Java, C++ and Python A Comprehensive Comparison

Jerry Tafoya
New Mexico Institute of Mining and Technology
Department of Computer Science
Socorro, New Mexico
jerry.tafoya@student.nmt.edu

ABSTRACT

This paper compares Java Python and C++ through many different scopes of programming. Major topics are brought up and each of these languages fundamentals are put up against each other to allow for in-depth analysis. There are many examples of source code to back up claims made in this article along with a full list of resources used to construct them.

KEYWORDS

Java, C++, Python, Programming, Polymorphism, Concurrency, debugging, code maintenance, programming security, typing system, compiler, program speed, program efficiency, platform independence

1 INTRODUCTION

Java is a High Level Language (HLL) based upon object oriented programming and class based logic, it runs with help from the JVM (Java Virtual Machine) which provides it a portable execution environment. C++ is an object oriented extension of C, it can be coded with both C and C++ syntax. Python is an interpreted object oriented programming language. It has very powerful HLL semantics, data structures and typing system. There are many factors that these languages differ on, one of such factor is power.

2 POWER

Power in programming generally implies short syntax with powerful semantics[10]. Something such as recursion or iterative programming are essential tools in powerful programming.

2.1 Polymorphism

Polymorphism is when something occurs in several different forms, where objects or functions of the different types can be accessed through the same interface [12]. Within polymorphism there are two subcategories **static & dynamic**. Static polymorphism uses method overloading so that a method with the same name can take different parameters [12]. Dynamic polymorphism is where the the program determines which method to use at run time (much like the JVM) where in an inheritance relationship, the subprogram can change or over write a method in a parent or superprogram [12].

2.1.1 Java. In Java polymorphism is done by method overloading and method overriding [13]. In method overloading (runtime polymorphism) the JVM can determine which method is being called, so a subclass can be an instance of a superclass and use the subclass method. For example in the following code the subclass method is being used.

```
class Bike{
  void run(){System.out.println("running");}
}
class Splendor extends Bike{
  void run(){System.out.println("running safely with 60km");}

  public static void main(String args[]){
    Bike b = new Splendor();//upcasting
    b.run();
  }
}
```

[13]

So the output of this code is "running safely with 60km". Here the JVM has determined that the subclasses method should be used because "Bike b" in main is an instance of its subclass, this is an example of method overriding

2.1.2 C++. Much like Java, C++ also has compiletime and runtime polymorphism through function overloading and function overriding [14]. in C++ when a derived class has a definition of one of the members of the base class, the base classes member is overridden. An example of this can be seen below.

```
// C++ program for function overriding

#include <bits/stdc++.h>
using namespace std;

class base
{
public:
    virtual void print ()
    { cout<< "print base class" <<endl; }

    void show ()
    { cout<< "show base class" <<endl; }
};

class derived:public base
{
public:
    void print () //print () is already virtual function in derived class, we could also declar
    { cout<< "print derived class" <<endl; }

    void show ()
    { cout<< "show derived class" <<endl; }
};

//main function
int main()
{
    base *bptr;
    derived d;
    bptr = &d;

    //virtual function, binded at runtime (Runtime polymorphism)
    bptr->print();

    // Non-virtual function, binded at compile time
    bptr->show();

    return 0;
}
```

Where the output of this program is "print derived class show base class", using the *Virtual* keyword in C++ yields this kind of behavior

where the member in the base class can be overriden at runtime [14].

2.1.3 Python. In Python there is only static polymorphism which is possible through inheritance, a common practice in python is multiple inheritance [15].

```python
class A:
    def explore(self):
        print("explore() method called")

class B:
    def search(self):
        print("search() method called")

class C:
    def discover(self):
        print("discover() method called")

class D(A, B, C):
    def test(self):
        print("test() method called")


d_obj = D()
d_obj.explore()
d_obj.search()
d_obj.discover()
d_obj.test()
```

[15]

The output of the following program is "explore() method called search() method called discover() method called test() method called" This kind of polymorhpism is very useful when it comes to having a object in main utilizing multiple tools.

