COMPARING AND CONSTRASTING: C and JAVA

INTRODUCTION

In this digital age we live in, programming languages have a very important place. The main element of this major technological developments and rapid development of technology is programming languages. Programming languages enable people to turn their ideas into functional software. In rapidly evolving technology, programming languages; It enpowers the developers to innovate, create, and solve complex problems. There are thousands of programming languages available today, and each serves a different purpose. The perfect programming language doesn't exist. Every programming language has a specific capability, functionality and importance in its own field. We do not see programming languages in our daily lives unless we are a developer, but in fact programming languages are there in every aspect of our lives. For example, the building blocks of the software we use on our mobile phones, computers, websites, vending machines, vehicles, airplanes, medical devices, home electronics, artificial intelligence, etc. are programming languages. In other words, programming languages are the heart of technological developments. In this essay we will compare the C and Java, two very important and fundamental programming languages. We will compare and contrast their syntax, semantics, availability, efficiency, learning curve etc. And we will have a look at their paradigms of "object oriented programming" for Java and "procedural programming" for C. Also, we will see how and where these two languages are used with real-world examples.

BACKGROUND and HISTORICAL CONTEXT

For the world of programming languages, C and Java are considered as the foundation of inovation. C was created by Dennis Ritchie in 1970s. It was designed for system development and gained popularity because of its efficiency and flexibility. C became the fundamental for many programming languages like Java, C++, C#, D, Rust, Javascript etc. For Java, it was born in 1995, created by James Gosling and his team at Sun Microsystems. A very important feautre of Java is it is built for being platform-independent which means it is runnable on any device via Java Virtual Machine (JVM). Java

became popular quickly because of it is platform-indepentend, object oriented nature which enables to build robust and dynamic applications.

C remains popular today because of it's direct influence on modern operating systems and embedded systems. C offers precise control over memory management via pointers which makes it very close to the hardware. In contrast to that, Java conduced to a new-era of internet-based applications. Java is favourite language for web and mobile applications because of it’s “write once run everywhere” philosophy which is being platfrom-independent. Both C and Java have shaped the whole software industry for sure. C formed the basis of software engineering princibles and Java formed the web-based technologies.

The programming paradigms of C and Java is very crusial. They changed the way of approach for problem solving and software design. The evolution of programming paradigms are very important as the evolution of programming languages because the new approaches can add more functuality, efficiency and depth to programming languages.

With C, there was a big shift from assembly language to C. Why? Because C provided a significant abstraction compared to assembly language. In assembly you had to think most of the detail about the hardware which makes it hard to track and code for the programmer. In C however, the programmer wasn’t had to concern about the hardware too much thanks to abstraction of C. While C was providing a well abstaction it was still close to hardware. So that it caused a big shift from assembly to C.

For Java, It was born in the era of object-oriented programming (OOP). Object oriented languages focus on data and objects. It encapsulates the data and the behaivor inside the objects. With Java and its object oriented nature enabled programmers to write modular, re-usable code so that the problem of writing complext applications is disappeared. And with the platform-independency via JVM, Java revolutionized how sofware was developed and deployed.

With the popularity of Java, the object oriented paradigm became popular and industry started to use it. But this didn't reduced the importance of procedural programming, no. The popularit of C is a proof of this.

Procedural programming is essential for embedded systems and system progrmaming. This actually caused a diversity in the industry. So that there are different paradigms in the industry and people can choose these tools according to their purpose.

SYNTAX

Syntax in programming languages mean the rules that control the structure of the symbols, punctuation, and words of a programming language. Without syntax, the meaning or semantics of a language is nearly impossible to understand. It determines how instructions are written and interpreted by a computer.

Now let's compare and contrast these two languages in terms of their choice of tokens.

Keywords: keywords are predefined words in a programming language with a specific use.

Control Flow Keywords:

- **break**
- **case**
- **continue**
- **default**
- **do**
- **else**
- **for**
- **if**
- **return**
- **switch**
- **while**

Data Type Keywords:

- **char**
- **double**
- **enum**
- **float**
- **int**
- **long**
- **short**
- **void**

Modifiers and Access Control Keywords:

- **const** (C)
- **final** (Java)
- **volatile** (C, Java)
- **static** (C, Java)

Object-Oriented Programming Keywords (Java-Specific):

- **abstract**
- **implements**
- **instanceof**
- **interface**

Exception Handling Keywords (Java-Specific):

- **throws**

Other Keywords:

- **auto** (C)
- **extern** (C)
- **register** (C)
- **signed** (C)
- **sizeof** (C)
- **struct** (C)
- **typedef** (C)
- **union** (C)
- **goto** (C) – Not recommended because it confuses the flow and it is not appropriate for procedural programming paradigm.
- **package** (Java)
- **transient** (Java)
- **native**
- **private**
- **protected**
- **public**
- **strictfp**
- **super**
- **this**

Identifiers: a sequence of characters used to identify or refer to a program or an element, such as a variable or a set of data, within it.

