Java Standard Edition 8

Sandeep Kulange
sandeepkulange@sunbeaminfo.com

Java Buzzwords

1. Simple
2. Object Oriented
3. Architecture Neutral
4. Portable
5. Robust
6. Multithreaded
7. Dynamic
8. Secure
9. High Performance
10. Distributed

Simple

- Java is **simple** programming language.
    - **Syntax of Java is simpler than syntax of C/C++ hence it is considered as simple.**
        - No need of header files and macros.
        - We can not define anything global
        - Do not support structure and union.
        - Do not support operator overloading.
        - Do not support copy constructor and assignment operator function
        - Do not support constructor member initializer list and default argument
        - Do not support constant data member and constant member function.
        - Do not support delete operator and destructor.
        - Do not support friend function and friend class.
        - Do not support multiple class(multiple implementation) inheritance.
        - Do not support private and protected mode of inheritance.
        - No diamond problem and virtual base class.
        - Do not support pointer and pointer arithmetic.
    - **Size of software(JDK), that is required to develop Java application is small hence Java is considered as simple programming language.**

Object Oriented

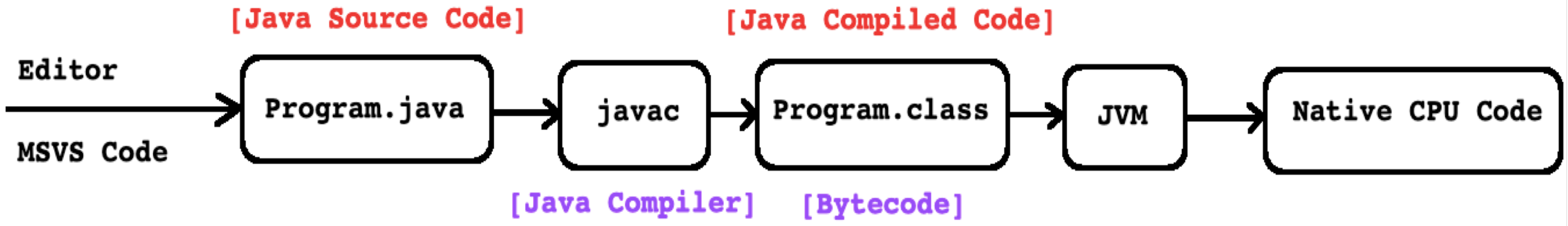
- Java is **object oriented** programming language.
    - Java Supports all the major and minor pillars of oops hence it is considered as object oriented programming language.

    - **Major pillars of oops.**
        1. Abstraction
        2. Encapsulation
        3. Modularity
        4. Hierarchy

    - **Minor pillars of oops.**
        1. Typing / Polymorphism
        2. Concurrency
        3. Persistence.

