CSC 110 2.0 Object Oriented Programming

Tute 01 - Introduction to Java Prepared by: Dulan S. Dias

-

01. Compare and contrast OOP and structured/ procedural programming.

Object Oriented Programming	Structured/ Procedural Programming
Follows bottom-up approach to program design	Follows top-down approach to program design
Data and functions are coupled together	Data and functions are not coupled together, instead is accessible throughout the program
Programs are divided into smaller entities called Classes, to create instances called Objects	Larger programs are divided into smaller code blocks called functions
Access to data within classes, functions and outside world can be controlled using access modifiers	Data is openly available throughout the system from function to function
Functions and Classes are not dependent, hence, reusability is possible	Reusability is not possible since functions are highly dependent on data and each other

02. Features of Java

a. OOP
Java is an object-oriented programming language. In Java, everything is an Object. Java can be easily extended since it is based on the Object model. Object-oriented means we organize our software as a combination of different types of objects that incorporates both data and behavior. Object-oriented programming (OOPs) is a methodology that simplifies software development and maintenance by providing some rules and guidelines.

b. Simple and Familiar
Java is designed to be easy to learn. If you understand the basic concept of OOP Java, it would be easy to master. Since Java is based on OOP concepts which are much similar to most real world scenarios, Java will make you feel familiar.

c. Portable - Java Bytecode, JVM and Platform Independency
Being architecture-neutral and having no implementation dependent aspects of the specification makes Java portable.
Unlike many other programming languages including C and C++, when Java is compiled, it is not compiled into platform specific machine, rather into platform-independent bytecode. This bytecode is distributed over the web and interpreted by the Virtual Machine (JVM) on whichever platform it is being run on.

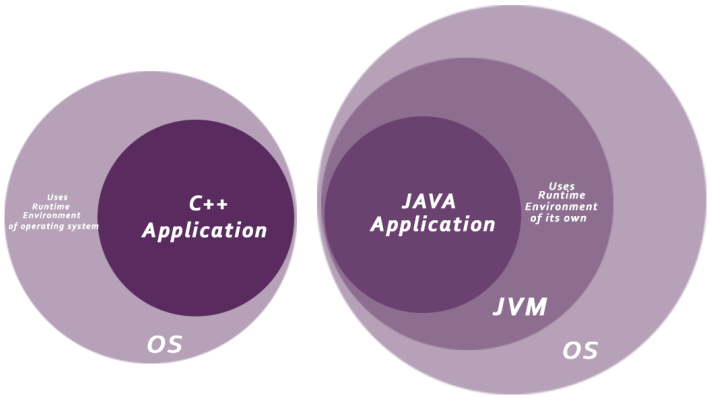
d. Interpreted

Java bytecode is translated on the fly to native machine instructions and is not stored anywhere. The development process is more rapid and analytical since the linking is an incremental and light-weight process.

e. Dynamic - dynamic compilation and automatic memory management (garbage collection)
Java is a dynamic language. It supports dynamic loading of classes. It means classes are loaded on demand.
Java Memory Management, with its built-in garbage collection, is one of the language's finest achievements. It allows developers to create new objects without worrying explicitly about memory allocation and deallocation, because the garbage collector automatically reclaims memory for reuse. This enables faster development with less boilerplate code, while eliminating memory leaks and other memory-related problems.

f. Secure
With Java's secure feature it enables to develop virus-free, tamper-free systems. Java is secured because:

- No explicit pointer
- Java Programs run inside a virtual machine sandbox



g. Multithreading and concurrency
With Java's multithreaded feature it is possible to write programs that can perform many tasks simultaneously. This design feature allows the developers to construct interactive applications that can run smoothly.

h. High performance
With the use of Just-In-Time compilers, Java enables high performance.

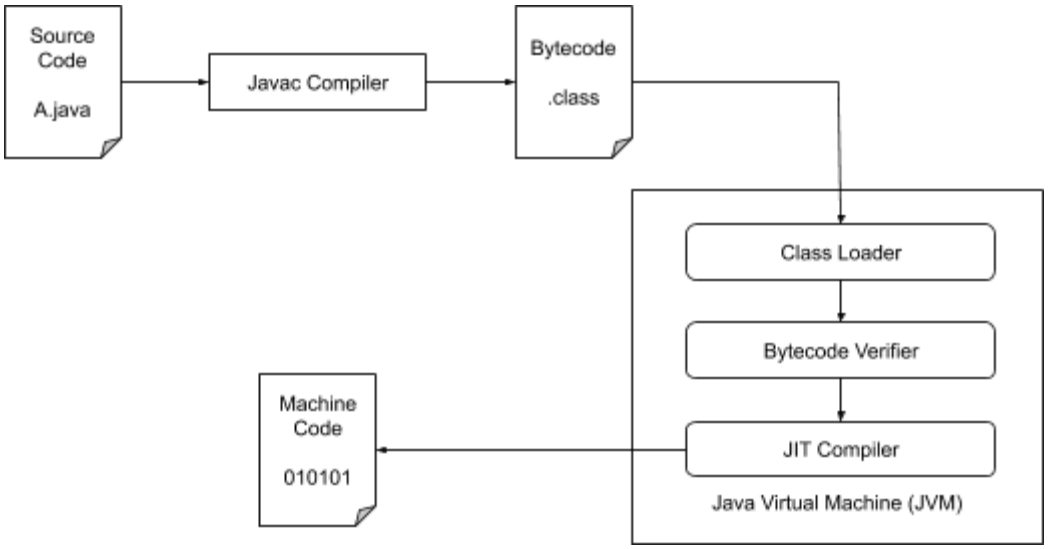
03. Component of Java

a. Class Loader
Classloader in Java is a part of the Java Runtime Environment (JRE) which is used to load Java classes into the Java Virtual Machine dynamically. It adds security by separating the package for the classes of the local file system from those that are imported from external sources.

b. Bytecode Verifier
It checks the code fragments for illegal code that can violate access rights to objects. After the bytecode of a class is loaded by the class loader, it has to be inspected by the bytecode verifier, whose job is to check that the instructions don't perform damaging actions.

c. Security Manager
It determines what resources a class can access such as reading and writing to the local disk.

04. Life Cycle of a Java Program



* * * * * * * *