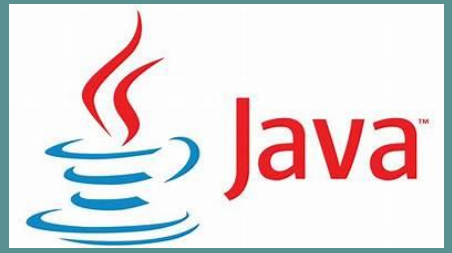


About Java

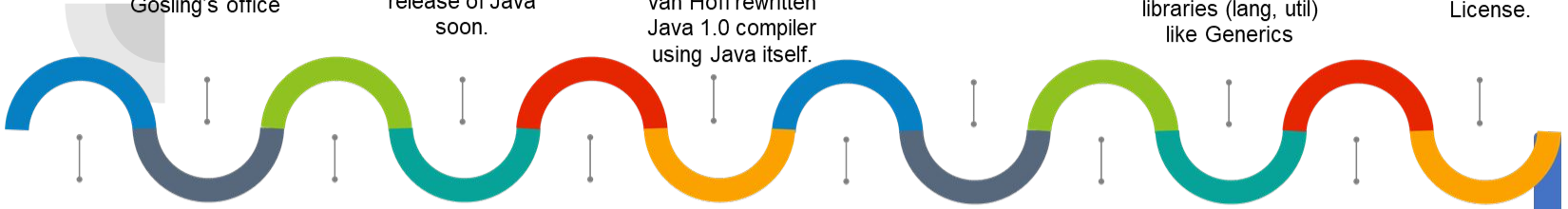
Prof. Madonna Lamin
ITMBU, SoCSET

Inventors of Java

- Java is one of the most popular programming languages worldwide. It was created by **James Gosling** and **Patrick Naughton**, employees of Sun Microsystems, with support from **Bill Joy**, co-founder of Sun Microsystems.
- Sun officially presented the Java language at SunWorld on **May 23, 1995**. Then, in **2009, the Oracle company bought the Sun company**, which explains why the language now belongs to Oracle.

What is Java?

- Java is described as being a multi-purpose, strongly typed, and Object-Oriented Programming (OOP) language. By design, it has as few implementation dependencies as possible.
- With this programming language, we can create applications on **multiple devices**. Its scope is very wide, allowing us to create software for mobile devices, point of sale terminals, ATMs, IoT (Internet of Things), as well as web pages.
- **Although Kotlin is the preferred language for Android development in 2021, Java is still the default language for developing apps for Android.** The big reason for that is Android's tight integration with Java. In fact, parts of that OS, like the UI and specific core libraries, are written in Java.



Jun 1991
Initially named as **Oak** when they discussed under Oak tree near Gosling's office

1995
JavaSoft was taken over by Sun Microsystems and announced the release of Java soon.

Feb, 1997
First stable release of Java as Dev Kit (1.1) released. Arthur van Hoff rewritten Java 1.0 compiler using Java itself.

2000
Java 1.3 released with JVM, JNDI with code name KESTRAL

2004
Java 1.5 (J2SE 1.5) released with most of standard features in core libraries (lang, util) like Generics

Nov, 2006
JVM officially released as free/OpenSource under GNU GPL License.

Jun 1991
Gosling, Sheridan and Naughton started development of new programming language with the name 'Oak'

Jul 1991
Later it was named as **Green** and then as **Java** to remind their favourite Java Coffee from Indonesia.

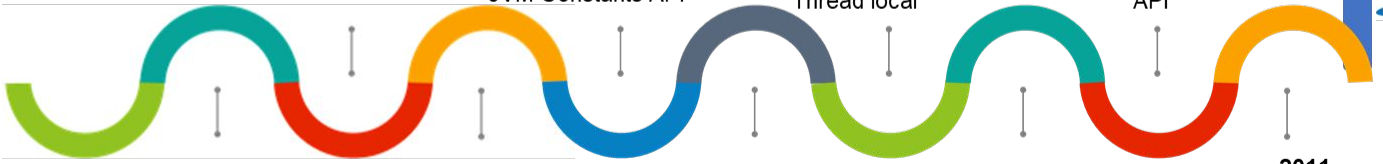
Jan 23, 1996
Sun Microsystems released first official Java 1.0

