Programming in Java

Introduction

Mahesh Kumar
(maheshkumar@andc.du.ac.in)

Course Web Page
(www.mkbhandari.com/mkwiki)

Outline

1. History of Java
2. Java Features
3. First Sample Program
4. Java Virtual Machine(JVM) Architecture
5. Difference between JDK, JRE, and JVM [Home Assignment 1]

Why a new Programming Language is Developed?

- Computer language innovation and development occurs for two fundamental reasons:
  - To adapt to changing environments and uses.
  - To implement refinements and improvements in the art of programming.
- The development of Java was driven by both elements in nearly equal measure.

Java's Lineage

- Java is related to C++, which is a direct descendant of C. Much of the character of Java is inherited from these two languages.
  - From C, Java derives its syntax.
  - Many of Java's object-oriented features were influenced by C++.
  - In fact, several of Java's defining characteristics come from—or are responses to—its predecessors.
- Moreover, the creation of Java was deeply rooted in the process of refinement and adaptation that has been occurring in computer programming languages for the past several decades.
- Each innovation in language design is driven by the need to solve a fundamental problem that the preceding languages could not solve.

The Birth of Modern Programming: C

- Prior to C:
  - Programming Languages such as BASIC, FORTRAN, and COBOL were relied upon the **GOTO** as a primary means of program control.
    - As a result, programs written using these languages tended to produce "spaghetti code" — a mass of tangled jumps and conditional branches that make a program virtually impossible to understand.
  - Although FORTRAN could be used to write fairly efficient programs for scientific applications, it was not very good for system code.
  - While BASIC was easy to learn, it wasn't very powerful, and its lack of structure made its usefulness questionable for large programs.
  - Assembly language can be used to produce highly efficient programs, but it is not easy to learn or use effectively. Further, debugging assembly code can be quite difficult.

The Birth of Modern Programming: C

- By the early 1970s:
  - The computer revolution was beginning to take hold, and the demand for software was rapidly outpacing programmers' ability to produce it.
  - A great deal of effort was being expended in academic circles in an attempt to create a better computer language.
  - Computer hardware was finally becoming common enough that a critical mass was being reached. No longer were computers kept behind locked doors.
  - For the first time, programmers were gaining virtually unlimited access to their machines. This allowed the freedom to experiment.
  - It also allowed programmers to begin to create their own tools.

The Birth of Modern Programming: C

- C was originally developed at Bell Labs by Dennis Ritchie between **1972 and 1973** to make utilities running on Unix.
- C was the result of a development process that started with an older language:
  - BCPL, developed by Martin Richards in **1967**.
  - BCPL influenced a language called B, invented by Ken Thompson in **1969**, which led to the development of C in the 1970s.
  - For many years, the de facto standard for C was the one supplied with the UNIX operating system
  - C was formally standardized in December 1989, when the American National Standards Institute (ANSI) standard for C was adopted.
  - C was a powerful, efficient, **structured language** that was relatively easy to learn.
  - Prior to the invention of C, computer languages were generally designed either as academic exercises or by bureaucratic committees.
  - C was designed, implemented, and developed by real, working programmers.

C++: The Next Step

- By the early 1980s, many projects were pushing the structured approach past its limits - once a project reaches a certain size, its complexity exceeds what a programmer can manage.
- To solve this problem, a new way to program was invented, called object-oriented programming (OOP).
- OOP is a programming methodology that helps organize complex programs through the use of:
  - Inheritance
  - Encapsulation
  - Polymorphism
- Although C is one of the world's great programming languages, there is a limit to its ability to handle complexity.
- C++ added features that enabled this threshold to be broken, allowing programmers to comprehend and manage larger programs.

C++: The Next Step

- C++ was invented by Bjarne Stroustrup in 1979, while he was working at Bell Laboratories in Murray Hill, New Jersey.
- Stroustrup initially called the new language "C with Classes." However, in 1983, the name was changed to C++.
- C++ extends C by adding object-oriented features.
- Because C++ is built on the foundation of C, it includes all of C's features, attributes,and benefits.
- The invention of C++ was not an attempt to create a completely new programming language. Instead, it was an enhancement to an already highly successful one.

The Creation of Java

- By the end of the 1980s and the early 1990s, object-oriented programming using C++ took hold.
- Indeed, for a brief moment it seemed as if programmers had finally found the perfect language.
- Because C++ blended the high efficiency and stylistic elements of C with the object-oriented paradigm, it was a language that could be used to create a wide range of programs.
- Within a few years, the World Wide Web and the Internet would reach critical mass.
- This event would precipitate another revolution in programming.