2.1.4 Critique. In Java polymorphism is mostly contained within the JVM or during the compile time, this is easy for the Java programmer to keep track of [13]. In C++ with the *Virtual* keyword you can recreate this same kind of polymorphism as foudn in Java which gives more power to the developer [14]. In Python only static polymorphism is available, however this is still powerful and keeps most of the work of polymorphism in compile time [14].
C++ gives the programmer the most tools when it comes to this concept, however this trusts the user more than Java so it can easily lead to a security issue. Python is the safest of the two only allowing for one type of polymorphism, however it will also be the safest in this case due to the limitation of the langauge[15].

2.2 Other Factors

Other aspects that contribute to a languages power are Modularity, Abstraction, Typing system Strength and Concurrent processes[11][6][8]. Modularity is the separation of different pieces of code, this helps with reusability, maintenance and **safe** separation between the user and implementer domains [11]. Abstraction shifts some of the complexity to the algorithmic solution encoding done by a HLL's compiler[11]. A Typing System is a set of rules that places a type onto various constructs in a program[6]. Concorrent processes are processes that run along side each other, this is a very good way to run programs quickly [8].

2.2.1 Concurrency.

2.2.2 Java. Java supports Concurrent programming, this is done by extending a subclass from Thread and overriding run() method, this uses the same core to run multiple processes at one [16].

```java
class MyThread extends Thread {
   // override the run() method
   @Override
   public void run() {
      // Thread's running behavior
   }
   // constructors, other variables and methods
   ......
}
```

```java
public class Client {
   public static void main(String[] args) {
      ......
      // Start a new thread
      MyThread t1 = new MyThread();
      t1.start();  // Called back run()
      ......
      // Start another thread
      new MyThread().start();
      ......
   }
}
```

This is the only way to do this since Java does not support multiple inheritance the same way that Python does[16].

2.2.3 C++. In C++ creating a Thread requires the "thread" library so that you can create that new object[17].

```cpp
#include <iostream>
#include <thread>
using namespace std;

void func(int x) {
    cout << "Inside thread " << x << endl;
}

int main() {
    thread th(&func, 100);
    th.join();
    cout << "Outside thread" << endl;
    return 0;
}
```

[17]

This is much easier than in Java since it can give you the thread object here in main. This is a very powerful and abstract way to create concurrency in C++ [17].

2.2.4 Python. In python there are the threading and multiprocessing librarys which do separate things. threading allows python to create and manage threads much like Java or C++, though python has poor performance in this area.[18].

```python
import threading


def countdown():
     x = 1000000000
     while x > 0:
          x -= 1

# Implementation 1: Multi-threading
def implementation_1():
     thread_1 = threading.Thread(target=countdown)
     thread_2 = threading.Thread(target=countdown)
     thread_1.start()
     thread_2.start()
     thread_1.join()
     thread_2.join()

# Implementation 2: Run in serial
def implementation_2():
     countdown()
     countdown()
```

Here this program is testing static (linear) implementation and comparing it against the threading method. it takes threading a

bit longer to run. This is because the Global Interpreter Lock only allows one thread from accessing the program at a time[18].

```
In [18]: %time implementation_1()
CPU times: user 1min 38s, sys: 200 ms, total: 1min 38s
Wall time: 1min 38s

In [19]: %time implementation_2()
CPU times: user 1min 34s, sys: 26.7 ms, total: 1min 34s
Wall time: 1min 34s
```

multiprocessing in python bypasses this Global Interpreter Lock and allows for multiprocessing.

```
import multiprocessing

# countdown() is defined in the previous snippet.

def implementation_3():
     process_1 = multiprocessing.Process(target=countdown)
     process_2 = multiprocessing.Process(target=countdown)
     process_1.start()
     process_2.start()
     process_1.join()
     process_2.join()
```

This method took about half the time as the other two processes, it is insecure however since it can allow you to mutate your code while its processing which would lead to undefined behavior[18].

