What is Java?

- Java is a **programming language** and a **platform**. Java is a high level, robust, object-oriented and secure programming language.

- Java was developed by *Sun Microsystems* (which is now the subsidiary of Oracle) in the year 1995. *James Gosling* is known as the father of Java.

- Before Java, its name was *Oak*. Since Oak was already a registered company, so James Gosling and his team changed the Oak name to Java.

History of JAVA

- The history of Java starts with the Green Team. Java team members (also known as **Green Team**), initiated this project to develop a language for digital devices such as set-top boxes, televisions, etc.

- Java was developed by James Gosling, who is known as the father of Java, in 1995.

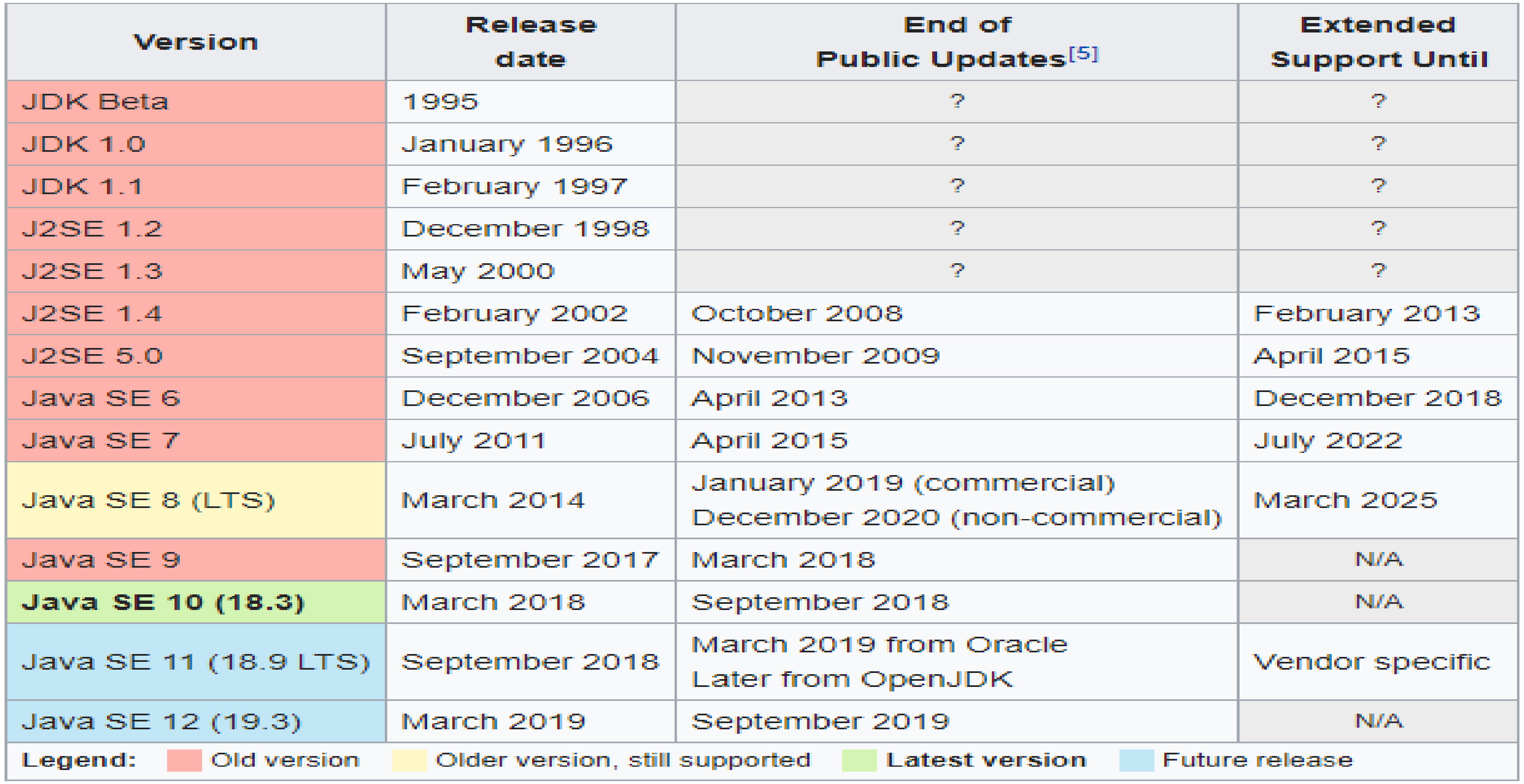
- Java is an island of Indonesia where the first coffee was produced (called java coffee). Java name was chosen by James Gosling while having coffee near his office.