The Creation of Java

- By the end of the 1980s and the early 1990s, object-oriented programming using C++ took hold.
- Indeed, for a brief moment it seemed as if programmers had finally found the perfect language.
- Because C++ blended the high efficiency and stylistic elements of C with the object-oriented paradigm, it was a language that could be used to create a wide range of programs.
- Within a few years, the World Wide Web and the Internet would reach critical mass.
- This event would precipitate another revolution in programming.

The Creation of Java

- James Gosling (Father of Java), Mike Sheridan, and Patrick Naughton initiated the Java language project in June 1991. The small team of **sun** engineers called Green Team.
- Initially designed for small, embedded systems in electronic appliances like set-top boxes.
- Firstly, it was called "**Greentalk**" by James Gosling, and the file extension was .gt.
- After that, it was called **Oak** and was developed as a part of the Green project.
- Why Oak? Oak is a symbol of strength and chosen as a national tree of many countries like the U.S.A., France, Germany, Romania, etc.

- In 1995, Oak was renamed as **"Java"** because it was already a trademark by Oak Technologies.
- Java is a **programming language** and a **platform**. Java has a JRE and API.

The Creation of Java

- Java is an island of Indonesia where the first coffee was produced (called java coffee). It is a kind of espresso bean. Java name was chosen by James Gosling while having coffee near his office.
- Notice that Java is just a name, not an acronym.
- Initially developed by James Gosling at Sun Microsystems (which is now a subsidiary of Oracle Corporation) and released in 1995.
- In 1995, Time magazine called Java one of the Ten Best Products of 1995.
- JDK 1.0 released in(January 23, 1996). After the first release of Java, there have been many additional features added to the language.

The Creation of Java - Versions

- Many **Java versions** have been released till now. The current stable release of Java is Java SE 10.

1. JDK Alpha and Beta (1995)
2. JDK 1.0 (23rd Jan 1996)
3. JDK 1.1 (19th Feb 1997)
4. J2SE 1.2 (8th Dec 1998)
5. J2SE 1.3 (8th May 2000)
6. J2SE 1.4 (6th Feb 2002)
7. J2SE 5.0 (30th Sep 2004)
8. Java SE 6 (11th Dec 2006)
9. Java SE 7 (28th July 2011)
10. Java SE 8 (18th Mar 2014)
11. Java SE 9 (21st Sep 2017)
12. Java SE 10 (20th Mar 2018)

The Creation of Java - Applications

- According to Sun, 3 billion devices run Java. There are many devices where Java is currently used. Some of them are as follows:

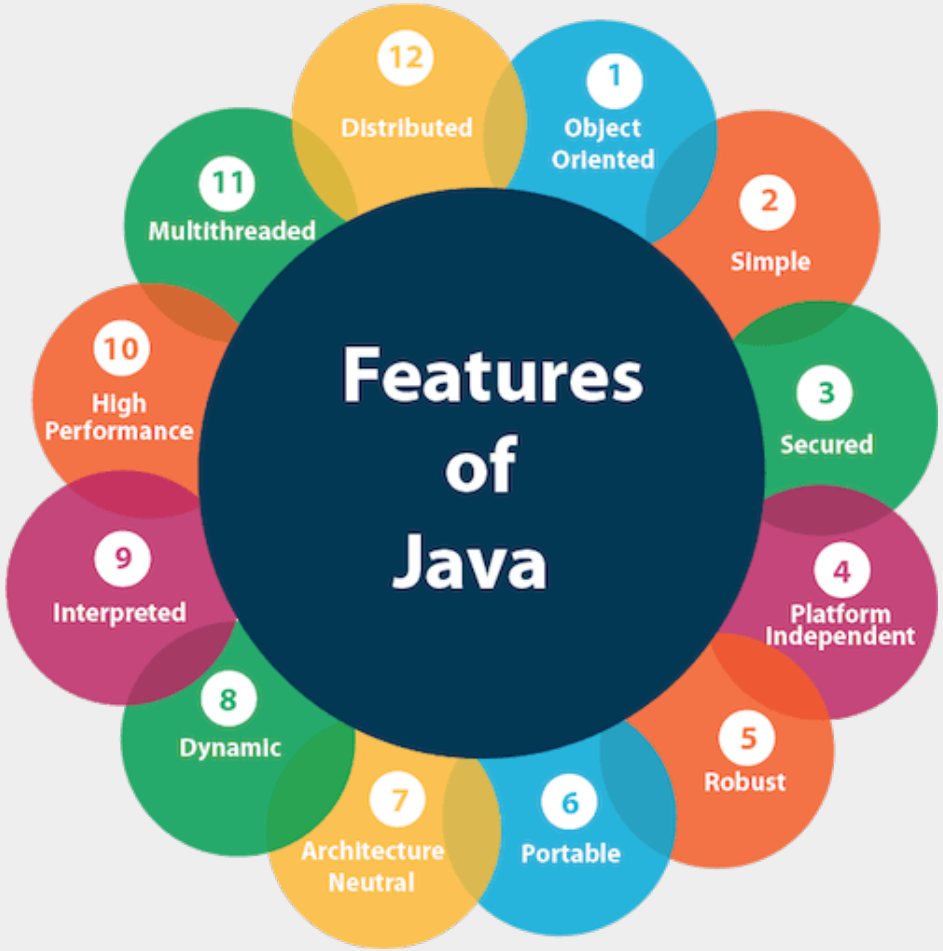
1. Desktop/Standalone/Window Applications such as acrobat reader, media player, antivirus, etc.
2. Web Applications such as irctc.co.in, javatpoint.com, etc.
3. Enterprise Applications such as banking applications.
4. Mobile
5. Embedded System
6. Smart Card
7. Robotics
8. Games