1998
Java 1.2 released as J2SE along with Swing graphical API library with codename **PlayArea**

2002
Java 1.4 with Regexp release with codename **Merlin**

2006
Multiple variants of Java released as J2SE, J2ME, J2EE with codename **Mustang**

Java became multi-faceted releases with from Jrocket, IBM, OpenJDK



March 2020
J2SE 14 released with Incubator, JFR Event Streaming, Java Records

March, 2019
J2SE 12 released with Shenandoah GC, Switch Expressions, JVM Constants API

March, 2018
J2SE 10 released with memory features like Heap allocation and Thread local

2014
J2SE 8 released with Major changes like Lambda and datetime API

Sept 2020
J2SE 15 released with Z GC, preview of sealed class, hidden classes.

Sept, 2019
J2SE 13 released with Text Blocks, Improved File IO API, Socket API

Sept, 2018
J2SE 11 released with Epsilon GC, Heap profiling and improved features of Lambda.

2017
J2SE 9 released with codename **Jigsaw** with features like Streaming API, JShell, improved features in Collections API

2011
J2SE 7 released with codename **Dolphin** with dynamic language, string optimization and GC optimization

Features of Java

- Platform Independence
- Simple
- Object-Oriented
- Robust
- Distributed
- Portable
- Dynamic
- Secure
- Performance
- Multithreaded
- Interpreted
- Architecture-Neutral

Platform Independence

The concept of Write-once-run-anywhere (known as the Platform independent) is one of the important key feature of java language that makes java as the most powerful language. Not even a single language is idle to this feature but java is more closer to this feature. The programs written on one platform can run on any platform provided the platform must have the JVM.

Simple

There are various features that makes the Java as a simple language.

Programs are easy to write and debug because **Java does not use the pointers explicitly**. It is much harder to write the Java programs that can crash the system but we can not say about the other programming languages. Java provides the bug free system due to the strong memory management. It also has the **automatic memory allocation and deallocation system**.

Object-Oriented

To be an Object Oriented language, any language must follow at least the four characteristics.

- **Inheritance** : It is the process of creating the new classes and using the behavior of the existing classes by extending them just to reuse the existing code and adding the additional features as needed.
- **Encapsulation**: It is the mechanism of combining the information and providing the abstraction.
- **Polymorphism**: As the name suggest one name multiple form, Polymorphism is the way of providing the different functionality by the functions having the same name based on the signatures of the methods.
- **Dynamic binding** : Sometimes we don't have the knowledge of objects about their specific types while writing our code. It is the way of providing the maximum functionality to a program about the specific type at runtime.

As the languages like Objective C, C++ fulfills the above four characteristics yet they are not fully object oriented languages because they are structured as well as object oriented languages. But in case of java, it is a fully Object Oriented language because object is at the outermost level of data structure in java. No stand alone methods, constants, and variables are there in Java. Everything in java is object even the primitive data types can also be converted into object by using the **wrapper class**.

Robust

Java has the **strong memory allocation and automatic garbage collection mechanism**. It provides the **powerful exception handling and type checking mechanism** as compare to other programming languages.

Compiler checks the program whether there any error and interpreter checks any run time error and makes the system secure from crash. All of the above features makes the java language robust.

Distributed

The widely used protocols like HTTP and FTP are developed in Java. Internet programmers can call functions on these protocols and can get access the files from any remote machine on the internet rather than writing codes on their local system.

Portable

The feature **Write-once-run-anywhere** makes the java language portable provided that the system must have interpreter for the JVM. Java also have the **standard data size irrespective of operating system or the processor**. These features makes the java as a portable language.

Dynamic

While executing the Java program the user can get the required files dynamically from a local drive or from a computer thousands of miles away from the user just by connecting with the Internet.

Secure

- Java does not use memory pointers explicitly. All the programs in Java are run under an area known as the sand box.
- Security manager determines the accessibility options of a class like reading and writing a file to the local disk.
- Java uses the **public key encryption system** to allow the Java applications to transmit over the internet in the secure encrypted form.
- The bytecode Verifier checks the classes after loading.

Performance

Java uses native code usage, and lightweight process called **threads**. In the beginning interpretation of bytecode resulted the performance slow but the advance version of JVM uses the adaptive and just-in-time compilation technique that improves the performance.