Features of JAVA

1. Simple
2. Object-Oriented
3. Portable---(JAVA Byte Code)
4. Platform independent
5. Secured (SSL)
6. Robust---(Memory Management)
7. Architecture neutral---(Size)
8. High Performance---(Interpreted Programming Language)
9. Multithreaded
10. Distributed---(RMI & EJB)
11. Dynamic

JAVA Versions & Release Date

LTS (long-term support)

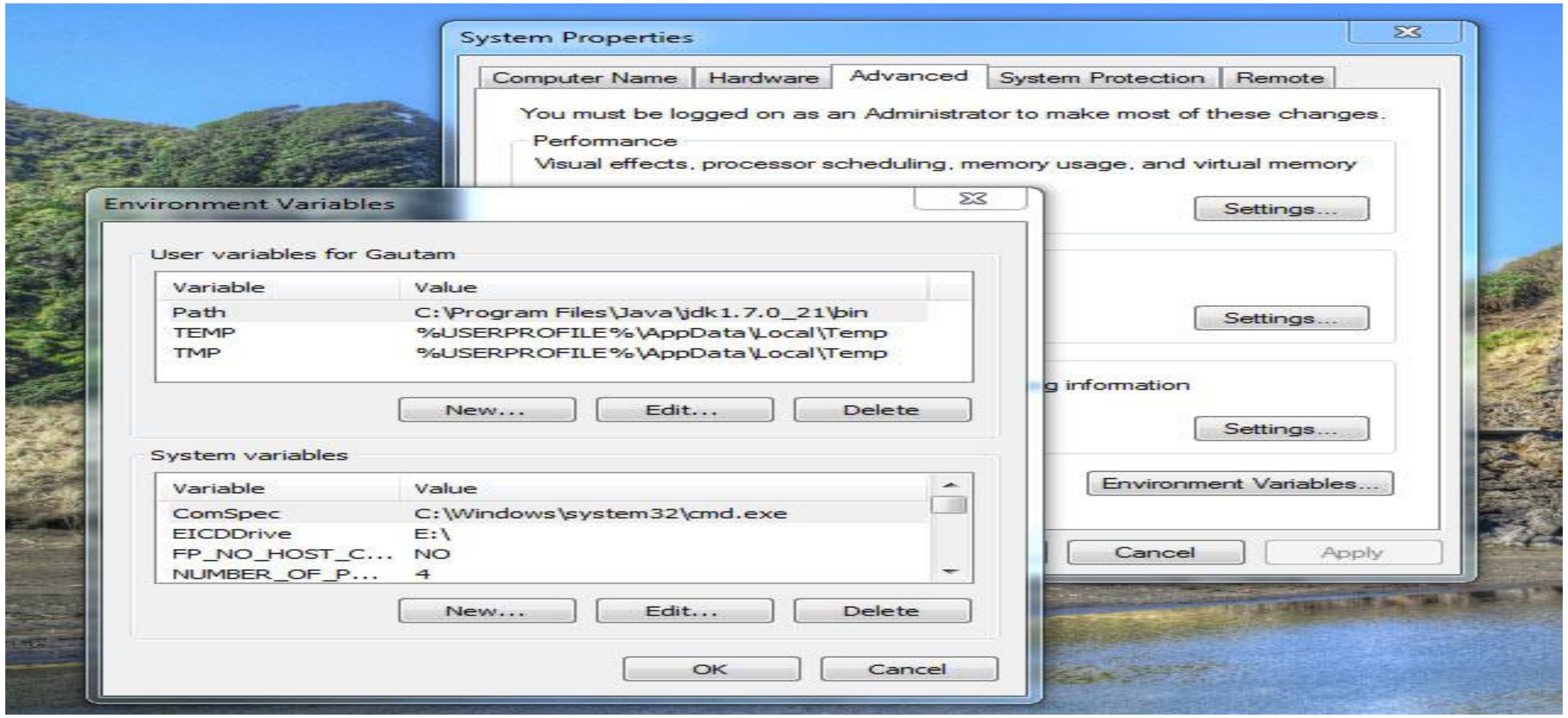


Version	Release date	End of Public Updates[5]	Extended Support Until
JDK Beta	1995	?	?
JDK 1.0	January 1996	?	?
JDK 1.1	February 1997	?	?
J2SE 1.2	December 1998	?	?
J2SE 1.3	May 2000	?	?
J2SE 1.4	February 2002	October 2008	February 2013
J2SE 5.0	September 2004	November 2009	April 2015
Java SE 6	December 2006	April 2013	December 2018
Java SE 7	July 2011	April 2015	July 2022
Java SE 8 (LTS)	March 2014	January 2019 (commercial) December 2020 (non-commercial)	March 2025
Java SE 9	September 2017	March 2018	N/A
Java SE 10 (18.3)	March 2018	September 2018	N/A
Java SE 11 (18.9 LTS)	September 2018	March 2019 from Oracle Later from OpenJDK	Vendor specific
Java SE 12 (19.3)	March 2019	September 2019	N/A

Legend: 🟥 Old version 🟨 Older version, still supported 🟩 **Latest version** 🟦 Future release

Set Path for JAVA in Windows OS

- Suppose you have installed Java in C:\Program Files\Java\jdk1.7.0_21\bin directory
- Right-click on My Computer icon and select Properties.
- Click on the 'Environment variables' button under the 'Advanced' or 'Advanced system settings' tab.
- Add Path as new variable name and C:\Program Files\Java\jdk1.7.0_21\bin as variable value.