2.2.5 Critique. In Java concurrency is done with one thread running at a time, this means that there is a small chance of starvation (when one or more threads in your program are blocked from gaining access to a resource) or deadlock (when two or more threads are waiting on a condition that cannot be satisfied) which makes it relatively safe[19]. In C++ concurrency can run separate functions as seen in 2.2.3, this is a safe way to use threading as well since it minimizes starvation and deadlock. Python gains much more power by allowing the user to run multiple processes at once, however this is much more insecure than Java and C++ since these simultaneous processes can lead to mutations in the code or amplified problems if they do arise. Trusting the user is a surefire way to bigger insecurity.

3 READABILITY & MAINTENANCE

Readability and Maintenance of a programming language depends on its simplicity, Syntax and Typing system. Where simplicity is related to orthogonality which states that a small set of primitive constructs can be combined into relatively small numbers of ways[1]. Syntax is a set of the rules that specify the sequence of symbols used to correctly construct a program [4]. a Typing System defines everything else like math functions or structures in the program with different user defined properties[6].

3.0.1 Typing System.

3.0.2 Java. In java there are many aspects of the typing system that allow the user to create anything that they want. For example to create a multilevel inheritance program there are many keywords that allow them to do so, such as *extends* which takes the properties of the super class and applies them to the base class [20].

```
Class X
{
   public void methodX()
   {
     System.out.println("Class X method");
   }
}
Class Y extends X
{
public void methodY()
{
System.out.println("class Y method");
}
}
Class Z extends Y
{
   public void methodZ()
   {
     System.out.println("class Z method");
   }
   public static void main(String args[])
   {
     Z obj = new Z();
     obj.methodX(); //calling grand parent class method
     obj.methodY(); //calling parent class method
     obj.methodZ(); //calling local method
  }
}
```

This example allows for an Object Z to inherit properties from Class Y and X, this typing system adds simplicity and abstraction to the language while sacrificing efficiency.

3.0.3 C++. In C++ its nearly the same case as java, the difference being you use the **:** keyword and then the class you want to inherit from [20].

```
class base    //single base class
{
   public:
     int x;
   void getdata()
   {
     cout << "Enter the value of x = "; cin >> x;
   }
};
class derive : public base    //single derived class
{
   private:
    int y;
   public:
   void readdata()
   {
     cout << "Enter the value of y = "; cin >> y;
   }
   void product()
   {
     cout << "Product = " << x * y;
   }
};

int main()
{
   derive a;     //object of derived class
   a.getdata();
   a.readdata();
   a.product();
   return 0;
}        //end of program
```

C++ will also abstract the **new** keyword that Java has (unless you need to allocate memory) making it slightly easier to read and maintain for the C++ programmer, this does however sacrifice efficiency.

3.0.4 Python. Python supports multiple inheritance as well, it can implement methods of multiple classes [21].

```
# Python example to show working of multiple
# inheritance
class Base1(object):
    def __init__(self):
        self.str1 = "Geek1"
        print "Base1"

class Base2(object):
    def __init__(self):
        self.str2 = "Geek2"
        print "Base2"

class Derived(Base1, Base2):
    def __init__(self):

        # Calling constructors of Base1
        # and Base2 classes
        Base1.__init__(self)
        Base2.__init__(self)
        print "Derived"

    def printStrs(self):
        print(self.str1, self.str2)


ob = Derived()
ob.printStrs()
```

This type of inheritance is more powerful than C++ and Java since it allows the Python user to utilize tools from different classes[21].

3.0.5 Critique. Java and C++ don't support multiple inheritance so when an issue is found it can be tracked back linearly reducing the workload on the compiler. Still, Pythons usage of multiple inheritance is very powerful as it allows the Python programmer to implement complex structures in an easier manner, at the loss of security and efficiency. In this scpoe of the language design Python is the easiest to read and implement, but Java and C++ are easier to maintain and debug due to their linear inheritance relationships.

