminos”, “paredes”, “objetivos” y la “entrada”. También es poderosamente útil para el apartado del controlador que se encargará de resolver la maqueta y recolectar los objetivos porque abre la puerta a que las validaciones sean más sencillas a la hora de moverse y verificar si son casillas “Objetivos” sumado a que los nodos guardan 4 referencias haciendo referencia a 4 posibles movimientos facilita aún más moverse por la maqueta y recolectar todos los objetivos en el orden específico.

Conclusiones

La relevancia de la programación orientada a objetos sigue y será por bastante tiempo siendo unos de los paradigmas más importantes en el arte de programar y también en la forma de pensar, entregándonos sistemas de software robusto de largo mantenimiento y sobre todo de reutilizacion de codigo, conjuntamente con los TDA nos otorgan formas y maneras de resolver problemas enfocándonos en las partes más importante del sistemas y/o problemas al que nos enfrentando evitando estrellarse en los detalles que son irrelevantes para el contexto que analizamos.

El utilizar un lenguaje como Python que al tener una sintaxis muy pulida y entendible, facilita el trabajo de programar, sin embargo al ser un lenguaje de tipado dinámico presenta en ciertas ocasiones algunas dificultades al inferir comportamientos de ciertos objetos que quizá en el contexto que se utilizan no son permitidos o no se comportan como esperamos entrando en conflicto y presentados el trabajo de analizar a detalle qué estamos haciendo con las instancias que declaramos.

Extensión:

Diagrama de Clases

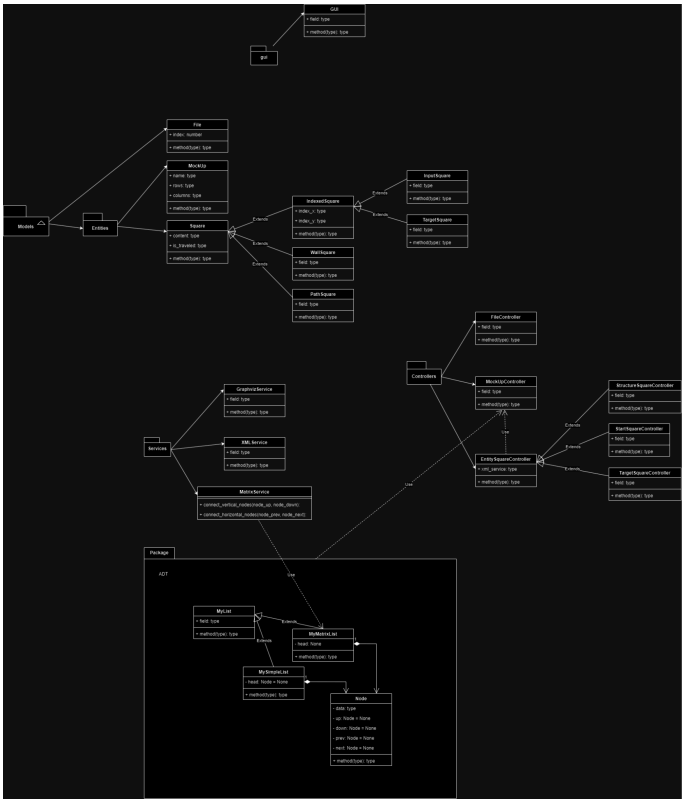

Imagen 1. Diagrama de Clases

Fuente: elaboración propia a través de Draw.io

Diagramas de Flujo

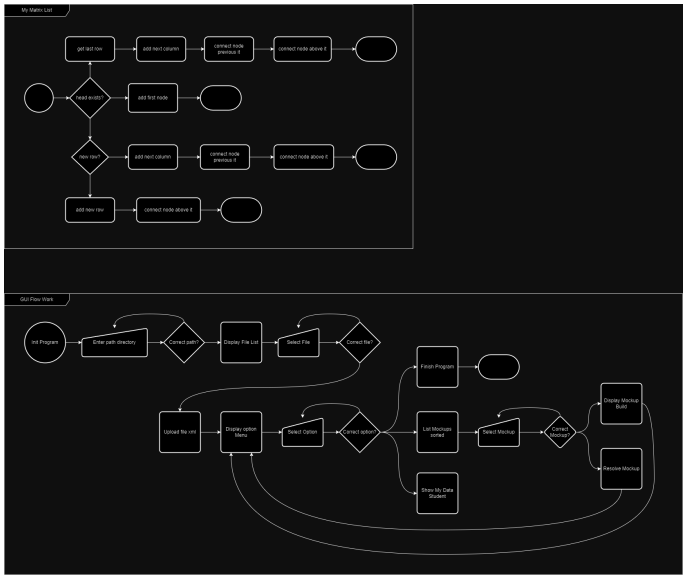

Imagen 2. Diagramas de flujo

Fuente: elaboración propia a través de Draw.io

Diagrama de Clases V2.0.0

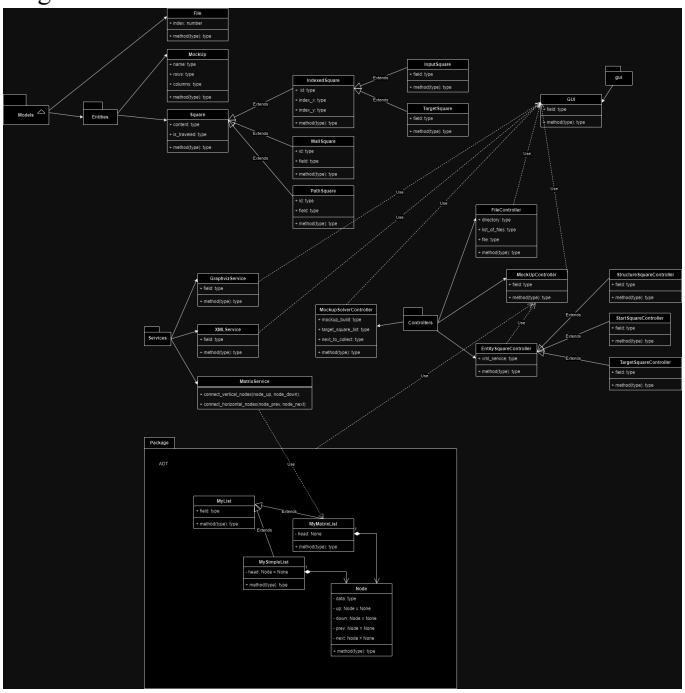

Imagen 3. Diagramas de Clases

Fuente: elaboración propia a través de Draw.io