You must be logged on as an Administrator to make most of these changes.

Visual effects, processor scheduling, memory usage, and virtual memory

User variables for Gautam

Variable	Value
Path	C:\Program Files\Java\jdk1.7.0_21\bin
TEMP	%USERPROFILE%\AppData\Local\Temp
TMP	%USERPROFILE%\AppData\Local\Temp

System variables

Variable	Value
ComSpec	C:\Windows\system32\cmd.exe
EICDDrive	E:\
FP_NO_HOST_C...	NO
NUMBER_OF_P...	4

Difference between JDK, JRE & JVM

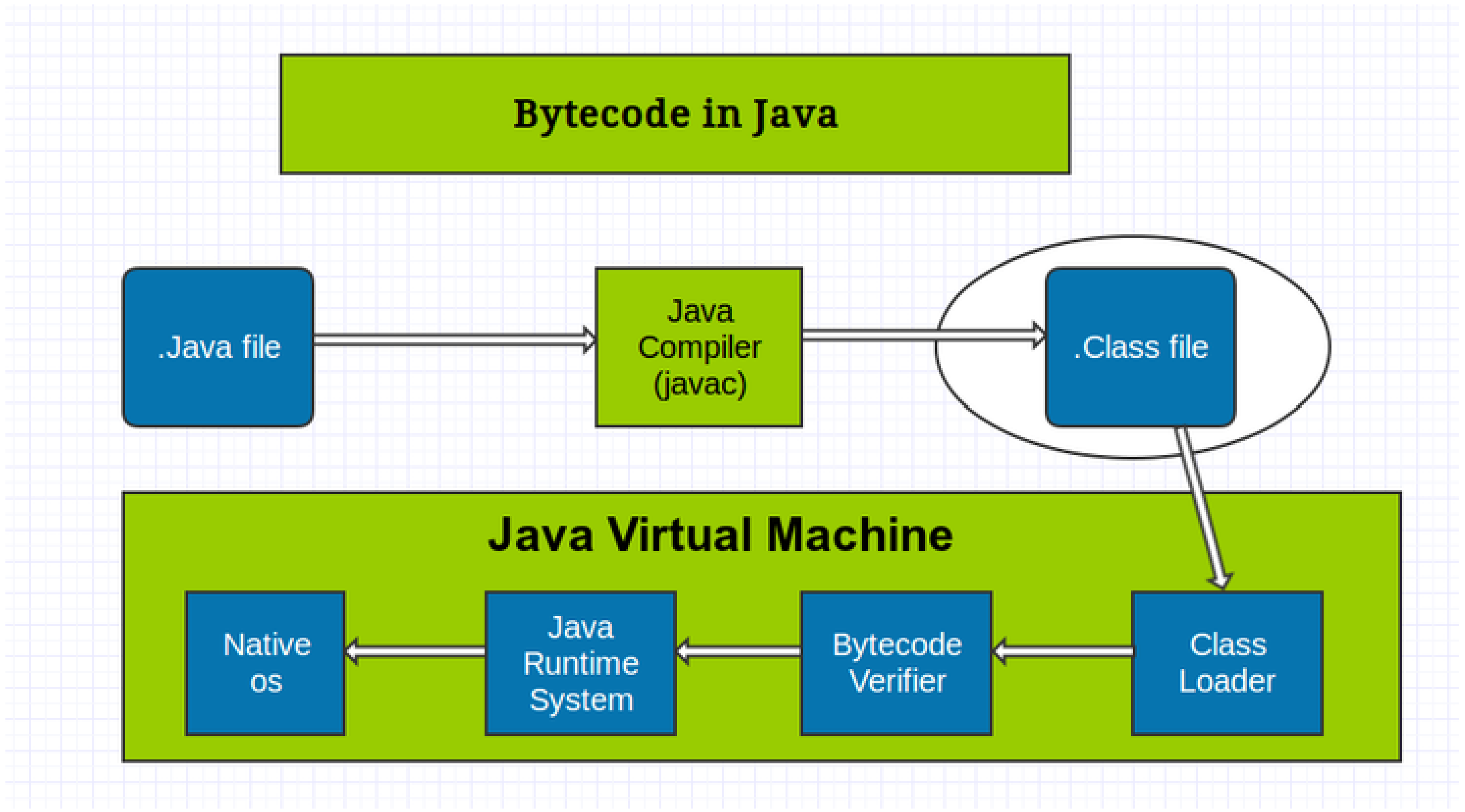
<u>JVM</u>

- ➢ JVM (Java Virtual Machine) is an abstract machine. It is called a virtual machine because it doesn't physically exist. It is a specification that provides a runtime environment in which Java bytecode can be executed. It can also run those programs which are written in other languages and compiled to Java bytecode.