- AWT, Swings
- Servlet, JSP, Struts, Spring, Hibernate, JSF
- EJB
- Android and Java ME

The Creation of Java - Platforms/Editions

- There are 4 platforms or editions of Java:
  1. Java SE (Java Standard Edition)
     - It is a Java programming platform. It includes Java programming APIs such as java.lang, java.io, java.net, java.util, java.sql, java.math etc
     - It includes core topics like OOPs, String, Regex, Exception, Inner classes, Multithreading, I/O Stream, Networking, AWT, Swing, Reflection, Collection, etc.
  2. Java EE (Java Enterprise Edition)
     - It is an enterprise platform which is mainly used to develop web and enterprise applications. It is built on the top of the Java SE platform.
     - It includes topics like Servlet, JSP, Web Services, EJB, JPA, etc.
  3. Java ME (Java Micro Edition)
     - It is a micro platform which is mainly used to develop mobile applications.
  4. JavaFX
     - It is used to develop rich internet applications. It uses a light-weight user interface API.

Features of Java (The Java Buzzwords)



[Source: 2]

Features of Java (The Java Buzzwords)

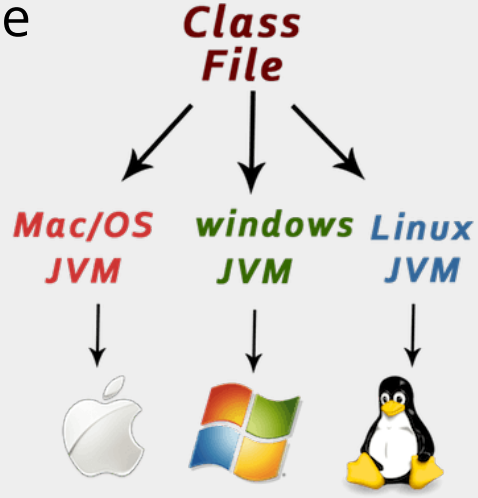
1. ***Object Oriented:*** Java is an object-oriented programming language.
   - Everything in Java is an object.
   - Object-oriented means we organize our software as a combination of different types of objects that incorporates both data and behavior.
   - OOPs is a methodology that simplifies software development and maintenance by providing some rules.
   - Basic concepts of OOPs are:
     - *Object*
     - *Class*
     - *Inheritance*
     - *Polymorphism*
     - *Abstraction*
     - *Encapsulation*

Features of Java (The Java Buzzwords)

2. ***Simple:*** Java is very easy to learn, and its syntax is simple, clean and easy to understand.
   - According to Sun, Java language is a simple programming language because:
     - *Java syntax is based on C/C++ (so easier for programmers to learn it after C++).*
     - *Java has removed many complicated and rarely-used features, for example, explicit pointers, operator overloading, etc.*
     - *There is no need to remove unreferenced objects because there is an Automatic Garbage Collection in Java.*

Features of Java (The Java Buzzwords)