Multithreaded

- Java is also a Multithreaded programming language.
- Multithreading means a single program having different threads executing independently at the same time. Multiple threads execute instructions according to the program code in a process or a program. Multithreading works the similar way as multiple processes run on one computer.

- Multithreading programming is a very interesting concept in Java. In multithreaded programs not even a single thread disturbs the execution of other thread. Threads are obtained from the pool of available ready to run threads and they run on the system CPUs.

Interpreted

- We all know that Java is an interpreted language as well. With an interpreted language such as Java, programs run directly from the source code.
- The interpreter program reads the source code and translates it on the fly into computations. Thus, Java as an interpreted language depends on an interpreter program.
- The versatility of being platform independent makes Java to outshine from other languages. The source code to be written and distributed is platform independent.
- Another advantage of Java as an interpreted language is its error debugging quality. Due to this any error occurring in the program gets traced. This is how it is different to work with Java.

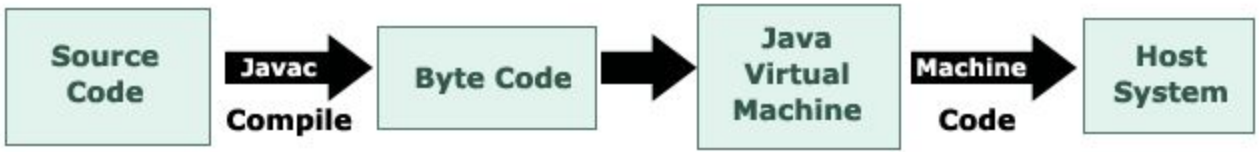
Architecture-Neutral

- The term architectural neutral seems to be weird, but yes Java is an architectural neutral language as well.
- The growing popularity of networks makes developers think distributed. In the world of network it is essential that the applications must be able to migrate easily to different computer systems. Not only to computer systems but to a wide variety of hardware architecture and Operating system architectures as well.
- The Java compiler does this by generating byte code instructions, to be easily interpreted on any machine and to be easily translated into native machine code on the fly.
- The compiler generates an architecture-neutral object file format to enable a Java application to execute anywhere on the network and then the compiled code is executed on many processors, given the presence of the Java runtime system. Hence Java was designed to support applications on network. This feature of Java has thrived the programming language.

References: https://www.roseindia.net/java/java-introduction/java-features.shtml?expand_article=1

Java Virtual Machine

- **JVM** is the main component of Java architecture and it is the part of the **JRE** (Java Runtime Environment) .
- It provides the **cross platform functionality** to Java.
- This is a software process that converts the compiled Java bytecode to machine code.
- **Bytecode** is an **intermediary language between Java source and the host system**. Most programming language like C and Pascal converts the source code into machine code for one specific type of machine as the machine language vary from system to system . Mostly compiler produce code for a particular system but **Java compiler produce code for a virtual machine** . JVM provides security to Java.
- The programs written in Java or the source code translated by Java compiler into byte code and after that the JVM converts the byte code into machine code for the computer one wants to run. JVM is a part of Java RunTime Environment that is required by every operating system requires a different JRE .
- The architecture of the JVM is given below . This architecture tell us how the JVM works . Firstly we write the simple java program(source code) the java compiler converts the source code into the bytecode , after that JVM reads this bytecode and converts this into the machine code.


Source Code
Javac
Compile
Byte Code
Java Virtual Machine
Machine
Code
Host System

JVM provides portability explain..?

- Implementations of Java specification for a variety of **CPUs and architectures** provides the feature of portability. Foremost, without the availability of a JRE for a given environment, it is impossible to run Java software. JVM forms the part of large system i.e. the Java Runtime Environment (**JRE**). Each operating system and CPU architecture requires a JRE. **JRE consists of a set of base classes i.e. an implementation of the base Java API as well as a JVM**. The byte code format is same on all platforms as it runs in the same JVM and it is totally independent from the Operating System and the CPU architecture.
- JVM is Java interpreter as it converts the byte code into machine code for the computer one wants to run.
- **JRE** consists of a number of classes based on Java API and JVM, and without JRE, it is impossible to run Java. So its portability really made it possible in developing write once and run anywhere software .

How to write the first Java Program

```
public class MyFirstProgram

{

  public static void main(String arr[] )

   {

    System.out.println(" I am creating my first java program");

  }

}
```

Save the file with same name as the public class just adding the extension ?.java? e.g. MyFirstProgram.java.