4 SECURITY

The Security of a language relies on many factors, some of which include Exception Handling and Data types. Exception Handling is how users process errors in a program by isolating exceptions that happen during run time[7]. Data types have predefined characteristics, for example in c its very easy to write code insecurely using strings and buffers [22].

4.0.1 Exception Handling.

4.0.2 Java. In Java Exception Handling is mainly utilized using the *Try* and *Catch* blocks, where try tests code for any exceptions and catch checks to see if a specified exception is caught[23].

```
// Java program to demonstrate exception is thrown
// how the runTime system searches th call stack
// to find appropriate exception handler.
class ExceptionThrown
{
    // It throws the Exception(ArithmeticException).
    // Appropriate Exception handler is not found within this method.
    static int divideByZero(int a, int b){

        // this statement will cause ArithmeticException(/ by zero)
        int i = a/b;

        return i;
    }

    // The runTime System searches the appropriate Exception handler
    // in this method also but couldn't have found. So looking forward
    // on the call stack.
    static int computeDivision(int a, int b) {

        int res =0;

        try
        {
          res = divideByZero(a,b);
        }
        // doesn't matches with ArithmeticException
        catch(NumberFormatException ex)
        {
            System.out.println("NumberFormatException is occured");
        }
        return res;
    }

    // In this method found appropriate Exception handler.
    // i.e. matching catch block.
    public static void main(String args[]){

        int a = 1;
        int b = 0;

        try
        {
            int i = computeDivision(a,b);

        }

        // matching ArithmeticException
        catch(ArithmeticException ex)
        {
            // getMessage will print description of exception(here / by zero)
            System.out.println(ex.getMessage());
        }
    }
}
```

Here the program is catching the *ArithmaticException* thrown from division by zero in Java. Processing Errors in this way allows Java users to Override default handling methods.

4.0.3 C++. C++ implements a very similar method as Java as it also uses a try and catch block structure[23].

```
#include <iostream>
using namespace std;

double division(int a, int b) {
   if( b == 0 ) {
      throw "Division by zero condition!";
   }
   return (a/b);
}

int main () {
   int x = 50;
   int y = 0;
   double z = 0;

   try {
      z = division(x, y);
      cout << z << endl;
   } catch (const char* msg) {
     cerr << msg << endl;
   }

   return 0;
}
```

However C++ does not have a mathematical error exception so the user has to implement that themselves[23].

4.0.4 Python. Python supports a Try, Except, Else block structure where the the except block runs if an exception is thrown and the else block runs if there is no error[24].

```
try:
    linux_interaction()
except AssertionError as error:
    print(error)
else:
    try:
        with open('file.log') as file:
            read_data = file.read()
    except FileNotFoundError as fnf_error:
        print(fnf_error)
```

This shows a simple implementation of this idea showing if there is no error in *linuxinteraction*, then the program will try the else block which holds another try except chunk.

4.0.5 Critique. C++, Java and Python all hold similar Exception Handling techniques where they test regions of code and check if a specific exception is thrown. This is a relatively simple and easy way to check for exceptions as it allows the programmer to change the behavior of their code depending on what fails.

5 ROBUSTNESS

A robust program runs well not only under ordinary conditions, but also under unusual conditions that stress its designer's assumptions [25]. Which means that the code can compensate for something that the user could not. An example of Robustness is how a language processes and collects dynamic memory allocation. Memory Garbage Collection is the automatic memory management that is done by the compiler every time memory is needed by the user[9].

5.0.1 Java. In java memory management is handled by the JVM, The JVM holds five main areas for memory management, the Heap area, Method area, JVM stack, Native Method stack and PC (Program Counter) Registers [26]. The Heap stores objects from runtime, there is only ever one heap in a JVM process. Consider the following code.
Scanner sc = new Scanner(System.in);
here the JVM stores the Scanner object on the heap and the sc reference onto the JVM stack[26].
the Method area is the logical part of the heap where this memory is allocated for class structures method data and constructor field data [26].
The JVM Stacks are used to store data and partial results needed for return values[26].
The Native method stacks is the memory allocated for each thread in concurrent programming[26].
finally the Program Counter (PC) keeps track of each methods specific program counter register[26].