3. ***Platform Independent:*** Java is a write once, run anywhere language.
   - Java code is compiled by the compiler and converted into bytecode (language like C, C++, which are compiled into platform specific machines)
   - The bytecode is a platform-independent code because it can be run on multiple platforms
   - There are two types of platforms software-based and hardware-based. Java provides a software-based platform that runs on the top of other hardware-based platforms.It has two components:
     - *JRE (Java Runtime Environment)*
     - *API (Application Programming interface)*



[Source: 2]

Features of Java (The Java Buzzwords)

4. ***Secured:*** Java is best known for its security. With Java, we can develop virus-free systems.
   - Java is secured because:
     - *No explicit pointer*
     - *Java Programs run inside a virtual machine sandbox (C/C++ uses runtime environment of OS)*
     - *Classloader: part of JRE, which load Java classes into the JVM dynamically. It adds security by separating the package for the classes of the local file system from those that are imported from network sources.*
     - *Bytecode Verifier: It checks the code fragments for illegal code that can violate access right to objects.*
     - *Security Manager: It determines what resources a class can access such as reading and writing to the local disk.*

Features of Java (The Java Buzzwords)

5. ***Robust:*** Robust simply means strong. Java is robust because:
   - There is automatic garbage collection in java which runs on the Java Virtual Machine to get rid of objects which are not being used by a Java application anymore (in C/C++, the programmer will often manually allocate and free all dynamic memory).
   - Exception handling: In a well-written Java program, all run-time errors can—and should—be managed by your program.
   - Because Java is a strictly typed language, it checks your code at compile time. However, it also checks your code at run time.
   - There is a lack of pointers that avoids security problems.

Features of Java (The Java Buzzwords)

6. ***Architecture-neutral:*** Java is architecture neutral because:
   - There are no implementation dependent features, for example, the size of primitive types is fixed.
     - In C programming, int data type occupies 2 bytes of memory for 32-bit architecture and 4 bytes of memory for 64-bit architecture.
     - However, it occupies 4 bytes of memory for both 32-bit and 64-bit architectures in Java.
   - Operating system upgrades, processor upgrades, and changes in core system resources can all combine to make a program malfunction.
   - The goal of Java designer was **"write once; run anywhere, any time, forever."** To a great extent, this goal was accomplished.

Features of Java (The Java Buzzwords)

7. ***Interpreted and High-performance:*** Java enables the creation of cross-platform programs by compiling into an intermediate representation called Java bytecode.
   - This code can be executed on any system that implements the Java Virtual Machine.
   - The Java bytecode is translated directly into native machine code for very high performance by using a just-in-time compiler.
   - Java run-time systems that provide this feature lose none of the benefits of the platform-independent code.

Features of Java (The Java Buzzwords)

8. ***Multithreaded:*** We can write Java programs that deal with many tasks at once by defining multiple threads.
   - A thread is like a separate program, executing concurrently.
   - The main advantage of multi-threading is that it doesn't occupy memory for each thread. It shares a common memory area.
   - Threads are important for multi-media, Web applications, etc.
   - Java was designed to meet the real-world requirement of creating interactive, networked programs.

Features of Java (The Java Buzzwords)

9. ***Distributed:*** Java is distributed because it facilitates users to create distributed applications in Java.

   - RMI and EJB are used for creating distributed applications.
   - This feature of Java makes us able to access files by calling the methods from any machine on the internet.
   - Java is designed for the distributed environment of the Internet because it handles TCP/IP protocols.

Features of Java (The Java Buzzwords)

10 ***Dynamic:*** Java is a dynamic language.

- It supports dynamic loading of classes. It means classes are loaded on demand.
- It also supports functions from its native languages, i.e., C and C++.
- Java supports dynamic compilation and automatic memory management (garbage collection).

The First Sample Program

```
1
// This is a simple Java Program
2   3
class HelloWorld{

        public static void main(String args[ ]) {

                System.out.println("Hello World");

        }

}
```

1. **Comment**
   - Describes how the program works or what a specific feature does.
   - The contents of a comment are ignored by the compiler.
2. **class**
   - The keyword class is used to define/create a new class.
3. **HelloWorld**
   - HelloWorld is an identifier that is the name of the class.
   - The entire class definition, including all of its members, will be between the opening curly brace ({) and the closing curly brace (}).
   - in Java, all program activity occurs within class.

The First Sample Program

```
1
// This is a simple Java Program
2   3
class HelloWorld{
        4   5   6
        public static void main(String args[ ]) {

                System.out.println("Hello World");

        }

}
```

4 **public**

- The keyword public is an access modifier (visibility specifier).
- Regulate access to classes, fields and methods in Java.

