Cross-platform benefits of the Java language

Chengyang Li & Siyuan Wang,

BASE 34, Nov. 9th, 2017

University of Waterloo

Nowadays, with the development of the IT field, computer languages are becoming more and more popular; they provide services for daily work and study. In specific words, we use operating systems along with software everyday. According to Hansen (1973), “An operating system is a set of manual and automatic procedures that enable a group of people to share a computer installation efficiently.”(p. 1) The operating system is used for managing the hardware, providing some functions, and providing an environment for the program to run on it (Silberschatz, Galvin, & Gagne, 2013). Program is a list of instructions to the computer to do some function, which is wrote by some computer program (Rouse, 2007). Nowadays, apart from Java, there are many popular computer languages such as Python, C and Ruby. However, a problem in these computer languages is that different operating systems require different program languages. The reason is that the device-independent graphics is not provided, and it is necessary for the programmers to know details of the hard devices in order to illustrate the graphics (Chapman, 1999).

One of the problem’s drawbacks is that programmers need to write the same program many times base on different platforms. Before Java came out, programmers can hardly port programs between different platforms (Harold, 2015). It was an impossible idea for the programmers to run the same instruction on different hardware and operating systems. Java solves the problem that computer programs are not portable, and it saves time for the programmers and increases their efficiency.

Java promotes a great concept that solves the problem of the nonportable codes, and the concept is implemented through three components (Gilbert, 1995). First of all, Java is a programming language that supports its customers to write codes in multiple platforms (devices). What’s more, Java has a famous slogan for the idea of being a portable programming language called WORE (Write Once Run Everywhere), and this is one of many key concepts in Java (Gilbert, 1995). Although this concept saved a lot of time for the programmers, the implementation is not sophisticated at all. In simple terms, a device will compile (translate) and run the Java codes as long as Java Virtual Machine (JVM), Java Runtime Environment (JRE), and Java Development Kit (JDK) are installed onto it. However, these three components all have different purposes. The JVM is where the codes, which are wrote by the programmers, are being executed and compiled (translated); hence it is the most important part that is necessary for the operation of programs. According to Lindholm and Yellin (1997), after the JVM received the bytecode files and being asked to operate, it will then start to run these files and follow the instructions wrote in the codes. Imagine the JVM as a machine box, then JRE would be a fancy packing box encased JVM since JVM is just a section of JRE (Rouse, 2005). JRE contains all kinds of key classes (which can be seen as the helper functions) that are essential for the operation of Java programs (Jaschob & Riffle, 2012; Rouse, 2005). For most of the time, Java applications will be able to run with only JRE installed on the device. However, if JRE and JVM are two boxes then JDK would be the outmost toolbox that accommodates both of them (Grehan, 1997). Although sometimes Java codes will be able to run with only the JRE package, JDK is the complete form of Java. The JDK includes all the tools and functions a programmer will ever need to use. All in a nutshell, the concept WORE made Java a portable programming language, and the JVM is where the codes are being translated to different languages for different platforms (devices).

The portability of Java provides the programmers many benefits. Portability means the computer applications can be used in platforms (devices), which are not where the applications was made, without rewrite the codes (Rouse, 2005). Since less work is needed to be done in the processes of duplicating programs for some of the main operating systems (e.g., Windows, MAC OS, iOS), it will save a lot of time and energy for the programmers and their companies. As Curtin (1998) said, “you can buy the program without having to worry whether it’s going to run on your computer”, this is because that the program you bought will always be able to run on all the main platforms (devices). One great example of a successful product made by Java would be Minecraft (Duncan, 2011). Minecraft is a game written in Java, and it could be played on all kinds of platforms like Macs, Windows, Android and iOS devices. According to the data provided by Duncan (2011), there are over 11 million individual users that had already been registered a game account for Minecraft. These benefits of the portability of Java were firm due to the result brought by the success of Minecraft.

Although the idea of WORE is fantastic, some problems come with it as well (Gilbert, 1995). First problem is that if the codes were written in higher versions of Java while the clients only have some lower versions of JVM, the codes will not be able to run. Second problem is that the positions of some components in an app might change due to different platforms. Third problem is that the performance of Java is relatively slower than similar applications written in C and C++, which is largely because of the unnecessary occupation of memory (the more idle memory there is, the more efficient devices can be) in the devices (Gilbert, 1995). On the other hand, there are some inefficiency in the running of Java. Before the operating of the program, a virtual pass of checking is required, which result in a delay in the running (Lindholm & Yellin, 1997). However, that problem was only been token into consideration in the days that virtual machines cannot performance as required (Harold, 2015).

In conclusion, Java helps programmers work more efficiently by the idea of WORE, which achieved by the cooperating of JRE, JVM and JDK. Even though Java has some limitations, it is a revolutionary computer language that solve an essential problem in the programming field. It would be better to use java if the program need to be developed by more than one platform.

# References

Borgerand, E., & Schulte, W. (1999). A Programmer Friendly Modular Denition of the Semantics of Java. *Formal Syntax and Semantics of Java*, p.357-408.

Chapman, S. J. (1999). *Java for Engineers and Scientists.* NJ, USA: Prentice-Hall, Inc. .

Curtin, M. (1998). Write once, run anywhere: why it matters.

Duncan, S. (2011). Minecraft, Beyond Construction and Survival. *Well Played*, p. 1-22.

Gilbert, S. (1995). *What About Java?* Retrieved from Orange Coast College: http://faculty.orangecoastcollege.edu/sgilbert/book/01-4-WhatAboutJava/index.html

Grehan, R. (1997). The latest release of the Java Development Kit, version 1.1, charts the future of the language. Java Development Kit, Take Two. *BYTE*, Vol. 22, No. 4.

Hansen, P. B. (1973). *Operating System Principles.* NJ, USA: Prentice-hall, Inc.

Harold, E. R. (2015, May 21st). *Java at 20: How it changed programming forever*. Retrieved from Info World: https://www.infoworld.com/article/2923773/java/java-at-20-how-java-changed-programming-forever.html

Jaschob, D., & Riffle, M. (2012). JobCenter: an open source, cross-platform, and distributed job queue management system optimized for scalability and versatility. *Source Code for Biology and Medicine*, p. 8.

Kim, L. (2015, Jun 1st). *10 Most Popular Programming Languages Today*. Retrieved from Inc.com: https://www.inc.com/larry-kim/10-most-popular-programming-languages-today.html

Lindholm, T., & Yellin, F. (1997). *Inside the Java Virtual Machine.* Reading, MA: Sun Microsystems Inc.

Mcghan, H. (1998). MicroJava chip set emerges, bit by bit. *Electronic Engineering Times*, p.96.

Rouse, M. (2005, Sep). *definition of Java Runtime Environment JRE*. Retrieved from The Server Side: http://www.theserverside.com/definition/Java-Runtime-Environment-JRE

Rouse, M. (2005, Apr). *Definition of protability*. Retrieved from Search Software Quality: http://searchstorage.techtarget.com/definition/portability

Rouse, M. (2007, Mar). *definition of program*. Retrieved from Search Software Quality: http://searchsoftwarequality.techtarget.com/definition/program

Silberschatz, A., Galvin, P., & Gagne, G. (2013). *Operating System Concepts Essentials, 2nd Edition.* Hoboken, NJ: John Wiley & Sons, Inc.