Now compile as:

c:\ javac MyFirstProgram.java

This creates a class file in with the same name, This is the bytecode form of Java program.

Now to execute this code:
c:\ java MyFirstProgram

OUTPUT

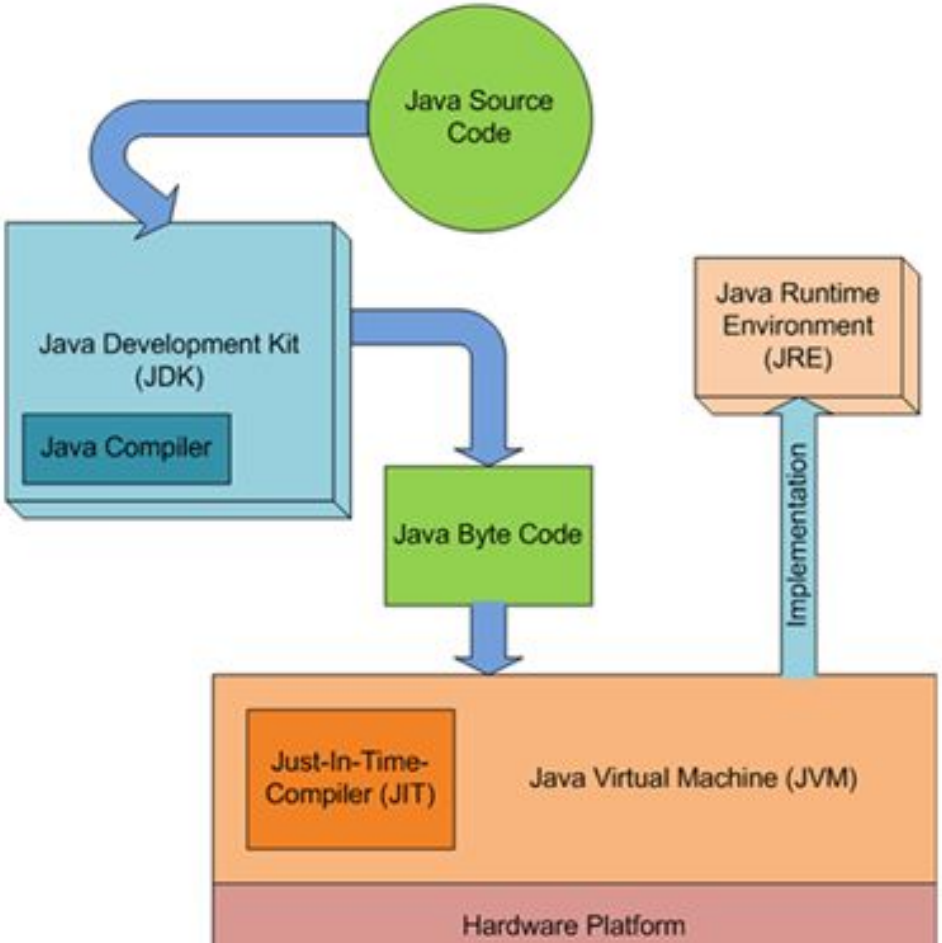
I am creating my first java program

Memory Management with JVM

the Java language in combination with runtime eliminates the problems of memory management and corrupted pointers.