5 **static**

- The keyword static allows main( ) to be called without creating object of the class.
- This is necessary since main( ) is called by the Java Virtual Machine before any objects are made.

6 **void**

- The keyword void simply tells the compiler that main( ) does not return a value.

The First Sample Program

```
1
// This is a simple Java Program
2  3
class HelloWorld{
        4  5  6  7    8
        public static void main(String args[ ]) {
                9
                System.out.println("Hello World");

        }

}
```

7 main( )

- The main( ) method is the starting point for JVM to start execution of a Java program.
- Without the main() method, JVM will not execute the program.

8 String args[ ]

- The parameter args is used to receive any command-line arguments when the program is executed.

9 System.out

- System is a predefined class that provides access to the system
- out is the output stream that is connected to the console(terminal window).

The First Sample Program

```
1
// This is a simple Java Program
2  3
class HelloWorld{
        4  5  6  7    8
        public static void main(String args[ ]) {
                9          10
                System.out.println("Hello World");

        }

}
```

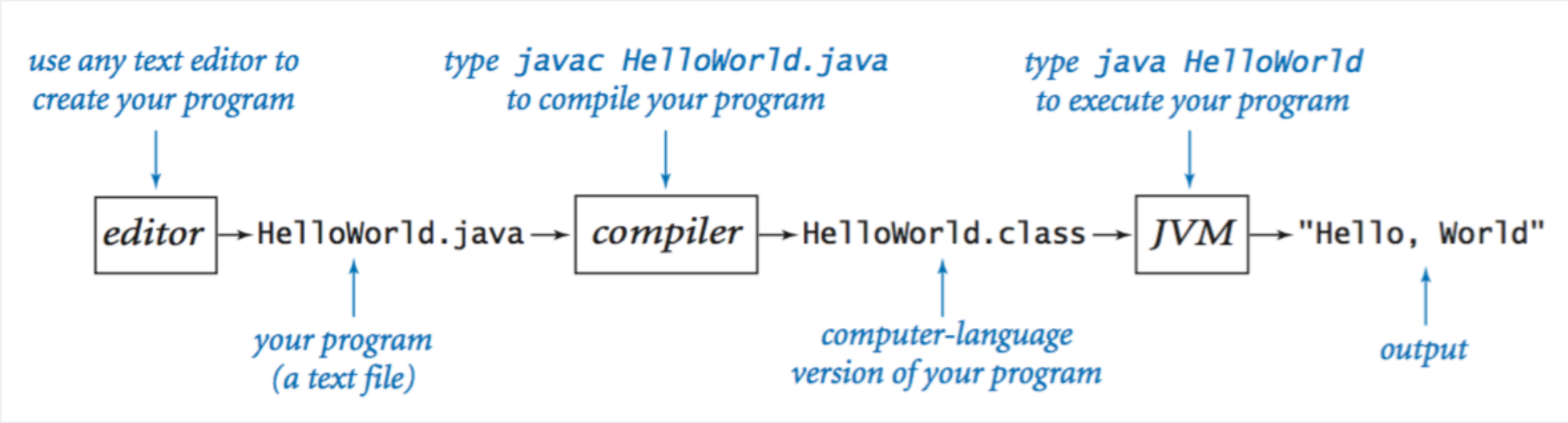
10 println( )

- The bulit-in println( ) method displays the string which is passed to it.

11 Other important information

- A complex program will have dozens of classes, only one of which will need to have a main( ) method to get things started.
- In some cases, you won't need main( ) at all. For example, when creating applets.
- All statements in Java end with a semicolon.

Editing, Compiling, and Executing.



[Source: 4]

Lexical Issues (Tokens)

- Java programs are a collection of whitespaces, identifiers, literals, comments, operators, separators, and keywords.