Identifiers can be variable names, function names, class names etc.

Identifiers are important because if they are well named, the code would be more easy to read, more understandable, and easier to debug.

For example in C *welcome_user*, *age_of_user, username* are identifiers:

```c
void welcome_user() {
  printf("Welcome!");
}

char username[100] = "Standby";

int age_of_user = 21;
```

There are rules for naming identifiers in C and Java.

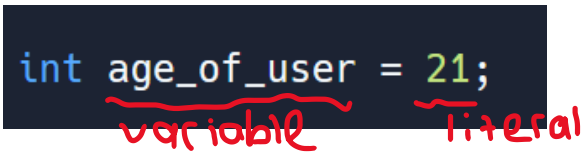
For *C*:

- The identifiers can can consists only letters, digits and underscores (_).
- The first charachter must be a letter or underscore. It cannot start with a number.
- C is case sensitive which means the upper and lower representation of a letter is important.
- “exampleVariable” and “ExampleVariable” are different identifiers in C.
- Identifiers cannot be a keyword or reserved word. Meaning you cannot use “if” or etc. as an identifier.
- C doesn’t have a convention for identifiers but it is a common and good practise to have descriptive identifiers like *get_user_name();* totalAmount;

For *Java*:

- It has similiar rules with C for identifiers.
- Identifiers consists of letters, digits, underscores AND dollar signs ($) different from C.
- The first character can be dollar sign ($) different from C.
- Java has a convention for naming identifiers named camelCase. The initial word doesn’t start with upper-letter but other words start with upper-letter. It improves readability and writing code. Example: *calculateTotalAmount();*

Literals: Literals are the constant values assigned to the constant variables. We can say that the literals represent the fixed values that cannot be modified. It also contains memory but does not have references as variables. For example:



C	*Java*
Integer literal => int a = 5;	Integer literal => int decVal = 26;
Float literal => float b = 27.89;	Float literal => Same with C
Character literal => char ch = 'c'; We can say the escape sequences are character literal too, like: "\n", "\b" etc.	Character literal => Same with C
String literal => char name[10] = "Yasir"; String literals have "\0" at the end always. It is there to understand the end of the string by compiler.	String literal => String fName = "Yasir"; Different from C. Java has keyword of "String".
	Null literals => String age = null;
	Boolean literals => boolean isEven = true;

Operators: An operator is a symbol that tells the compiler to perform specific mathematical or logical functions.

Common Operators in C and Java:

Arithmetic Operators:

C: + (addition), - (subtraction), * (multiplication), / (division), % (modulo)

Java: Same as C

Relational Operators:

C: ==, !=, <, >, <=, >=

Java: Same as C

Logical Operators:

C: && (logical AND), || (logical OR), ! (logical NOT)

Java: Same as C

Assignment Operators:

C: = (assignment), += (addition assignment), -= (subtraction assignment), *= (multiplication assignment), /= (division assignment), %= (modulo assignment)

Java: Same as C

Increment and Decrement Operators:

C: ++ (increment), -- (decrement)

Java: Same as C

Bitwise Operators:

C: & (bitwise AND), | (bitwise OR), ^ (bitwise XOR), ~ (bitwise NOT), << (left shift), >> (right shift)

Java: Same as C

Conditional (Ternary) Operator:

C: condition ? expression1 : expression2

Java: Same as C

Java-Specific Operators:

String Concatenation Operator:

Java: + operator can be used for string concatenation in addition to arithmetic operations. For example: "Hello, " + "World" will result in "Hello, World".

Instanceof Operator:

Java: instanceof operator is used to test if an object is an instance of a particular class or interface.

Type Cast Operator:

Java: Type casting operators like (type) are used for explicit type casting in Java.

Method Reference Operator:

Java: :: is used for method references in Java 8 and above.

Null Coalescing Operator (Java 14 and above):

Java: ?? operator provides a null-safe alternative to the conditional (ternary) operator. It returns the first non-null expression among its operands.

C Specific Operators:

Comma Operator (,):

In C, the comma operator evaluates multiple expressions separated by commas. It evaluates each expression from left to right and returns the value of the rightmost expression. It is often used in for loops and function calls.

Example:

```
  int a = 5, b = 10, c;
  c = (a++, b++, a + b); // Here, a and b are
incremented, and c becomes 16
```

Pointer Operators:

C has special operators for dealing with pointers.

& (Address-of Operator): Returns the memory address of a variable.

* (Dereference Operator): Accesses the value stored at a pointer's address.