- ➢ The JVM performs the following main tasks:

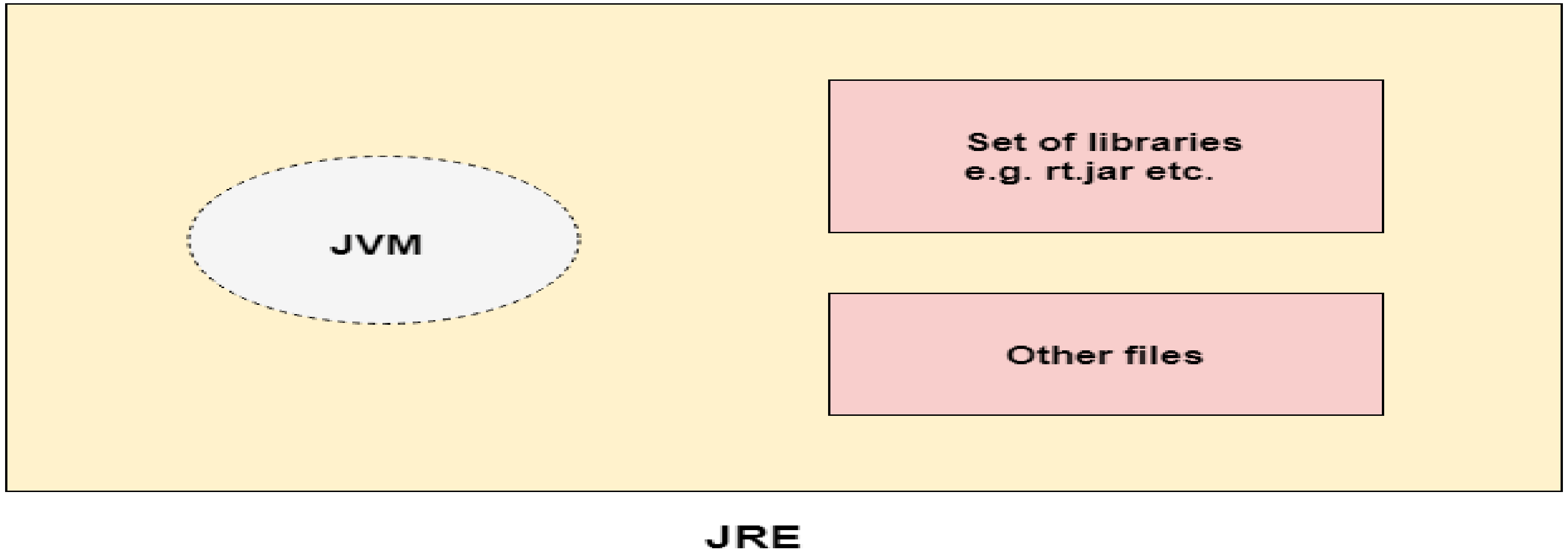
- Loads code
- Verifies code
- Executes code
- Provides runtime environment

JVM


Bytecode in Java
.Java file
Java Compiler (javac)
.Class file
Java Virtual Machine
Native os
Java Runtime System
Bytecode Verifier
Class Loader

