Known Languages

Armine Hayrapetyan

Outline

- Static vs Dynamic Typing
- Strong vs Weak Typing
- Inferred Types
- Static Member
- Static Function

Static vs Dynamic Typing

Dynamically-typed languages perform type checking at runtime, while statically typed languages perform type checking at compile time. This means that scripts written in dynamically-typed languages (like Groovy) can compile even if they contain errors that will prevent the script from running properly (if at all). If a script written in a statically-typed language (such as Java) contains errors, it will fail to compile until the errors have been fixed.

Second, statically-typed languages require you to declare the data types of your variables before you use them, while dynamically-typed languages do not. Consider the two following code examples:

```
// Java example
int num;
num = 5;
```

```
// Groovy example
num = 5
```

Strong

add(5, '7') --> ?

concatenate('5', 7) --> ?

Throws Error

Explicit Conversion of DataType Required

add(5, **converToInt**('7')) -->12

concatenate('5', **converToString**(7)) --> 57

Weak

add(5, '7') --> ?

concatenate('5', 7) --> ?

NO Error

Implicit DataType Conversion Done by Compiler

add(5, '7') --> 12

concatenate('5', 7) --> 57

Inferred Type

Inferred type = set ONCE and at compile time. Actually the inferred part is only a time saver in that you don't have to type the Typename IF the compiler can figure it out.

Dynamic type = no fixed Type -> type can change at runtime

```
var i = true; //compiler can infer that i most be of type Bool
i = "asdasdad" //invalid because compiler already inferred i is an Bool!
```

```
var i: bool = true; //You say i is of type Bool
i = "asdasdad" //invalid because compiler already knows i is a Bool!
```

Static Member

When we declare a member of a class as static it means no matter how many objects of the class are created, there is only one copy of the static member. A static member is shared by all objects of the class.

Static Function

By declaring a function member as static, you make it independent of any particular object of the class. A static member function can be called even if no objects of the class exist and the static functions are accessed using only the class name.

A static member function can only access static data member, other static member functions and any other functions from outside the class.

Static member functions have a class scope and they do not have access to the this pointer of the class. You could use a static member function to determine whether some objects of the class have been created or not.

References

- https://www.tutorialspoint.com/cplusplus/cpp_static_members.htm#:~:text=When%20we%20declare%20a%20member,no%20other%20initialization%20is%20present.
- https://en.hexlet.io/courses/intro_to_programming/lessons/types/theory_unit#:~:text=Strong%20versus%20weak%20is%20about,JavaScript%20has%20very%20weak%20typing.
- https://stackoverflow.com/questions/2690544/what-is-the-difference-between-a-strongly-typed-language-and-a-statically-typed