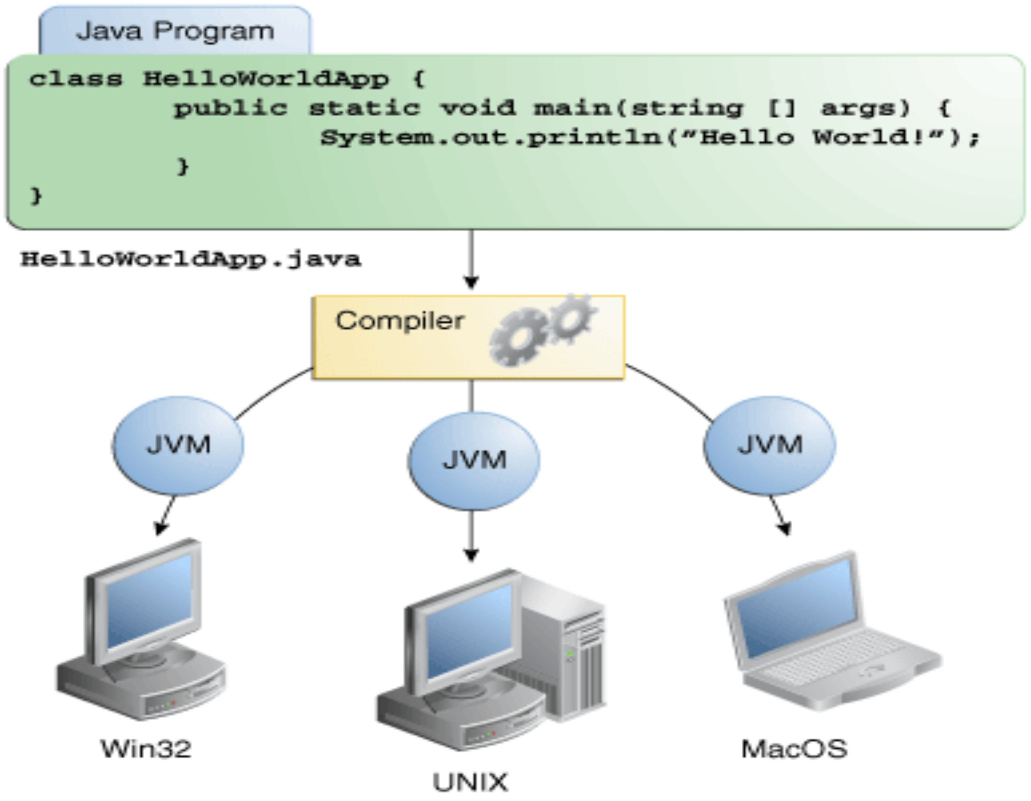
Architecture Neutral

- Java is object **architecture neutral** programming language.
    - Java technology is designed to support applications that will be deployed into heterogeneous network environments. In such environments, applications must be capable of executing on a variety of hardware architectures. Within this variety of hardware platforms, applications must execute on the top of a variety of operating systems. To accommodate the diversity of operating environments, the Java Compiler product generates *bytecodes*--an *architecture neutral* intermediate format designed to transport code efficiently to multiple hardware and software platforms.



Portable

- Java is **portable** programming language.
    - Architecture neutrality is just one part of a truly *portable* system.

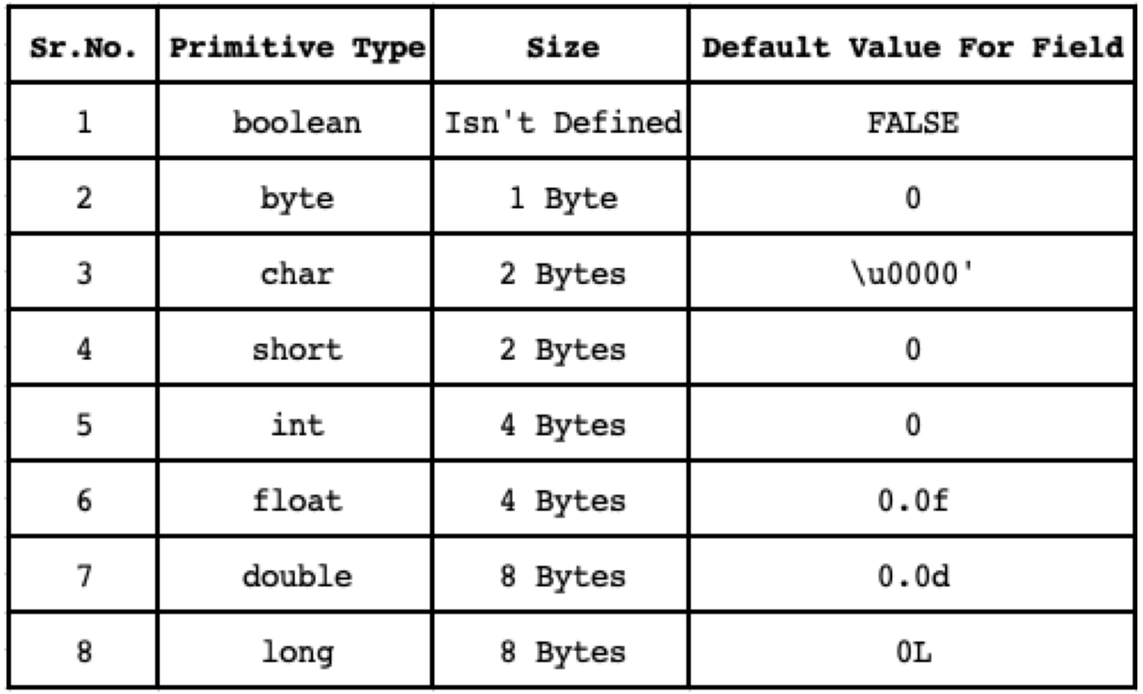


```
class HelloWorldApp {
        public static void main(string [] args) {
                System.out.println("Hello World!");
        }
}
```

-
    - Java's slogan is "**Write Once Run Anywhere(WORA)**".

