Técnicas de Programación

Realizado por: Pedro Alonso Tapia Lobo

EL PROBLEMA DE PARADA- ALAN TURING

Realizado por: Pedro Alonso Tapia Lobo

1- Explicación de la implementación:

He desarrollado una aplicación que simula un sistema para analizar programas y verificar si un código se detendrá o entrará en un bucle infinito. Utilizando el patrón de diseño **Chain of Responsibility**, se implementaron múltiples manejadores (handlers) que verifican patrones como bucles *while(true)* o bucles for sin condiciones de salida claras.

2- Presentación de las clases:

CountUpHandler: Verifica si un bloque de código contiene un bucle infinito al contar hacia arriba

CountDownHandler: Verifica si un bloque de código cuenta hacia abajo hasta detenerser

ForLoopHandler: Detecta bucles for potencialmente infinitos

InfiniteLoopHandler: Detecta bucles while(true) que resultan en bucles infinitos

HaltChecker: Gestiona una cadena de handlers para determinar sin un bloque de código se detiene o no

ReverserController, ReverserModel y ReverserView: Implementan un patrón MVC para el análisis de código y la interacción con la GUI.

ReverserLogic: Lógica para el análisis del comportamiento del programa Reverser

CountUp: Programa que cuenta indefinidamente hacia arriba

CountDown: Programa que cuenta indefinidamente hacia abajo y se detiene

3- Patrones de diseños utilizados y por qué:

Chain of Responsibility: Permite que cada handler analice una solicitud y decidad si manejarla o pasarla al siguiente de la cadena, permitiendo agregar nuevas reglas de análisis sin modificar la estructura siguiente

Modelo Vista-Controlador: Separa la presentación, lógica de control y datos del modelo, haciendo que la app sea modular y fácil de entender.

4- Entendimiento y desarrollo de la práctica:

La práctica se entendió como un ejercicio para simular el problema de parada y aplicar patrones de diseño que permitan analizar el código de manera eficiente. Se

identificaron los tipos de bucles finitos e infinitos. Se desarrollaron los handlers para cada tipo y finalmente, se implementó la cadena de responsabilidad que gestiona el análisis.

5- Reflexión sobre los límites de la Computación

El ejercicio demuestra los límites de la computación tal como lo descubrió Turing, puesto que ningún algoritmo puede determinar infaliblemente si un programa arbitrario se detendrá o no. En la práctica, se puede observar la importancia de los patrones de diseño para abordar problemas complejos y como crear simulaciones con técnicas modernas.