5.0.2 C++. C++ has defined keywords for the creation and removal of memory. These keywords are **new** and **delete** where new is used to allocate heap memory to a variable or class object [27] and where delete is used to deallocate the memory space that is used from the new keyword[27]. Here is an example of the following code
int *op = new int[10];
/* Code here */
delete op;
This form of memory management can lead to dangling pointers however so its up to the users skill to successfully utilize this[27]

5.0.3 Python. In python memory management is handled behind the scenes, for example in CPython the memory management is handled in the underlying C code where memory management is up to the user[28]. This implementation describes this allocation as "a fast, special-purpose memory allocator for small blocks, to be used on top of a general-purpose malloc" so in this way the program will only allocate for every 8 bytes requested[28].

5.0.4 Critique. Each of these languages use different methods to handle memory management. Java keeps everything on the back end with its own virtual machine while C++ makes the user handle all of the memory management with python letting the program that it was created on (c) handle all of the memory management for it. C++ has the most efficient method of memory management because it takes less resources for the compiler and run time to allocate this itself, however its the most insecure since the user can forget to deallocate the memory leading to memory leakage. Python and Java on the other hand are much more abstract yet they lose efficiency for this advantage so its up the user to decide which language is better for them in their case.

6 EFFICIENCY

Efficiency is the speed of execution in both design and implementation of the software which include compilation speed, program execution speed and memory space utilization[11]. Examples of efficiency include smart a use of Control Structures and type of compilation. Control Structures control how a program is ran, there are three types of control structures that include sequential, selection and repetition [2]. Sequential control is the default mode of programming that takes code and processes it line by line[2]. Selection control is used for decisions or branching, such as the if else block in Java or C++[2]. Repetition Control is used for looping code to perform an action over and over, much like while blocks or for blocks in Java[2]. There are also three types of compilation which include Compiled, Interpreted and Hybrid methods. Compiled code takes one program statement at a time and processes it into machine code (the 1's and 0's that the CPU runs), this method is fast at analyzing code but is overall relatively slow[10]. Interpreted code scans the entire program and translate it as a whole into machine code, this method takes a large amount of time to analyze the source code but the execution is much faster compared to compiled code[10]. Hybrid compilation uses both methods optimizing on both [10]

6.0.1 Java. Java is a compiled language, its compiled down to bytecode by the java compiler and executed by the JVM[29].

6.0.2 C++. C++ is also a compiled language and has three main steps. Preprocessing, Compilation & Assembly and Assembly[30]. Preprocessing takes the header files and defining values such as ifndef and endif and temporarily expends it to prepare for compilation[30].

Compilation & Assembly converts the C++ into assembly code. at this stage the compiler detects errors[30]. Assembly takes the assembly code and turns it into machine code[30].

6.0.3 Python. Python is a hybrid language being both compiled and interpreted[31]. Python is not compiled, yet it has the same byte code as the other two languages Java and C++, it also uses a virtual machine to execute the bytecode much like Java does. the compilation is implicit so its a hybrid of interpreted and compilation.

6.0.4 Critique. A Compiled language is in byte form and it takes several steps to compute in order to get to this point, a compiled program will only work for the platform it was designed for[32]. For an Interpreted program the source code is the program, and its run as a script, this is very portable if the machine has the correct script software to run the program. Interpreted compilation is generally slower than compiled programs. But interpreted programs are easier to debug and edit[32]. If you choose Java with its JVM technology then you would have the quickest run time speed since Java employs a *Just In Time Compilation* Scheme[32]. However poor Java programming can force the Java run time environment to generate most of the code upfront making it slow. Choosing C++ will give you the most speed at compile time since compilation beats interpretation here. Yet also choosing C++ will give you the hardest code to debug and maintain. Choosing Python will give you code that's the easiest to debug and maintain, with a downside of being generally slower than compiled languages so less efficiency.