Portable

- Java is **portable** programming language.
    - Java technology takes portability a stage further by being strict in its definition of the basic language.
    - Java technology puts a stake in the ground and specifies the sizes of its basic data types and the behavior of its arithmetic operators.
    - Your programs are the same on every platform--there are no data type incompatibilities across hardware and software architectures.



Sr.No.	**Primitive Type**	**Size**	**Default Value For Field**
1	boolean	Isn't Defined	FALSE
2	byte	1 Byte	0
3	char	2 Bytes	\u0000'
4	short	2 Bytes	0
5	int	4 Bytes	0
6	float	4 Bytes	0.0f
7	double	8 Bytes	0.0d
8	long	8 Bytes	0L

Robust

- Java is **robust** programming language.
    - The Java programming language is designed for creating highly *reliable* software. It provides extensive compile-time checking, followed by a second level of run-time checking. Language features guide programmers towards reliable programming habits.
    - Java is robust because of following features:
        1. *Architecture Neutral.*
            - ➢ Java developer is free from developing H/W or OS specific coding.
        2. *Object orientation.*
            - ➢ Reusability reduces developer's effort.
        3. *Automatic memory management.*
            - ➢ Developer need not to worry about memory leakage / program crashes.
        4. *Exception handling.*
            - ➢ Java compiler helps developer to provide try-catch block.

Multithreaded

- Java is **multithreaded** programming language.
  - When we start execution of Java application then JVM starts execution of two threads hence every Java is considered as multithreaded.
    1. Main thread
       - ➢ It is user thread / non daemon thread.
       - ➢ It is responsible for invoking main method.
       - ➢ Its default priority is 5(Thread.NORM_PRIORITY).
    2. Garbage Collector / Finalizer
       - ➢ It is daemon thread / background thread.
       - ➢ It is responsible for releasing / deallocating memory of unused objects.
       - ➢ Its default priority is 8(Thread.NORM_PRIORITY + 3).

  - The Java platform supports multithreading at the language level with the addition of sophisticated synchronization primitives: the language library provides the Thread class, and the run-time system provides monitor and condition lock primitives. At the library level, moreover, Java technology's high-level system libraries have been written to be thread safe: the functionality provided by the libraries is available without conflict to multiple concurrent threads of execution.

Dynamic

- Java is **dynamic** programming language.
    - While the Java Compiler is strict in its compile-time static checking, the language and run-time system are *dynamic* in their linking stages. Classes are linked only as needed. New code modules can be linked in on demand from a variety of sources, even from sources across a network.
    - Java is designed to adapt to an evolving environment.
    - Libraries can freely add new methods and instance variables without any effect on their clients.
    - In Java finding out runtime type information is straightforward.
    - In Java, all the methods are by default virtual.

Secure

- Java is **secure** programming language.
    - Java is intended to be used in networked/distributed environments. Toward that end, a lot of emphasis has been placed on security. Java enables the construction of virus-free, tamper-free systems.
    - From the beginning, Java was designed to make certain kinds of attacks impossible, among them:
        1. Overrunning the runtime stack—a common attack of worms and viruses
        2. Corrupting memory outside its own process space
        3. Reading or writing files without permission

    - For more details : https://www.artima.com/insidejvm/ed2/security.html

High Performance

- Java is **high performance** programming language.
    - The Java platform achieves superior performance by adopting a scheme by which the interpreter can run at full speed without needing to check the run-time environment.
    - The *automatic garbage collector* runs as a low-priority background thread, ensuring a high probability that memory is available when required, leading to better performance.
    - Applications requiring large amounts of compute power can be designed such that compute-intensive sections can be rewritten in native machine code as required and interfaced with the Java platform.
    - In general, users perceive that interactive applications respond quickly even though they're interpreted.

Distributed

- Java is **distributed** programming language.
    - Java has an extensive library of routines for coping with TCP/IP protocols like HTTP and FTP.
    - Java applications can open and access objects across the Net via URLs with the same ease as when accessing a local file system.
    - Nowadays, one takes this for granted, but in 1995, connecting to a web server from a C++ or Visual Basic program was a major undertaking.

Thank you