Comparing Dynamic and Static Type Systems in C#

Introduction

The type systems are some of the most important factors in a programming language's syntax system, as it is a logical structure that describes how the data can be classified and manipulated by developers. Type systems can be implemented in most programming languages, including C#. Speaking of which, C# is a general-purpose programming language that provide support for both static and dynamic type systems. There are a few differences between the dynamic and static type systems. For example, dynamic type systems will run type checks when the program is being executed. Meanwhile, static type systems will run type checks when the program is being compiled. This research report aims to understand the traits of each type systems and their advantages and disadvantages.

Dynamic Type Systems

This system also has its own unique technique called duck typing, where programmers only need to check for the existence of a method or attribute without worrying about data types. Dynamic type systems will not perform type checking until the program is being executed. By doing this, dynamic type systems provide flexibility and adaptability for programmers to make changes to certain data and objects during the compile time. However, this will lead to errors remain undetected until the program's execution which may cause the program to crash. Dynamic type systems can be more difficult to debug with, as type checks at compile time is not applied. Furthermore, it is also a bit harder to read due to absence of type declarations.

Static Type Systems

Unlike the dynamic type systems, static type systems will require the type constraints to be implemented during compile time. This guarantees that there are specific data types and methods that are compatible with each other before execution time. As the result, static type systems will allow the developer to recognise bugs sooner before the program begins to run, reducing the chances of the program to crash. It also makes the code easier to read due to the type declarations. On the other hand, static type systems will be stricter on data type and constraints and have the programmers to write more code because of the enforced requirements of type annotations.

Method

It is decided that we will write two different programs that utilises two different programming languages that have different type systems in each language. The purpose of these programs is to compare the ease of implementation, code complexity, and the amount of time spent for debugging. C# will be used to test static typed systems and Python will be used to test dynamic typed systems. There will be two simple functions. Function example1 will do a string concatenation, while function example2 will have 2 test cases: one where

both data are compatible to be added into a sum, and one where both data are incompatible. Additionally, function example2 will be timed to test its run time speed.

Relevancy to current project

This research topic is relevant to my current custom project BankATM because I use the static type systems for building an Object-Oriented Programming console application. The research I have done in this report can help me understand the fundamental concepts of both dynamic and static system types. I can also use the timing testing method to see how optimised my code is and make adjustments if needed.