1. **Whitespace**
   - whitespace is a space, tab, or newline. Java is a free-form language.
2. **Identifiers**
   - Identifiers are used to name things, such as classes, variables, and methods. Java is case-sensitive, so Name is a different identifier than name.
3. **Literals**
   - A constant value in Java is created by using a literal representation of it.
4. **Comments**
   - Three types of comments defined by Java (Single-line, Multiline, Documentation).
5. **Separators**
   - There are 7 types of separatrors(**( )**, **{ }**, **[ ]**, **;** , **,** , **.** , **::**)
6. **Keywords**
   - There are 51 keywords currently defined in the Java language(49 are in use)

Separators

Symbol	Name	Purpose
( )	Parentheses	Used to contain lists of parameters in method definition and invocation. Also used for defining precedence in expressions, containing expressions in control statements, and surrounding cast types.
{ }	Braces	Used to contain the values of automatically initialized arrays. Also used to define a block of code, for classes, methods, and local scopes.
[ ]	Brackets	Used to declare array types. Also used when dereferencing array values.
;	Semicolon	Terminates statements.
,	Comma	Separates consecutive identifiers in a variable declaration. Also used to chain statements together inside a **for** statement.
.	Period	Used to separate package names from subpackages and classes. Also used to separate a variable or method from a reference variable.
::	Colons	Used to create a method or constructor reference. (Added by JDK 8.)

Keywords

abstract	continue	for	new	switch
assert	default	goto	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp	volatile
const	float	native	super	while

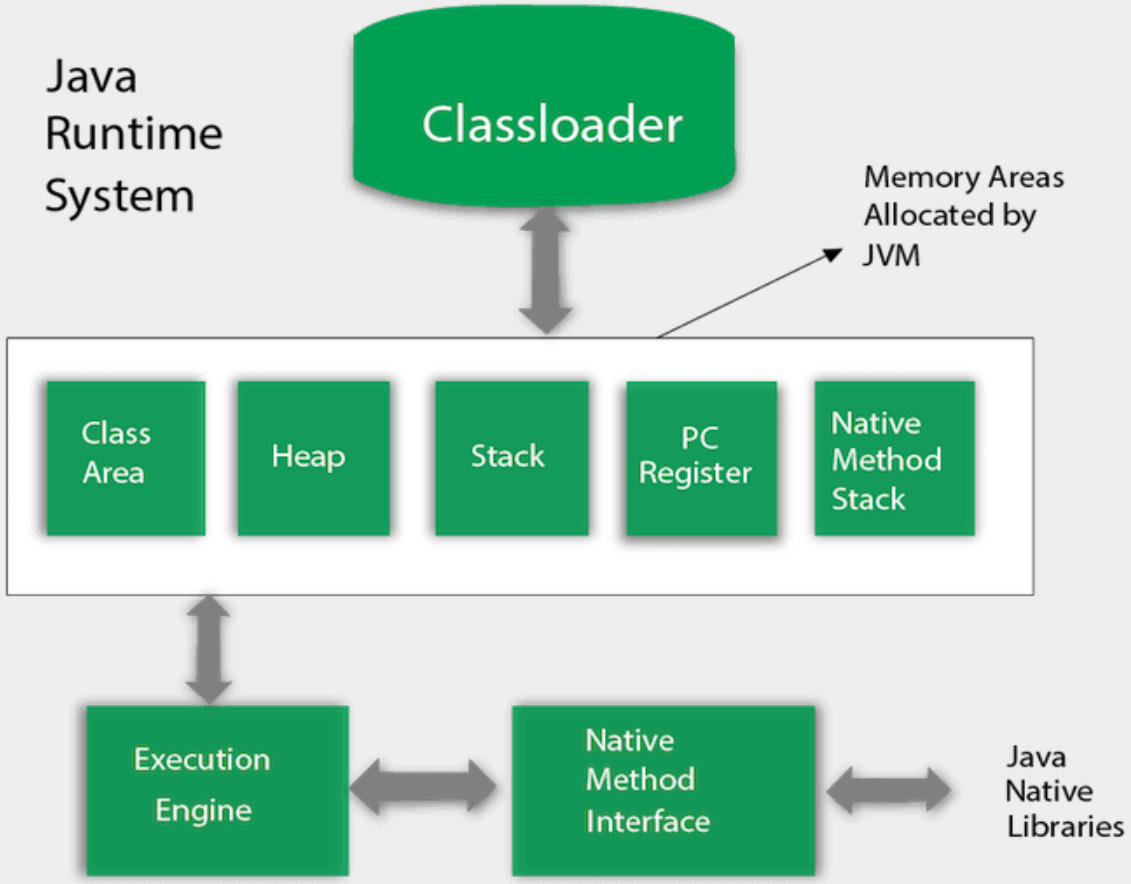
JVM Architecture

- JVM (Java Virtual Machine) is an abstract machine. It is a specification that provides runtime environment in which java bytecode can be executed.
- JVMs are available for many hardware and software platforms (i.e. JVM is platform dependent).
- What is JVM?
  1. **A specification:** where working of Java Virtual Machine is specified. Its implementation has been provided by Oracle and other companies.
  2. **An implementation:** Its implementation is known as JRE (Java Runtime Environment).
  3. **Runtime Instance:** Whenever you write java command on the command prompt to run the java class, an instance of JVM is created.