Example:

```
  int x = 10;
  int *ptr = &x; // ptr now holds the memory address of x
  int y = *ptr;  // y gets the value stored at the
memory address ptr points to (which is 10)
```

Arrow Operator (->):

Used to access members of a structure or a union through a pointer.

Example:

```
struct Student {
    int id;
    char name[20];
};

struct Student s;
struct Student *ptr = &s;
ptr->id = 1; // Equivalent to (*ptr).id = 1;
```

Bitwise Shift Operators (<< and >>):

These operators perform left and right shifts, respectively, on the bits of an integer.

Example:

int num = 8; // Binary: 0000 1000

num = num << 2; // Left shift by 2 positions: 0010 0000 (Decimal: 32

Bitwise NOT Operator (~):

Performs a bitwise NOT operation on its operand, changing each 1 to 0 and each 0 to 1.

Example:

int num = 5; // Binary: 0000 0101

num = ~num; // Bitwise NOT: 1111 1010 (Decimal: -6 in two's complement)

Sizeof Operator (sizeof):

You get the size of a data type in bytes as return.

Example:

int sizeVar = sizeof(int); // sizeVar contains the size of an integer (4 bytes).

Punctuations: Programming languages use punctuation symbols, such as brackets, parentheses, semicolons, and commas, to create a syntax for writing code. These symbols help to structure and organize code, making it more readable and easier to understand.

Semicolon (;) : Both C and Java uses semicolon (;) to terminate statements with the same way.

Comma (,) : Both in C and Java; commas are primarily used in function calls to separate arguments, to declare multiple variables of the same type in a single statement, to separate elements within structures and arrays, to separate multiple statements within loops and conditionals, allowing multiple actions to occur in a single line of code.

Paranthesis () : Both in C and Java, parentheses are used to enclose the arguments passed to a function during a function call, group expressions and control the order of operations, enclose conditions in if statements, loops, and other control flow constructs.

In Java, parentheses are used to instantiate objects of a class (calling constructors).

Curly Brasis ({ }): Both in C and Java, they serve the purpose of defining blocks of code, including function bodies, loops, conditional statements, and other control flow structures.

Now lets talk about the Syntax Rules of C and Java.

SYNTAX RULES:

1) Statements and Blocks:

Statements are the building blocks of programs. They are terminated by ";".

Code blocks are expressed with { }.

Denoting of code blocks are same in C and Java.

```
if (condition) {
    // Code block in C, denoted by curly braces
    // Statements go here
}
```

2) Control Structures:

a) Conditional Statements:

Same in C and Java.

```
 if (condition) {
    // Code to execute if the condition is true
 } else if (anotherCondition) {
    // Code to execute if the second condition is true
(optional)
 } else {
    // Code to execute if none of the conditions are
true (optional)
 }
```

b) Loops:

Same in C and Java.

```c
for (int i = 0; i < 10; i++) {
    // Code for a for loop in C
}

while (condition) {
    // Code for a while loop in C
}

do {
    // Code for a do-while loop in C
} while (condition);
```

3) Function and Methods:

There is function for C and method for Java. Because they have different programming paradigm.

But definiton of them are syntaticly same.

```c
int add(int a, int b) {
    // Function definition in C
    return a + b;
}
```

4) Parameters and Return types:

In C, function parameters and return types are explicitly declared.

In Java, method parameters and return types are explicitly declared.

5) Function/ Method overriding:

C: Function overloading is not supported in C.

Java: Method overloading allows defining multiple methods with the same name but different parameter lists.

6) Classes and Structures:

a) Creating Class:

Class can be created in Java because it is object oriented. However C, uses struct similiar to classes.

```java
public class MyClass {
    // Class definition in Java
    // Fields, methods, and other members go here
}
```

b) Creating Struct:

```c
struct Point {
    int x;
    int y;
}; // Structure definition in C
```

7) Differences in Syntax, Inheritance, and Access Control:

Java: Classes support inheritance, encapsulation, and access control (public, private, protected). Methods and variables can have access modifiers.

C: Structures do not support inheritance or access control. Members are public by default.

SEMANTICS

The semantic of a programming language is the logic side of syntax of the code. What to implement with that code the meaning of that code.

About semantics, we will compare and contrast: 1) Data Types and Type Checking, 2) Control Structures, 3) Paradigms, 4) Memory Management, 5) Exception Handling.

1) Data Types and Type Checking:

Both C and Java are strongly typed languages which means the variables are declared with their type and their type cannot be changed afterwards. For example: int a = 15; a = “hehey”; It is an error.

Data types of C: 1) Basic Data Types, 2) Derived Data Types, 3) void

1) Basic Data Types: int, char, float => integer, character, floating point number

2) Derived Data Types: int arr[10], int* ptr, struct, union, enum => array, pointer, ..

3) void => indicates that function returns nothing.