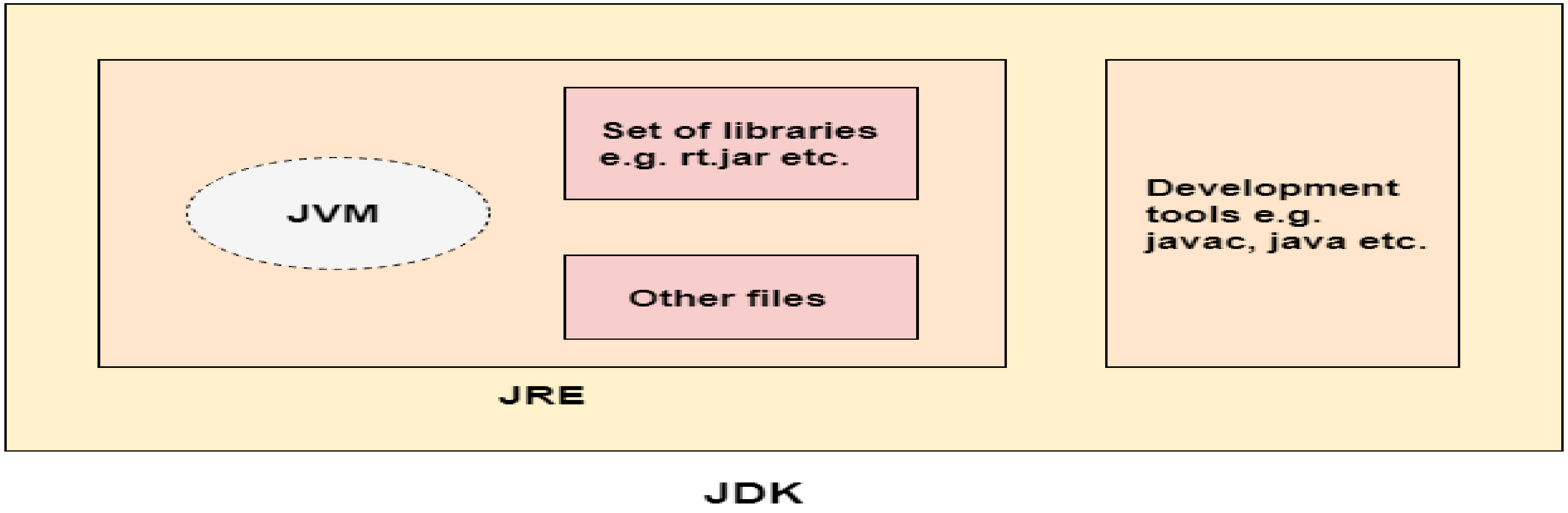
JRE

- JRE is an acronym for Java Runtime Environment. It is also written as Java RTE. The Java Runtime Environment is a set of software tools which are used for developing Java applications. It is used to provide the runtime environment. It is the implementation of JVM. It physically exists. It contains a set of libraries + other files that JVM uses at runtime.
- The implementation of JVM is also actively released by other companies besides Sun Micro Systems.



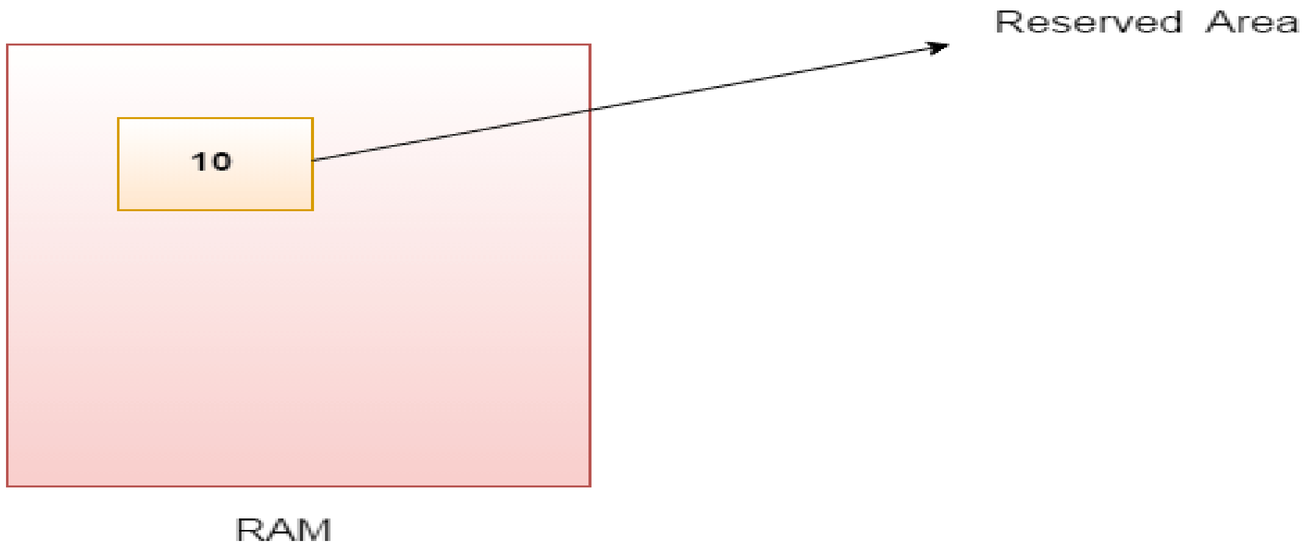
JDK

JDK is an acronym for Java Development Kit. The Java Development Kit (JDK) is a software development environment which is used to develop Java applications and applets. It physically exists. It contains JRE + development tools. The JDK contains a private Java Virtual Machine (JVM) and a few other resources such as an interpreter/loader (java), a compiler (javac), an archiver (jar), a documentation generator (Javadoc), etc. to complete the development of a Java Application.



Variable

- A variable is a container which holds the value while the java program is executed. A variable is assigned with a datatype.
- Variable is a name of memory location. There are three types of variables in java: local, instance and static.
- **Variable** is name of *reserved area allocated in memory*. In other words, it is a *name of memory location*. It is a combination of "vary + able" that means its value can be changed.



Ex: int a=10;

Variable continued…

1) **Local Variable**

A variable declared inside the body of the method is called local variable. You can use this variable only within that method and the other methods in the class aren't even aware that the variable exists.

A local variable cannot be defined with "static" keyword.

2) **Instance Variable**

A variable declared inside the class but outside the body of the method, is called instance variable. It is not declared as static.

It is called instance variable because its value is instance specific and is not shared among instances.

3) **Static variable**

A variable which is declared as static is called static variable. It cannot be local. You can create a single copy of static variable and share among all the instances of the class. Memory allocation for static variable happens only once when the class is loaded in the memory.

```
class A{
int data=50;//instance variable
static int m=100;//static variable
void method(){
int n=90;//local variable
}
}//end of class
```

Keyword

Java keywords are also known as **reserved words**. Keywords are particular words which acts as a key to a code. These are predefined words by Java so it cannot be used as a variable or object name. You can't use keyword as identifier in your Java programs, its reserved words in Java library and used to perform an internal operation. *true*, *false* and *null* are not reserved words but cannot be used as identifiers, because it is literals of built-in types.

