**Rust: Ownership and Borrowing**

fn main() {

let s = String::from("Hello, Rust"); // memory allocated on the heap

print\_string(&s); // borrow reference, no ownership transfer

println!("Original string is still valid: {}", s);

}

fn print\_string(s: &String) {

println!("Borrowed string: {}", s);

}

**Java: Garbage Collection**

public class MemoryDemo {

public static void main(String[] args) {

String s = new String("Hello, Java"); // heap allocation

System.out.println("Created string: " + s);

s = null; // dereference

System.gc(); // suggest garbage collection

System.out.println("Garbage collector will reclaim memory later.");

}

}

**C++: Manual Memory Management**

#include <iostream>

using namespace std;

int main() {

string\* s = new string("Hello, C++"); // manual allocation

cout << "Created string: " << \*s << endl;

delete s; // manual deallocation

// cout << \*s; // Uncommenting this line would cause a dangling pointer error

return 0;

}