Data types of Java:

Primitive Data Types => store simple values.	**Reference Data Types** => store references (memory adresses), they don't hold the actual data instead a reference to where the data is stored in memory.
Integers => byte, int, short, long Floating Point Numbers => float, double Characters => char Booleans => boolean	Classes => instances of classes (objects) Interfaces => instances of it Arrays => int [] arr; Enums => enum String => String name = "Yasir"

Type Checking: Type checking is the process of verifying that the types of operands in a program are compatible with the operator being used and that the program is free from type errors before the code is executed.

Both in C and Java type checking is done at compile-time which means before program has executed if there is an error program doesn't start in contrast of interpreted language like Python. This provides more safety and prevents unintented behaivor.

However Java has dynamic binding for objects allowing method calls to be resolved at runtime based on the actual type of the object. This enables greater flexibility in method invocation while preserving static typing for variables.

2) Control Structures:

C	Java
• Basic Control Structures => if-else statements, for, while loops, switch-case statements. • No Enchanced Structures • Exception handling handled by global error codes by crashing the program. Not so smoothly like Java.	• Basic Control Structures => Just like C. • Addition to C like structures. Java has enchanced for loop which is making it easier to iterate through the collections. • Exception Handling => Java has try,catch structures for error handling which allows developers to handle runtime errors smoothly.

3) Paradigms

C:

Procedural Programming: C uses a procedural programming paradigm, focusing on procedures or functions to operate on data.

C does not have built-in support for object-oriented programming concepts like classes and objects.

Java:

Object-Oriented Programming: Java is designed as an object-oriented programming language, emphasizing the use of objects and classes for organizing code and data.

Java supports key object-oriented concepts, including classes, objects, inheritance and polymorphism.

Java allows encapsulation of data within objects, and abstraction, enabling the creation of complex systems by modeling real-world entities with simplified interfaces.

4) Memory Management:

C:

In C, developers are responsible for allocating memory using functions like malloc() and deallocating it using free(). This manual management provides control but it is very imporant to carefully handling to prevent memory leaks or dangling pointers.

Pointers: C uses pointers which are variables that store memory addresses, allowing direct manipulation of data in memory.

Java:

Automatic Memory Management (Garbage Collection): Java handles automatic memory management with process called Garbage Collection. The Java Virtual Machine (JVM) automatically identifies memory occupied by objects no longer in use. This reduces the risk of memory leaks.

References: Java uses references instead of pointers. References are high-level handles to objects, but they do not expose direct memory addresses, enhancing safety and security.

AVAILABILITY, EFFICIENCY AND LEARNING CURVE:

Avaliability:

C	*Java*
Libraries and Frameworks: C doesn't have a built-in libraries for high-level functionalities. But there are plenty of third-party libraries available like graphics, data processing etc.	Libraries and Frameworks: Java comes with a very rich standart library from basic data structures to more advanced things. Also Java has plenty of frameworks especially Spring Boot framework is very popular for enterprise applications.

C developers use GCC compiler and IDE's like Dev-C++, Code Blocks, Visual Studio.	Java developers use JDK (Java Development Kit) and IDE's like IntelliJ IDEA, Eclipse.

Efficiency and Performance:

C	*Java*
C is popular for its speed. Because of it is a low-level language and its closeness to the hardware it in the fastest programming languages.	Java is not as fast as C because of being a higher-level language so its additional abstractions. But it's performance has significantly improved in years thanks to the optimizations. So that java is a good choice for server-side applications and large-scale systems.

Learning Curve:

C's learning curve is steep because of it's closeness to the hardware and low-level concepts like pointer and memory management.

For Java, it has a nicer learning curve because it's high-level abstactions and automatic memory management with garbage collector. Also Java has much more community support, tutorials and nice, good looking, simple documentation compared to C.

REAL WORLD APPLICATONS

C*: used in system programming because of it is close to hardware and low-level language.*	***Java****: used in enterprise applications, web development, mobile development etc. because of it is high-level, scalable and has high performace.*
• Linux Kernel • Git • MySQL • Microsoft Windows OS • Oracle Database • Wireshark • Mozilla Firefox	• Android OS • Eclipse IDE • Twitter • Amazon Web Services • Linkedin • Netflix • Google Cloud Platform

CONCLUSION

In conclusion, our comparing and contrasting of C and Java shows the different strengths and applications of these programming languages. C is very important for system-level programming and projects requiring optimal performance with its close to hardware nature. On the other hand, Java's portability, strong object-oriented features, and rich ecosystem makes it a very handy in web development, mobile applications, and enterprise applications.

C, by being close to hardware, by being fast while providing abstraction on the other hand Java by being platform independent with its object oriented nature and being fast; they made revolution in the industry.

Understanding the differences of programming languages enriches our knowledge and creates a better vision for us for how we can reach our goal by selecting the right language.