Programming Languages Course

Homework – 1

Comparing C and Java

Bahadir Etka Kilinc - 1901042701

Historical Development of Programming Languages

Programming languages have been developed by influencing each other. This influence generally takes the form of quoting a language and adding new features to it. If programming languages are thought of as a tree, again, Fortran is at the top. (1954-1957). Fortran is the world's first high-level programming language. Fortran was followed by the Algol and Cobol languages. The C programming language was designed by Dennis Ritchie at Bell Laboratories in 1970. The Java programming language was designed by James Gosling of Sun.

Classification of Programming Languages

We can classify programming languages in three ways:

- Classification by level
- Classification according to usage areas
- Classification according to the programming model

Classification by Level

Level is a measure of the closeness of a programming language to human perception. High-level languages are languages that are close to human perception, which are easy to learn both in terms of language features and application development. Low-level languages are closer to the natural operation of the machine. The most low-level language possible is pure machine language.

Classification According to Usage Areas

General Purpose languages, Scientific and engineering languages, Game and animation languages, Artificial intelligence languages, Web languages, System Programming languages etc.

Classification by Programming Model

The general model (paradigm) used when writing a program is important. Some languages do not have classes. The program is written by various subprograms (functions) calling each other. This technique is called the procedural model. Some languages have classes. The program is written using classes. This model is called the object-oriented model. In some languages, programs are written as if they were writing a mathematical formula. These languages are called functional languages, and the programming model is called the functional programming model. Some languages support more than one model. These are called multiparadigm languages.

Translator Program, Compiler and Interpreter

Programs that convert a program written in one programming language to another language equivalently are called translator programs. In converter programs, there are source language and target language. For example, it is a program converter program that converts the Fortran programming language to the C programming language. Here, the source language is Fortran, and the target language is C. If the target language in a translator program is a low-level language (pure machine language, symbolic machine language, or bytecode), such translator programs are specifically called compilers. The JVM's activity of converting bytecode to machine code is a compilation. This process is done by the JIT compiler (Just in Time compiler) in the running environment.

Some programs run directly without generating any code. Programs that run them are called interpreters. Some languages only work with compilers, some languages only work with interpreters, and some languages work with both. Java is a programming language that works with a compiler.

Procedural Programming vs Object Oriented Programming Paradigm

Think of all programming as managing the relationship between two fundamental concepts: state and behavior. State is the data of your program. Behavior is the logic. Procedural Programming is based on implementing these two concepts separately. State is held in data structures. Behavior is held in functions (also known as procedures or subroutines). A procedural application therefore passes data structures into functions to produce some output. Object-Oriented Programming is based on implementing these two concepts together. State and Behavior are combined into one new concept: an Object. An OO application can therefore produce some output by calling an Object, without needing to pass data structures. Advantages of OO include the potential for information hiding: if a caller needn't pass any data structure, then the caller needn't be aware of any data structure and can therefore be completely decoupled from the data format. One fundamental difference between the logic of procedures and the logic of objects is in the way selection is handled. Procedures handle selection using branching logic: the familiar if/else syntax. Objects prefer to handle selection using polymorphism. There are similarities between Procedural and OO as well. Both represent an imperative style of programming, meaning they operate by mutating their state (whether inside a data structure or an object) and providing step-by-step instructions on how to compute output. Imperative programming is like writing a recipe. Finally note that these are idealistic or "pure" definitions. In the real world, paradigms merge. You will rarely, if ever, see a pure OO application. Features from multiple paradigms will be combined, for better or worse.

Imperative versus Declarative Programming

Declarative programming places much of its focus on the overall goal and intended outcome of a program's operations. Developers don't necessarily have to mind how that desired outcome is accomplished, nor hardcode the program's control flow. Declarative code does, however, rely on descriptions of the logic behind various computations. Declarative code does not dictate the order of execution. Instead, it lists the number of available operations and the scenarios for program execution. Unlike the declarative approach, imperative programming emphasizes direct instruction on how the program executes functions. The code consists of a step-by-step sequence of command imperatives. For imperative programming, the order in which operations occur is crucial; it explicitly outlines the steps that dictate how the program implements desired functionality.

Token Concept

The smallest meaningful unit in a programming language is called a token. The program is created by the juxtaposition of tokens. (Tokens can be likened to words in natural languages). For example, "hello world"

include < stdio.h > int main (void) { printf (“Hello World”) ; return 0 ; }

Tokens cannot be further divided into parts. We can divide tokens into six groups:

1) Keywords (Reserved Words): These are words that have a special meaning for the language and are forbidden to be used as variables. For example, if, for, int, return...