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

Data Types

- ➢Data types specify the different sizes and values that can be stored in the variable. There are two types of data types in Java:

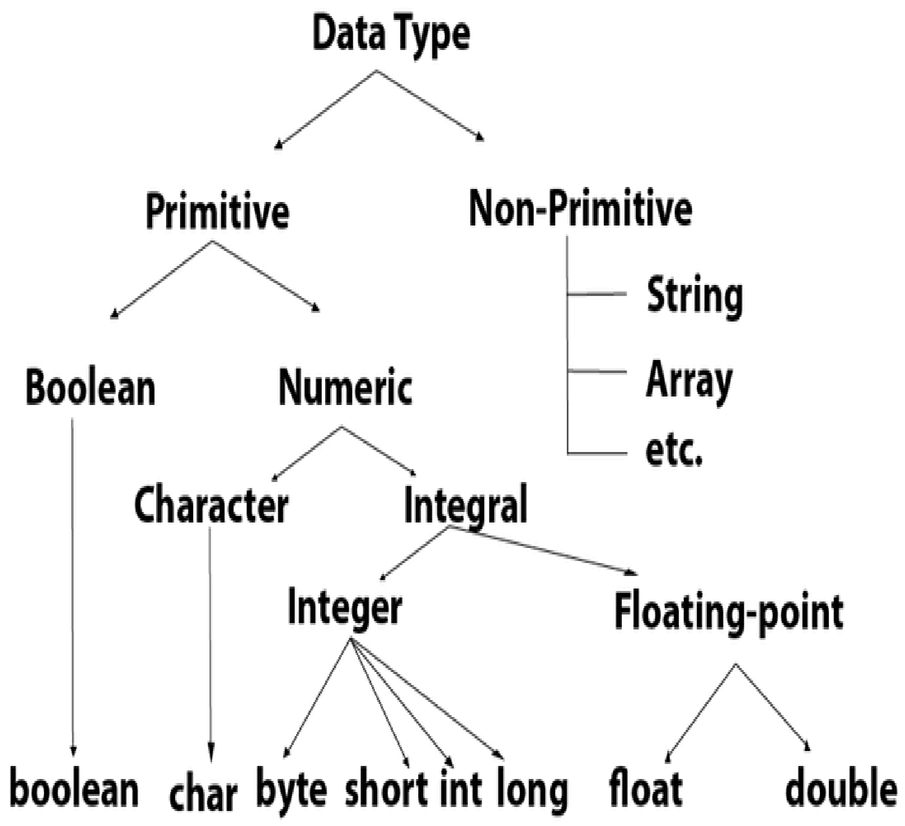
- **Primitive data types:** The primitive data types include Boolean, char, byte, short, int, long, float and double.
- **Non-primitive data types:** The non-primitive data types include Classes, Interfaces, and Arrays.

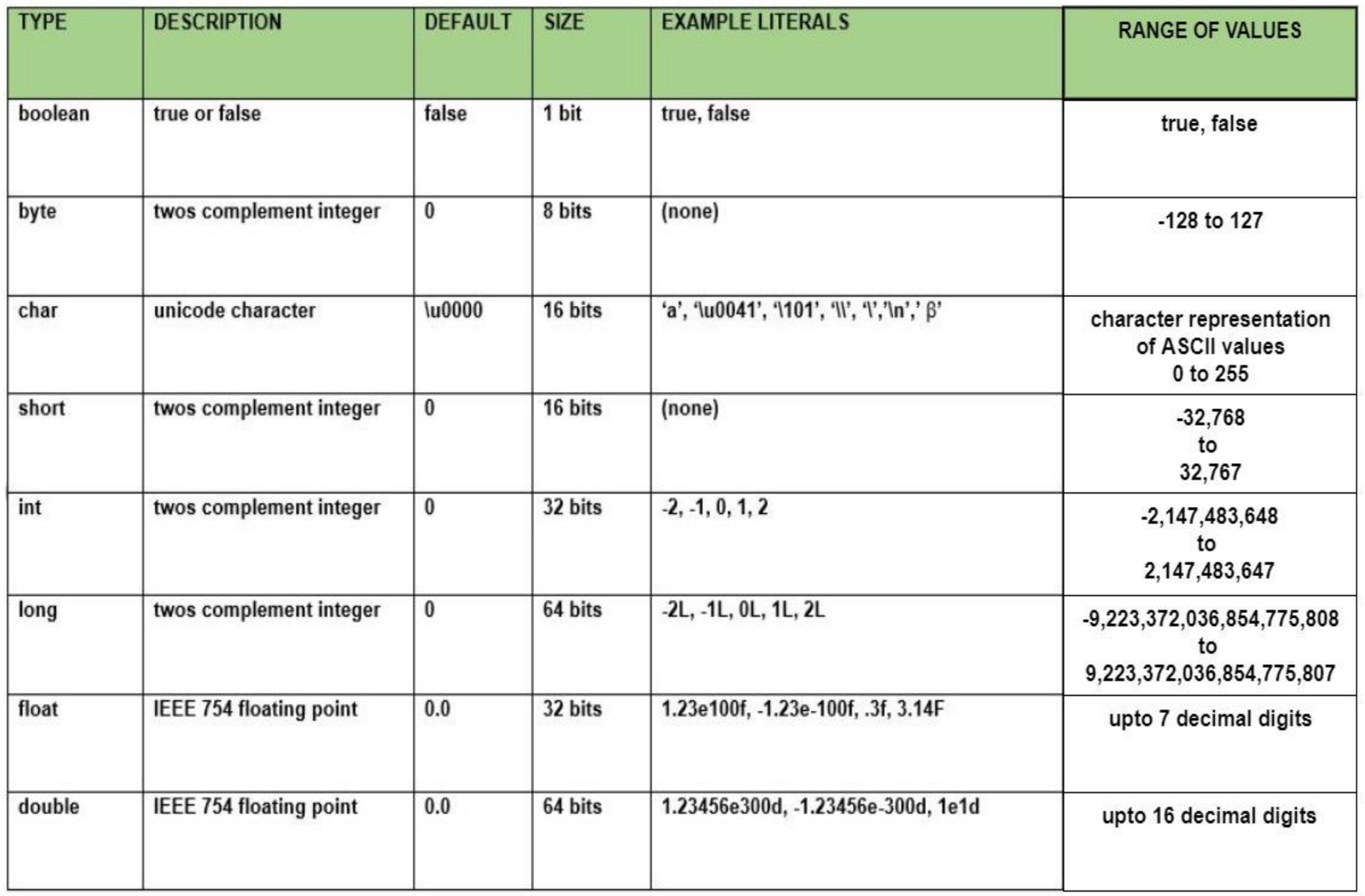
- ➢In Java language, primitive data types are the building blocks of data manipulation.
- ➢Java is a statically-typed programming language. It means, all variables must be declared before its use. That is why we need to declare variable's type and name.