7 PLATFORM INDEPENDENCE

Platform independence is the property that you can run code with little to no modification on multiple systems[33]. This can depend on the hardware/software of the machine and there are many corner cases [33].

7.0.1 Java. Java is very platform independent. A .jar file on Windows will run identically as that same .jar file on linux[33]. The Java Libraries are generally platform independent if they are written in pure Java[33]. The JVM and the runtime environment are also independent of platform as the runtime environment is supported on many machines and as the JVM which only needs the right version for your machine[33].

7.0.2 C++. C++ is considered to be platform dependent, as its compiled into machine code only the machine it was compiled on can run that code[34]. In order to execute that code on another machine you would need to move the entire program there as well, completely removing security from the code since it can be changed by others[34].

7.0.3 Python. Python is platform independent, there are many python interpreters that allow the program to run on many operating systems[35]. Python can even be packaged into stand alone executable programs for popular operating systems[35].

7.0.4 Critique. For a program to be platform independent it must compile so something other than that system can execute the program. Java uses a JVM and a runtime environment that are built specifically for each system, so Java can be run on nearly all machines[33]. C++ is platform dependent since its compiler turns it into machine code that only one system can use, so you would have share the code with the other system and have that system compile it there for that machine, which means there is no code security since others can edit the code[34]. Python like Java is platform independent since there are many interpreters that can execute python code[35].

8 SUMMARY

Depending on what you need in a program there is a lot to think about when choosing a programming language for a project. If you want a language with a wide range of classes and platform independency then Java is definitely the best choice. If you want a language that is short yet powerful, interpreted and very easy to code in then python is good for you. If you need more Lower Level functions and more control over Memory management and generally more user control then C++ is good. But each of these languages have their downsides that dont make them entirely the correct choice each time a project needs to be started. So its up to you to determine if you are willing to pay the price for the benefits that any one of these programming languages offers.

REFERENCES

[1] What Is Orthogonality in the Context of Programming Language Design. Edited by Malikravi908@gmail.com, www.javatpoint.com, 16 Nov. 2013, www.javatpoint.com/q/4339/what-is-orthogonality-in-the-context-of-programming-language-design.

[2] Control Structures - Intro, Selection. www.fsu.edu https://www.cs.fsu.edu/ myers/c++/notes/control1.html

[3] Data Types and Data Structures. www.sqa.org.uk https://www.sqa.org.uk/elearning/LinkedDS01CD/page 03.html

[4] What is syntax? www.techopedia.com https://www.techopedia.com/definition/3959/syntax

[5] OOP Concept for Beginners: What is Abstraction? Edited by Thorben Janssen stackify.com 23 Nov. 2017, https://stackify.com/oop-concept-abstraction/

[6] An introcution to programming type systems. Edited by Zach Grossbart, www.smashingmagazine.com, 18 April. 2013 https://www.smashingmagazine.com/2013/04/introduction-to-programming-type-systems/

[7] What is an exception? docs.oracle.com https://docs.oracle.com/javase/tutorial/essential/exceptions/definition.html

[8] What is concurrent programming? Edited by Marcas Neal www.quora.com 22, Dec. 2015 https://www.quora.com/What-is-concurrent-programming-1

[9] What is Java Garbage Collection? How it works, best practices, tutorials, and more. stackify.ocm 11 May, 2017 https://stackify.com/what-is-java-garbage-collection/

[10] Interpreter Vs Compiler: Difference Between Interpreter and Compiler. www.programiz.com https://www.programiz.com/article/difference-compiler-interpreter

[11] MacLennan, Bruce J. Principles of Programming Languages: Design, Evaluation, and Implementation. 3rd ed., Oxford University Press, 1999.