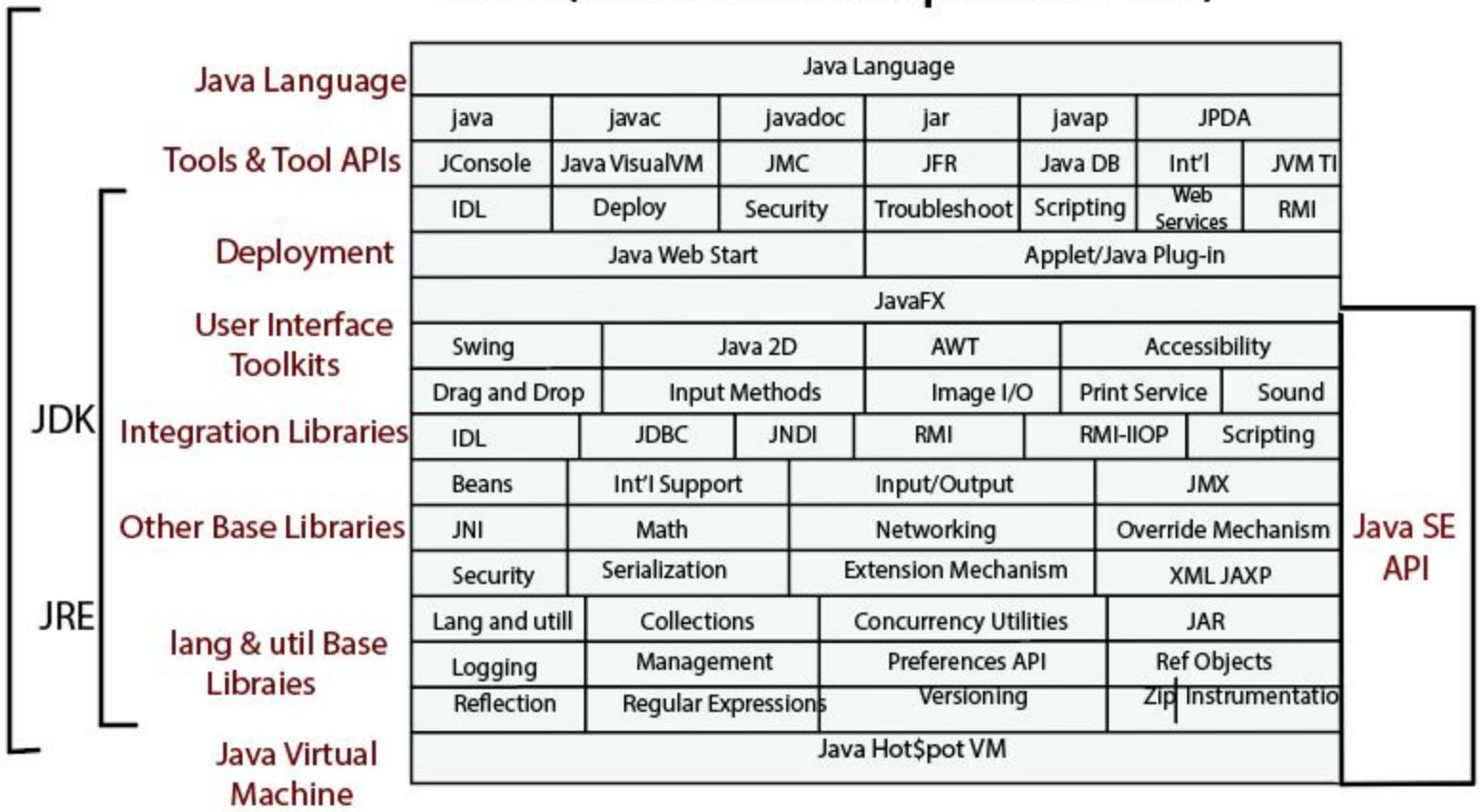
1. There is no explicit allocation of memory in Java, memory is allocated only to objects.
2. JVM's **heap** stores all objects created.
3. JVM ask the operating system for enough memory to run the JVM itself and some free memory for the application to create new objects.
4. If the free memory area is getting too small, the JVM will ask the operating system for more and if there is no more additional memory available from the operating system, then JVM stops the application and issues the "**Out of memory error**".
5. The Java runtime employs a **garbage collector** to reclaim the memory occupied by an object.

Reference: https://www.roseindia.net/java/java-virtual-machine.shtml?expand_article=1