➢**<u>Why char uses 2 byte in java and what is \u0000 ?</u>**

- It is because java uses Unicode system not ASCII code system. The \u0000 is the lowest range of Unicode system.

Data Types continued….





TYPE	DESCRIPTION	DEFAULT	SIZE	EXAMPLE LITERALS	RANGE OF VALUES
boolean	true or false	false	1 bit	true, false	true, false
byte	twos complement integer	0	8 bits	(none)	-128 to 127
char	unicode character	\u0000	16 bits	‘a’, ‘\u0041’, ‘\101’, ‘\\’, ‘\’,’\n’,’ β’	character representation of ASCII values 0 to 255
short	twos complement integer	0	16 bits	(none)	-32,768 to 32,767
int	twos complement integer	0	32 bits	-2, -1, 0, 1, 2	-2,147,483,648 to 2,147,483,647
long	twos complement integer	0	64 bits	-2L, -1L, 0L, 1L, 2L	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	IEEE 754 floating point	0.0	32 bits	1.23e100f, -1.23e-100f, .3f, 3.14F	upto 7 decimal digits
double	IEEE 754 floating point	0.0	64 bits	1.23456e300d, -1.23456e-300d, 1e1d	upto 16 decimal digits

Data Types continued….

```
class DataTypes
{
 public static void main(String args[])
{
byte byteVar = 5;
short shortVar = 20;
int intVar = 30;
long longVar = 60;
float floatVar = 20;
double doubleVar = 20.123;
boolean booleanVar = true;
char charVar =‘W’;
System.out.println("Value Of byte Variable is " + byteVar); System.out.println("Value Of short Variable is " + shortVar);
System.out.println("Value Of int Variable is " + intVar); System.out.println("Value Of long Variable is " + longVar);
System.out.println("Value Of float Variable is " + floatVar); System.out.println("Value Of double Variable is " + doubleVar);
System.out.println("Value Of boolean Variable is " + booleanVar); System.out.println("Value Of char Variable is " + charVar);
}
}
```

Unicode System

- Unicode is a universal international standard character encoding that is capable of representing most of the world's written languages.

Before Unicode, there were many language standards:

- **ASCII** (American Standard Code for Information Interchange) for the United States.
- **ISO 8859-1** for Western European Language.
- **KOI-8** for Russian.
- **GB18030 and BIG-5** for chinese, and so on.

Unicode System continued…

➢**Problem:**

1. A particular code value corresponds to different letters in the various language standards.
2. The encodings for languages with large character sets have variable length. Some common characters are encoded as single bytes, other require two or more byte.

➢**Solution:**

To solve these problems, a new language standard was developed i.e. Unicode System.

In unicode, character holds 2 byte, so java also uses 2 byte for characters.

lowest value: \u0000

highest value: \uFFFF

Operators

- **Operator** in java is a symbol that is used to perform operations. For example: +, -, *, / etc.