[12] OOP Concepts for Beginners: What is Polymorphism? www.stackify.com Edited by Thorben Janssen 22 Dec, 2017 https://stackify.com/oop-concept-polymorphism/

[13] Polymorphism in Java. www.javatpoint.com https://www.javatpoint.com/runtime-polymorphism-in-java

[14] Polymorphism in C++. www.geeksforgeeks.org https://www.geeksforgeeks.org/polymorphism-in-c/

[15] Inheritance and Polymorphism in Python. www.overiq.com https://overiq.com/python-101/inheritance-and-polymorphism-in-python/

[16] Java programming tutorial Multithreading & Concurrent Programming. www.ntu.edu.org http://www.ntu.edu.sg/home/ehchua/programming/java/j5e multithreading.html
[17] Concurrency in C++11. uchicaho.edu https://www.classes.cs.uchicago.edu/archive/2013/spring/12300-1/labs/lab6/
[18] Concurrent Programming in Python is not what you think it is. www.hackernoon.com Edited by Melvin Koh 07 Oct, 2018 https://hackernoon.com/concurrent-programming-in-python-is-not-what-you-think-it-is-b6439c3f3e6a
[19] Starvation and Deadlock. www.uni-hamburg.com https://www.math.uni-hamburg.de/doc/java/tutorial/essential/threads/deadlock.html
[20] Types of inheritance in Java: Single, Multiple, Multilevel & Hybrid. www.beginnersbook.com Edited by Chaitanya Singh
[21] Inheritance in Python, www.geeksforgeeks.com Edited by Shivangi Srivastava. https://www.geeksforgeeks.org/inheritance-in-python/
[22] Writing Insecure C, Part 3. www.infomit.com Edited by David Chisnall 24 Oct. 2008 http://www.informit.com/articles/article.aspx?p=1249299
[23] Exceptions in Java. www.geeksforgeeks.com https://www.geeksforgeeks.org/exceptions-in-java/
[24] Python Exceptions: An Introduction. www.realpython.com Edited by Said van de Klundert 03 April, 2018 https://realpython.com/python-exceptions/
[25] Robust Definition. 20 June, 2005 http://www.linfo.org/robust.html
[26] Java Memory Management. Edited by thevirtualink https://www.geeksforgeeks.org/java-memory-management/
[27] Dynamic Memory Allocation in C++. www.studytonight.com https://www.studytonight.com/cpp/memory-management-in-cpp.php
[28] Memory Management in Python. www.realpython.com Edited by Alexander VanTol 21 Nov. 2018 https://realpython.com/python-memory-management/
[29] Is Java a Compiled or an Interpreted programming language? www.stackoverflow.com Edited by Mehrdad Afshari 23 Oct 2015 https://stackoverflow.com/questions/1326071/is-java-a-compiled-or-an-interpreted-programming-language
[30] How C++ Works: UNderstanding Compilation. www.freelancer.com Edited by Lucy Karinsky 31 Aug. 2017 https://www.freelancer.com/community/articles/how-c-works-understanding-compilation
[31] Is Python interpreted or compiled? Yes. www.nedbatchelder.com Edited by Ned Batchelder 29 Mar. 2018 https://nedbatchelder.com/blog/201803/is python interpreted or compiled yes.html
[32] What is the difference between a compiled and an interpreted program? kb.iu.edu 18 Jan. 2018 https://kb.iu.edu/d/agsz
[33] What is the exact meaning of Platform independence? www.stackexchange.com Edited by mikera 18 Jun. 2011 https://softwareengineering.stackexchange.com/questions/85175/what-is-the-exact-meaning-of-platform-independence
[34] Why C,C++ are platform dependent languages and how Java archives platform independency? www.facebook.com Edited by Learn Java Programming 15 Nov. 2014 https://www.facebook.com/807685839255768/posts/q-6-why-c-c-are-platform-dependent-languages-and-how-java-achives-platform-indep/893563670667984/
[35] Is Python Platform independent? www.quora.com Edited by Kamal Seth 3 Aug. 2016 https://www.quora.com/Is-python-platform-independent