C Keywords: auto, break, case, char, const, continue, default, do, double, else, enum, extern, float, for, goto, if, int, long ,register, return, short, signed, sizeof, static, struct, switch, typedef, union, unsigned, void, volatile, while

Java Keywords: abstract, continue, for, new, switch, assert, default, goto, package, synchronized, boolean, do, if, private, this, break, double, implements, protected, throw, byte, else, import, public, throws, case, enum, instanceof, return, transient, catch, extends, int, short, try, char, final, interface, static, void, class, finally, long, strictfp,, volatile, const, float, native, super, while

2) Variables (Identifiers/Variables): These are the tokens that we can name as we want. For example: like x, y, count, printf...

3) Constants (Literals/Constants): A value is either contained in a variable or written directly. Here directly written numbers are called constants. For example:

a = b + 10;

In the expression a and b are floating tokens and 10 are fixed tokens.

4) Operators: Tokens that lead to a transaction and generate a value as a result of the transaction are called operators. For example:

a = b + 10;

Here + and = are operators.

5) Strings (Strings): In programming languages, texts written in double quotes indicate a single token with double quotes. These are called strings. For example: "Hello World\n" like...

6) Separators (Delimiters/Punctuators): All tokens used to separate expressions, except for the above groups, are separator tokens.

Data Types

Type is a basic concept that describes how many bytes an object will occupy in memory and how to interpret the 0's and 1's in it.

Data Types in C

Type	Byte Windows(UNIX/Linux)	Bounded Values
[signed] int	4(4)	[-2147483648, +2147483647]
unsigned [int]	4(4)	[0, +4294967295]
[signed] short [int]	2(2)	[-32768, +32767]
unsigned short [int]	2(2)	[0, +65535]
[signed] long [int]	4(8)	[-2147483648, +2147483647]
unsigned long [int]	4(8)	[0, +4294967295]
char	1(1)	[-128, +127] [0, +255]
signed char	1(1)	[-128, +127]
unsigned char	1(1)	[0, +255]
float	4(4)	±3.6*10-38, ±3.6*10+38]
double	8(8)	[±1.8*10-308, ±1.8*10+308]
long double	8(8)	[±1.8*10-308, ±1.8*10+308]
[signed] long long	8(8)	[-9223372036854775808, +9223372036854775807]
unsigned long long	8(8)	[0, +18446744073709551615]
_Bool	1(1)	true, false

Data Types in Java

Type	Byte	Bounded Values
short	2	[-32768, +32767]
int	4	[-2147483648, +2147483647]
long	8	[-9223372036854775808, +9223372036854775807]
byte	1	[-128, +127]
float	4	[+-3.6*10^-38, +-3.6*10^+38]
double	8	[+-1.8*101-308, +-1.8*10^+308]
boolean	-	true, false
char	2	[0, +65535]

Portability

Portability is the ability of a program written in one programming language to compile and run smoothly when imported into another system. Portability requires a standardization in this sense. Because portability can occur if compilers have always accepted the same rules. The C Programming language is a highly portable language. The portability mentioned here is the portability of the source code. Otherwise, we cannot take a compiled and exe program to another system and run it there. However, there are environments where this is possible. Java is also included in this concept (JVM). Code compiled in these environments can be run when imported into other systems without being recompiled.

Declaration & Definitions

In strongly typed languages such as C and Java, a variable must be introduced to the compiler before it can be used. The process of introducing variables to the compiler before they are used is called declaration. When a declaration is made, if the compiler allocates a memory space for the declared variable, that declaration is also a definition operation. That is, the definition is the declaration operations that the compiler reserves space for. The declaration is more general. Every description is a statement, but not every statement is a description.

The general format of the declaration process is as follows:

<type> <variable list>;

For example:

int a;

long b, c, d;

double x, y;

If the variable list consists of more than one variable, the ',' token is used to separate them. Since space characters can be left as desired between tokens, the following statement is also valid.

long b

,

c, d;

In C declarations can be made in three places:

1) At the beginning of the blocks. In other words, when the block is opened, declarations should be made before any function calls are made. Variables declared at the beginning of the block are also called local variables.

2) Out of all blocks. Variables declared outside of all blocks in C are also called global variables.

3) Within the parameter brackets of the functions. Variables declared in this way are called "parameter variables".

In the Java programming language, variable declarations can be made in three places:

1) Inside the methods: The variables declared inside the methods are called local variables.

2) In the parameter brackets of the methods: These types of variables are called parameter variables

3) Outside of methods but within classes: Such variables are called "member variables of classes".

The following rules apply to variable naming C and Java:

- C and Java are case sensitive programming languages. That is, uppercase letters and lowercase letters are treated as if they were completely different characters.

- Variables do not contain spaces. Variable names cannot be started with numeric characters. However, they can be started with alphabetic characters or _ and continued numeric.

- Only English capital letters, lowercase letters, numeric characters and underscore characters can be used in variable naming.