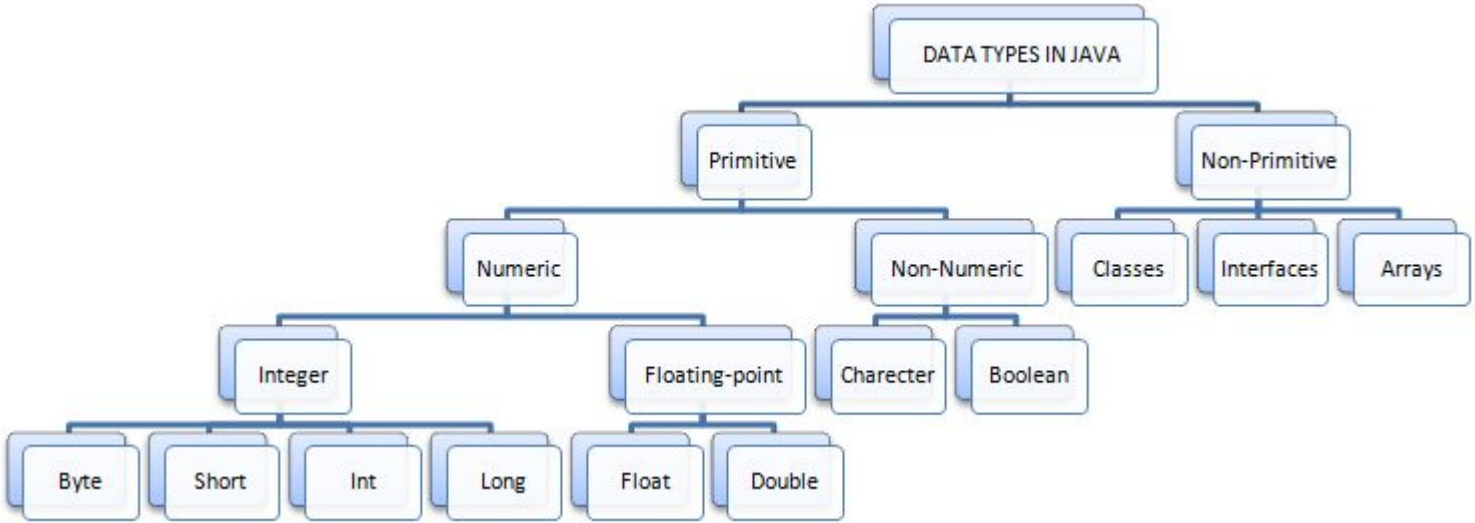

Java Source Code
Java Development Kit (JDK)
Java Compiler
Java Byte Code
Java Runtime Environment (JRE)
Implementation
Just-In-Time-Compiler (JIT)
Java Virtual Machine (JVM)
Hardware Platform

JDK	JRE	JVM
The full form of JDK is Java Development Kit.	The full form of JRE is Java Runtime Environment.	The full form of JVM is Java Virtual Machine.
JDK is a software development kit to develop applications in Java.	It is a software bundle which provides Java class libraries with necessary components to run Java code.	JVM executes Java byte code and provides an environment for executing it.
JDK is platform dependent.	JRE is also platform dependent.	JVM is platform-independent.
It contains tools for developing, debugging, and monitoring java code.	It contains class libraries and other supporting files that JVM requires to execute the program.	Software development tools are not included in JVM.
It is the superset of JRE	It is the subset of JDK.	JVM is a subset of JRE.
The JDK enables developers to create Java programs that can be executed and run by the JRE and JVM.	The JRE is the part of Java that creates the JVM.	It is the Java platform component that executes source code.
JDK comes with the installer.	JRE only contain environment to execute source code.	JVM bundled in both software JDK and JRE.

JDK(Java Development Kit)



Data Types in Java



Java Primitive Data Types

Primitive Types					
Type Name	Wrapper class	Value	Range	Size	Default Value
`byte`	`java.lang.Byte`	integer	−128 through +127	8-bit (1-byte)	`0`
`short`	`java.lang.Short`	integer	−32,768 through +32,767	16-bit (2-byte)	`0`
`int`	`java.lang.Integer`	integer	−2,147,483,648 through +2,147,483,647	32-bit (4-byte)	`0`
`long`	`java.lang.Long`	integer	−9,223,372,036,854,775,808 through +9,223,372,036,854,775,807	64-bit (8-byte)	`0`
`float`	`java.lang.Float`	floating point number	±1.401298E−45 through ±3.402823E+38	32-bit (4-byte)	`0.0`
`double`	`java.lang.Double`	floating point number	±4.94065645841246E−324 through ±1.79769313486232E+308	64-bit (8-byte)	`0.0`
`boolean`	`java.lang.Boolean`	Boolean	`true` or `false`	8-bit (1-byte)	`false`
`char`	`java.lang.Character`	UTF-16 code unit (BMP character or a part of a surrogate pair)	`'\u0000'` through `'\uFFFF'`	16-bit (2-byte)	`'\u0000'`

Summary

- Inventors of Java
- History of Java
- Features of Java
- JVM, JRE and JDK
- Java Datatypes
- Java Primitive Data types

THATS ALL FOR TODAY!!!!