JVM Architecture

- The JVM performs following operation:
  - Loads code
  - Verifies code
  - Executes code
  - Provides runtime environment
- JVM provides definitions for the:
  - Memory Area
  - Class file format
  - Register Set
  - Garbage-collected heap
  - Fatal error reporting etc.

JVM Architecture



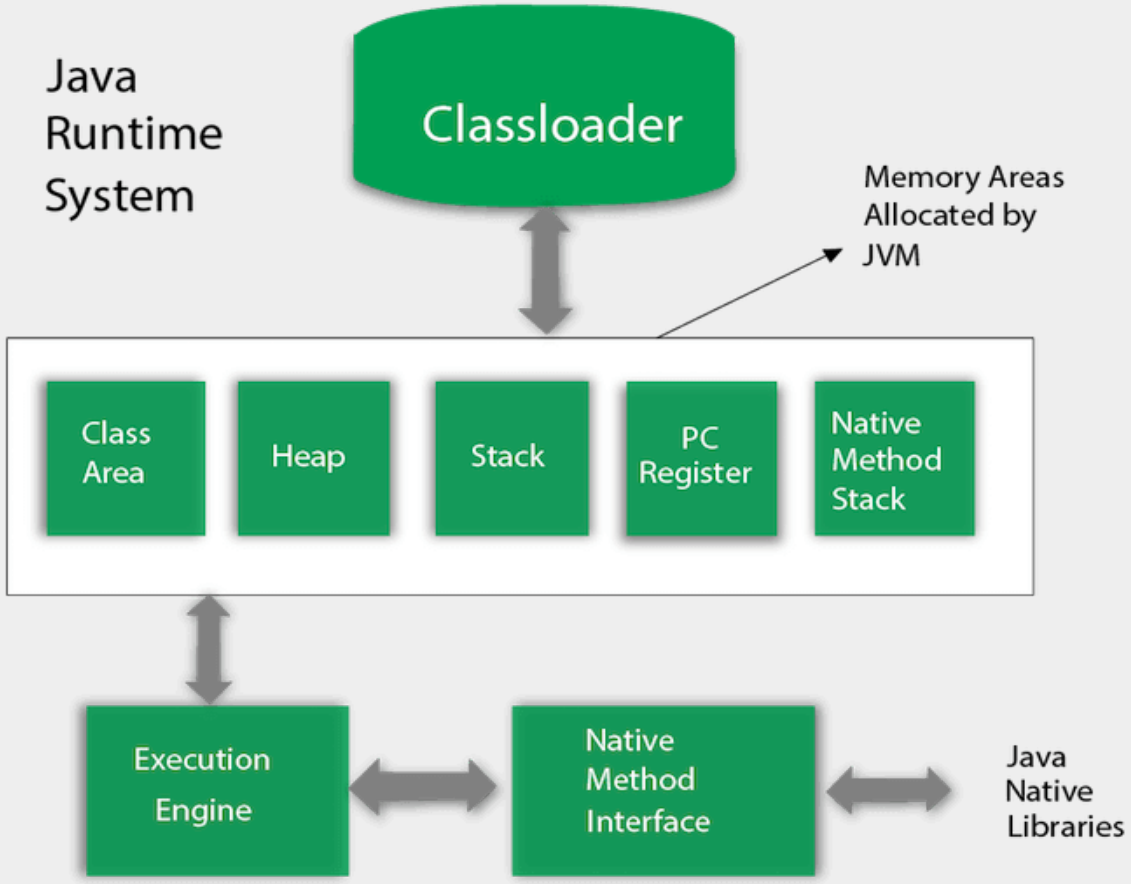
[Source: 2]

1. **Classloader**
   - Classloader is a subsystem of JVM which is used to load class files.
   - Whenever we run the java program, it is loaded first by the classloader.

2. **Class(Method) Area**
   - Class(Method) Area stores per-class structures such as the runtime constant pool, field and method data, the code for methods

3. **Heap**
   - It is the runtime data area in which objects are allocated.

