Language Evaluation for CS4700

DJ Wiley

I. INTRODUCTION

For my language evaluation assignment I decided to analyze C#. At the beginning of the semester Dr. Sundberg went on one of his many rants about the shortcomings of Java, in the process of the rant he mentioned that C# was much better than Java in his opinion. This semester I am taking the Object Oriented Programming class and we had the opportunity to do our assignments in either Java, or C#. Prior to this class I had no experience in C# and decided to give it a try mostly based on Dr. Sundbergs honorable mention. The following is my evaluation of C# and I decided to evaluate it based on some of the main points that we have covered in this course.

II. HISTORY

The programming language C# is a popular programming language built on the .NET framework and was developed by Microsoft. Anders Hejlsberg is often referred to as the father of C# and led the team that created this language. It was initially called the COOL language. The team seriously considered keeping the acronym COOL, it stood for "C-like Object Oriented Language". In the end they werent able to keep the name due to trademark conflicts. They chose C# instead as a play on words. In music a sharp sign (#) raises a note by a half a step. So by calling it C# they were saying that the language should be seen as a step above C, maybe only half a step.

III. IDENTIFIERS AND BINDINGS

A. Identifiers

In C# an identifier must begin with a letter or an underscore. Most of the time it is not allowed to have the same name as a keyword. If a user wants to use an identifier with the same name as a keyword they must prefix the identifier with the @ symbol. Using this is frowned upon though because it impacts the readability of the code. In C# identifiers are case sensitive. There isnt a rule that enforces the proper casing for the language, but there is the common convention that parameters, local variables, and private fields should be camel case (fooBar) and all other identifiers should use Pascal case (FooBar).

B. Bindings

C# is a statically typed language for the most part, dynamic binding is also available. The big difference between static binding and dynamic binding is the time that the binding happens. Static binding happens at compile time, and dynamic binding happens at runtime.

IV. DATA TYPES

C# has a variety of data types. This includes value types, pointer types, reference types, and generic type parameters. Value types seems kind of broad, but that includes all numeric types (int, float, etc.), char type, bool type, and struct and enum types. Pointers in C# are interesting, users can only utilize them by using the unsafe keyword. This is so they are not tracked by the default garbage collecting mechanism.

V. EXPRESSIONS

Expressions in C# are similar to expressions in mathematics. A list of these includes primary expressions, void expressions, and expression statements. Operators are also significant in C#. Many of the operators can be overloaded. Each operator fits into a particular category, some of these categories include primary, unary, multiplicative, additive, logical, and conditional.

VI. CONTROL STRUCTURES

The purpose of control structures are to direct the flow of the program. In C# there are three main types of control structures. These include: selection, iteration, and unconditional branching. Unconditional branching refers to jump statements like break, continue, return, throw, and goto. Most control structures in C# must be specified in parenthesis and also be of a Boolean type.

A. Selection Statements

Selection statements include the common if-else statement and also the switch statement. To enhance a programs readability, complex if-else statements should be replaced with switch statements.

B. Iteration

A common type of iteration are loops. Like all C based languages, C# loops can be exited with jump statements. C# has four types of loops, they are while loops, do-while loops, for loops, and foreach loops. Bot the while and do-while loops use a Boolean condition to exit the loop. For loops loop a specific number of times and then exit. They adopt the same format as other for loops in C based languages. They have an initializer, the condition (number of times to iterate), and the iterator (often times just incrementing a value).

VII. SUBPROGRAMS

C# refers to subprograms as methods which can be of any type including void.

A. Parameter Passing

C# passes parameters to a method in a few different ways. The default is a pass-by-value. Other ways include pass-by-result, pass-by-value-result, and pass-by-reference.

B. Type Checking

Like most languages, type checking is a very important parameter in regards to reliability. Without type checking being implemented in a language, the small typo errors can lead to hard to detect problems. Mixing up a float and a double can cause huge errors that are hard to pinpoint. C# requires the reference type of the parameter to be identical to the the corresponding parameter in order to prevent this.

C. Nested Methods

C# used to not allow nested methods, now it sort of does. Most often they are used for closures. The reason why nested methods are sometimes not allowed in languages is because they can be hard to trace and therefore impact the readability and testability of code.

D. Methods as Parameters

When I think of passing methods as a parameter, I think of C++ where that is handled by passing a pointer to the method. In C# since almost everything is an object, the object itself is passed. In C# they are referred to as delegates. The program calls the delegate, which then calls the sought after method providing a layer of abstraction.

E. Overloading Methods

Overloading a method is when a language allows a user to use the same name as a specified method (like print) and then change them to another specified method. C# does allow this, and users should be wary when overloading methods.

VIII. CONCURRENCY

The concurrency support in C# seems very similar to how Java implements concurrency, but it seems like C# has done it better in my opinion. The reason why I think that is because C# has more thread capabilities than Java.

IX. EXCEPTION AND EVENT HANDLING

C# has a built in system class for exceptions (System.Exception). Similar to most mainstream languages, the exception handling in C# utilizes try, catch, and finally blocks. This is a good thing when thinking about performance, using a System.Exception class can take much longer than a simple try catch block.

Event handling in C# is done almost identically to that of Java. Both languages are popular when developing GUI's and both languages utilize an EventHandler object with the purpose of raising an event notification when something happens.

X. SUMMARY OF EVALUATION

This is my overall summary of the C# language. I apologize if my opinion clashes with whoever may be reading this.

A. Reliability

I would say that C# overall is a pretty reliable language. What makes me come to this conclusion are a couple of things. For one it has very strict type-checking. Also implicit casting is only utilized when precision won't be sacrificed. All other variables that can make the code unreliable and unsafe are strongly related to how a user programs in C#. Improper or sloppy use of semaphores and aliases can make the code much less reliable.

B. Readability

In my opinion C# is fairly readable, I would give it a seven or eight out of ten. The system variable types and methods all have descriptive names and use the common terms like int or bool. If a user has some experience in an object-oriented programming language, they should be able to go through someones code and have a pretty good idea of what is going to happen.

The short comings in my opinion stem from common features of most object-oriented languages. Having everything be an object or a class can make it hard to follow some code and can negatively impact readability.

C. Writability

C# has many different ways to implement a certain task. This can be very helpful for the writability, you can take a stab in the dark at what a library might call a commonplace function and chances are you are pretty close. However, C# is very particular about syntax, this can be annoying and hard to master for beginners. Overall i would say that the writability in C# is average with other mainstream languages.

XI. CONCLUSION

Overall I like C# a lot more than Java. It feels a little more natural to code in, that could just be because of the C++ background that I started with at USU. I would highly recommend using C# for desktop applications, GUI's, and cross platform app development. There are many other uses, but that is what I like to use C# for. I know C# is also popular for game development.

REFERENCES

[1] Hejlsberg, Anders. *The A-Z of Programming Languages: C#*, Naomi Hamilton. 1 October 2008.