- According to C standards, compilers must consider a minimum of 32 characters for variable lengths. This limit can vary from compiler to compiler, provided it is at least 32. However, longer variable names are considered valid by the compiler. However, at least the first 32 characters will be considered as separators.

- Keywords cannot be used as variable names either.

C Pointers versus Java References

In C, addresses refer to a completely separate type. Just as there are types such as int, long, double, there are also types that can hold addresses. In C, address types (pointer types) are expressed by the type component of the address. Address types are expressed in C by the type component of the address. For example, this type is not called "address", but "pointer to int", "long type address". Just as there is the concept of int type, long type, double type in C, there is also the concept of address constant. The address constant of type T is created with the type conversion operator. The general format is:

(<type component> *) <numeric component>

The numeric component of the address specifies the physical address number. The type component specifies the type of information at that physical address.

For example:

(int *)0x001B1010

This expression specifies an address constant of type int. For example:

(double *)0x001C2000

This expression means address constant of type double. Asterisk in general format means address. The type component does not have to be specified in hexadecimal when writing address constants.

But tradition is that way. Just as objects holding int type, holding type long, holding type double can be declared in C, objects holding address information can also be declared. These are called pointers.

In Java, a type is related to either value or reference types as a category. If a variable of type T, of which T is a type, directly holds the value itself, type T belongs to value types as a category. If a variable of type T holds an address information and the actual value is located at that address, type T is related to reference types as a category.

The basic types we have seen so far, such as int, long, double, are related to value types as categories. In Java, all class types belong to reference types as categories. That is, it holds a variable address of a class type. When a variable is declared from a class type, only a reference that will hold the address is declared. It is also necessary to create the class object itself and assign its address to this reference. The new operator is used to create class objects in Java. The general form of the new operator is:

new <class name> ([arguments list)

The new operator allocates the object and produces its address. The address given with the new operator must be assigned to a reference of the same type. For example:

Sample s;

s = new Sample();

The operation can also be done by initializing the reference:

Sample s = new Sample();

Type of a reference is much more strictly controlled in Java than the type of a pointer is in C. In C you can have an int* and cast it to a char* and just re-interpret the memory at that location. That re-interpretation doesn't work in Java: you can only interpret the object at the other end of the reference as something that it already is (i.e. you can cast a Object reference to String reference only if the object pointed to is actually a String).

C Structures versus Java Classes

A struct in the C programming language (and many derivatives) is a composite data type (or record) declaration that defines a physically grouped list of variables under one name in a block of memory, allowing the different variables to be accessed via a single pointer or by the struct declared name which returns the same address. The struct data type can contain other data types so is used for mixed-data-type records such as a hard-drive directory entry (file length, name, extension, physical address, etc.), or other mixed-type records (name, address, telephone, balance, etc.). The C struct directly references a contiguous block of physical memory, usually delimited (sized) by word-length boundaries.

In Java classes, a class is a basic building block. It can be defined as template that describes the data and behavior associated with the class instantiation. Instantiating is a class is to create an object (variable) of that class that can be used to access the member variables and methods of the class. A class can also be called a logical template to create the objects that share common properties and methods. For example, an Employee class may contain all the employee details in the form of variables and methods. If the class is instantiated i.e., if an object of the class is created (say e1), we can access all the methods or properties of the class.

Comparing Security

Java is more secured than C. Though, Java uses syntax as same as C which makes it simple and easy. There were many such characteristics which is not available in Java. Some of them are:

- Use of pointer: Java doesn't support pointer (it is only done implicitly).

-Automatic garbage collection.

- Bytecode checking whenever a java compiler creates a bytecode JVM checks bytecode whether it is same or not.

- Java supports Exception Handling. Similarly, there are many other more features which makes java more secure than any other programming languages.

Conclusion

We concluded that C and Java are both two different programming languages. Java is the most popular industry-level language for the development of web applications as well as mobile applications. But we cannot deny that the oldest language, C, is also a very popular language. Because of its flexibility and versatility, everything from microcontroller to operating systems is written in C, providing maximum control with minimum instruction. Both C and Java programming language have their own position in a different aspect, so there are always points that we can compare but not interchangeably.

References:

https://en.wikipedia.org/wiki/C_(programming_language)

https://www.quora.com/Why-doesn-t-C-provide-exception-handling

https://docs.oracle.com/javase/tutorial/java/nutsandbolts/_keywords.html

https://www.quora.com/Is-there-any-relationship-between-structures-in-C-and-classes-in-Java

https://en.wikipedia.org/wiki/Java_(programming_language)

https://www.guru99.com/difference-between-java-and-c.html

https://www.quora.com/Why-is-c++-not-portable-but-Java-is

https://www.quora.com/unanswered/Is-Java-a-portable-language

https://www.calltutors.com/blog/java-vs-c/

https://www.javatpoint.com/java-tokens