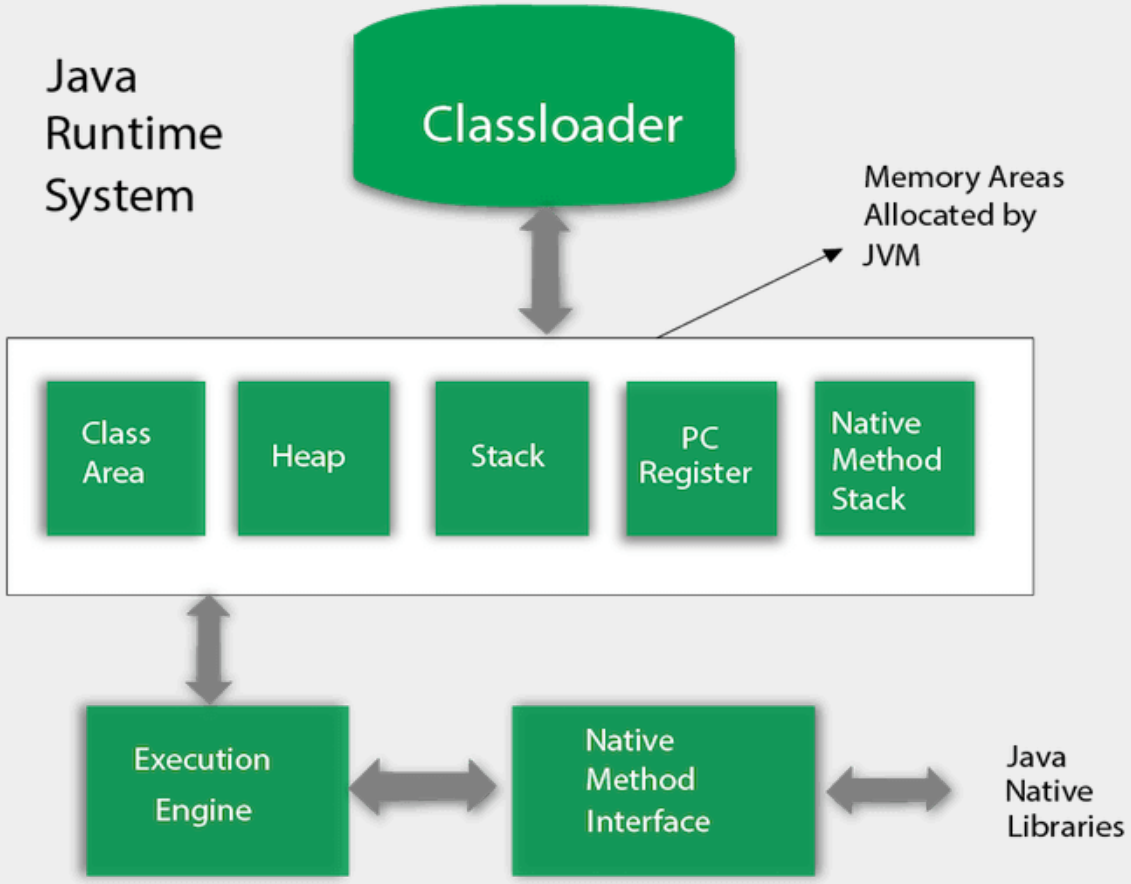
JVM Architecture



[Source: 2]

4. **Stack**
   - It holds local variables and partial results, and plays a part in method invocation and return.
   - A new frame is created each time a method is invoked. A frame is destroyed when its method invocation completes.
5. **PC Register**
   - PC (program counter) register contains the address of the Java virtual machine instruction currently being executed.
6. **Native Method Stack**
   - It contains all the native methods used in the application.

JVM Architecture



[Source: 2]

7. **Execution Engine** contains:
   - A virtual processor
   - Interpreter: Read bytecode stream then execute the instructions.
   - Just-In-Time(**JIT**) compiler: helps improve the performance of Java programs by compiling bytecodes into native machine code at run time.

8. **Java Native Interface**
   - Java Native Interface (JNI) is a framework which provides an interface to communicate with application written in C, C++, Assembly etc. Java uses JNI framework to send output to the Console or interact with OS libraries.

References

Reference for this topic

- [Book] Java: The Complete Reference, Tenth Edition: Herbert Schildt

- [Web] Java T Point tutorial
  https://www.javatpoint.com/java-tutorial

- [Web] GeeksforGeeks
  https://www.geeksforgeeks.org/java/

- [Web] Java Programming Cheatsheet (cs.princeton.edu)
  https://introcs.cs.princeton.edu/java/11cheatsheet/