Comparative Analysis of Multi Paradigms Languages

Faisal Mumtaz
i16-1024
MS(CS)

Faisal Mumtaz
i16-1024
MS(CS)

Umar Munir
i16-1024
MS(CS)

Shahran Gohar
i16-1024
MS(CS)

Mehreen Alam
i16-1024
MS(CS)

Shahid Hussain
i16-1024
MS(CS)

ABSTRACT

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

Theory

Keywords

ACM proceedings, LaTeX, text tagging

1. INTRODUCTION

2. RELATED WORK

3. FEATURES

3.1 Bound Checking

In computer programming, bound checking is any method of whether variable detecting variable is within bound before it is used. A failed bounds check usually results in the generation of some sort of exception signal.

3.1.1 Range checking

It is usually used to check that whether a number fits into a given type. A range check is a check to make sure a number is within a certain range; for example, range check will ensure that a value that will assign to a 16-bit integer is within the capacity of a 16-bit integer. Some range checks may be more restrictive; for example, a variable to hold the number of a calendar month may be declared to accept only the range 1 to 12

Table 1: My caption

	Index checking	Range checking
Scala	✔	✔ (statically check)
Swift	✔	✔
F#	✔	✔
Rust	✔ (at run time)	-
Vb.net	✔	✔
C#	✔	✔
D	✖	✔
Oz	-	-
Matlab	✔	✔(statically check)
Python	✔	✔

3.1.2 Index checking

In index checking a variable being used as an array index is within the bounds of the array. Index checking means all expressions indexing an array, the index value is checked against the bounds of the array, which were created when the array was defined, and if the index is out-of-bounds, an error occur and further execution is suspended. If a number outside of the upper range is used in an array, it may cause the program to crash, or may introduce security vulnerabilities, index checking is a part of many high-level languages.

3.1.3 Examples

3.1.4 Scala

Array representation in scala
scala> val a1 = Array(1, 2, 3)
a1: Array[Int] = Array(1, 2, 3)

3.1.5 Swift-range checking

func contains(Bound) Returns a Boolean value indicating whether the given element is contained within the range.

3.2 Type Safety

The compiler will validate types and through an error if you assign a wrong type to a variable. Type safety is checking for matched data types during compile time. For example, int a ="John" returns error as variable 'a' is an integer and we are assigning a string value. These data type mismatches are checked during compile time. Type safe code

can access only the memory locations that it has permission to execute. Type safe code can never access any private members of an object. Type safe code ensures that objects are isolated from each other and are therefore safe for inadvertent or malicious corruption

3.2.1 The advantages type safety

At compile time, we get an error when a type instance is being assigned to an incompatible type; hence preventing an error at runtime. So at compilation time itself, developers come to know such errors and code will be modified to correct the mistake. So developers get more confidence in their code. Run time type safety ensures, we don't get strange memory exceptions and inconsistent behavior in the application.

3.2.2 scala

Scala is strongly type and smart about static type. Scala has powerful type inference. It will figure out itself mostly no need to tell it the types of your variables.

3.2.3 Swift

Swift is type safe, it performs type checks when compiling code and flags any mismatched types as errors. This help in early catch and fix error in the development process. It provides type inference which basically means that coders donâĂŹt require to spend more time in defining what types of variables they are using.

3.2.4 F#

In f#, static type checking can use almost as an instant unit test âĂŞ making sure that your code is correct at compile time. F# is more type-safe than C#, and how the F# compiler can catch errors that would only be detected at runtime in C#.

3.2.5 Rust

Rust is a type-safe language. Rust has an escape valve from the safety rules. When you absolutely have to use a raw pointer. This is called unsafe code, and while most Rust programs dont need it, how to use it and how it fits into Rustś overall safety scheme in https://www.safaribooksonline.com/library/view/programming-rust/9781491927274/ch21.html#unsafe-code

3.2.6 VB.net

Type safety in .NET has been introduced to prevent the objects of one type from peeking into the memory assigned for the other object.

3.2.7 C#

Type safety prevents assigning a type to another type when are not compatible. public class Employee

public class Student In the above example, Employee and Student are two incompatible types. We cannot assign an object of employee class to Student class variable. If you try doing so, you will get an error during the compilation process. Type safety check happens at compile time it's called static type checking Example Cannot implicitly convert type 'Program.Employee' to 'Program.Student'. . When tried to type cast object of wrong type. We get Unable to cast object of type âĂŸfirst objectâĂŹ to âĂŸsecond objectâĂŹ type checking happens at runtime, hence it is called runtime type checking

Table 2: My caption

Languages	Type Safety
Scala	Strongly type, Static type, powerful type Inference
Swift	Type check at compile time, Support type inference
F#	Static Type Checking, Compile Time
Rust	Type Safe, escape valve, unsafe to use raw pointers
VB.net	Type safety use for memory security
C#	Static type checking, type checking compile time
D	Type safe, compile time
Oz	Single Assignment variables, Once value is assigned to variable it can never be change
Matlab	Weakly type language, no need to assign type explicitly,
Python	Dynamically type language,

3.2.8 D

D has compile-time type safety.

3.2.9 OZ

OZ also known as MOZART. Oz variables are single-assignment variables or more appropriately logic variables. A single assignment variable has a number of phases in its lifetime. Initially it is introduced with unknown value, and later it might be assigned a value, in which case the variable becomes bound. Once a variable is bound, it cannot itself be changed.

3.2.10 Matlab

MATLAB is a loosely or weakly-typed language. Difference between MATLAB and a strongly-typed language is that you don't have to explicitly declare the types of the variables you use. For example, the declarations x=5; x='foo' immediately following one another are perfectly acceptable; the first declaration causes x to be treated as a number, the second changes its treatment to a string

3.2.11 Python

Python or Ruby are often referred to as dynamically typed languages, which throw exceptions to signal type errors occurring during execution

3.3 Exception Handling Umar

3.4 Nodularity Umar

3.5 compiled / interpreted Maam Mehreen

3.6 Assertion Maam Mehreen

3.7 conditional compilation Shaad