- There are 8 types of operators in java which are given below:

1. Unary Operator
2. Arithmetic Operator
3. Shift Operator
4. Relational Operator
5. Bitwise Operator
6. Logical Operator
7. Ternary Operator
8. Assignment Operator

Operators continued…

Operator Type	Category	Precedence
Unary	postfix	*expr*++ *expr*--
	prefix	++*expr* --*expr* +*expr* -*expr* ~ !
Arithmetic	multiplicative	* / %
	additive	+ -
Shift	shift	<< >> >>>
Relational	comparison	< > <= >= instanceof
	equality	== !=
Bitwise	bitwise AND	&
	bitwise exclusive OR	^
	bitwise inclusive OR	|
Logical	logical AND	&&
	logical OR	||
Ternary	ternary	? :
Assignment	assignment	= += -= *= /= %= &= ^= |= <<= >>= >>>=

Control Statements in JAVA

I. Decision Making in Java
   1. Simple if Statement
   2. if…else Statement
   3. Nested if statement
   4. if...else if…else statement
   5. Switch statement

II. Looping Statements in Java
   1. While
   2. Do…while
   3. For
   4. For-Each Loop

III. Branching Statements in Java
   1. Break
   2. Continue
   3. Return

I. Decision Making in Java

- Decision making statements are statements which decides what to execute and when. They are similar to decision making in real time.

1. Simple if Statement

- Simple if statement is the basic of decision-making statements in Java. It decides if certain amount of code should be executed based on the condition.

Syntax:

```
if (condition) {
Statement 1; //executed if condition is true
}
Statement 2; //executed irrespective of the condition
```

2. **if…else Statement**

- In if…else statement, if condition is true then statements in if block will be executed but if it comes out as false then else block will be executed.

Example:

```
public class Main
{
public static void main(String args[])
{
int a =10 ;
if (a > 5)
System.out.println(“a is Greater");
else
System.out.println("a is Lesser");
}
}
}
```

3. Nested if statement

- Nested if statement is if inside an if block. It is same as normal if…else statement but they are written inside another if…else statement.

Syntax:

```
if (condition1) {
 Statement 1; //executed when condition1 is true
 if (condition2)
{
 Statement 2; //executed when condition2 is true
}
else
{
 Statement 3; //executed when condition2 is false
}
}
```

4. if...else if…else statement

- if…else if statements will be used when we need to compare the value with more than 2 conditions. They are executed from top to bottom approach. As soon as the code finds the matching condition, that block will be executed. But if no condition is matching then the last else statement will be executed.

Syntax:

```
if (condition2) {
 Statement 1; //if condition1 becomes true then this will be executed
}
 else if (condition2)
{
 Statement 2; // if condition2 becomes true then this will be executed
 }
else
{
Statement 3; //executed when no matching condition found
}
```

5. Switch statement

- Java switch statement compares the value and executes one of the case blocks based on the condition. It is same as if…else if ladder. Below are some points to consider while working with switch statements:

- ➢case value must be of the same type as expression used in switch statement.
- ➢case value must be a constant or literal. It doesn’t allow variables.
- ➢case values should be unique. If it is duplicate, then program will give compile time error.

Example:

```
public class Music {
public static void main(String[] args)
{
int instrument = 4;
String musicInstrument;
// switch statement with int data type
switch (instrument) {
case 1:
musicInstrument = "Guitar";
break;
case 2:
musicInstrument = "Piano";
break;
case 3:
musicInstrument = "Drums";
break;
case 4:
musicInstrument = "Flute";
break;
case 5:
musicInstrument = "Ukelele";
break;
case 6:
musicInstrument = "Violin";
break;
case 7:
musicInstrument = "Trumpet";
break;
default:
musicInstrument = "Invalid";
break;
}
System.out.println(musicInstrument);
}
}
```

II. Looping Statements in Java

- Looping statements are the statements which executes a block of code repeatedly until some condition meet to the criteria. Loops can be considered as repeating if statements.

1. While

- While loops are simplest kind of loop. It checks and evaluates the condition and if it is true then executes the body of loop. This is repeated until the condition becomes false. Condition in while loop must be given as a Boolean expression. If int or string is used instead, compile will give the error.

Syntax:

```
while (condition)
{
statement1;
}
```

2. Do…while

Do…while works same as while loop. It has only one difference that in do…while, condition is checked after the execution of the loop body. That is why this loop is considered as exit control loop. In do…while loop, body of loop will be executed at least once before checking the condition.

Syntax:

```
do
{
statement1;
}while(condition);
```

Example:

```
class dowhileDemo
{
public static void main(String args[])
{
 int j = 10;
do
{
 System.out.println(j);
j = j+1;
 } while (j <= 10)
 }
}
```

3. **For Loop**

It is the most common and widely used loop in Java. It is the easiest way to construct a loop structure in code as initialization of a variable, a condition and increment/decrement are declared only in a single line of code. It is easy to debug structure in Java.

Syntax:

```
for (initialization; condition; increment/decrement)
 {
statement;
}
```

Example:

```
class forLoopDemo
{
public static void main(String args[])
{
for (int i= 1; i<= 5; i++)
{
System.out.println(i);
}
}
}
```

4. For-Each Loop

For-Each loop is used to traverse through elements in an array. It is easier to use because we don’t have to increment the value. It returns the elements from the array or collection one by one.

Example:

```
class foreachDemo
{
public static void main(String args[])
{
int a[] = {10,15,20,25,30};
 for (int i : a)
{
System.out.println(i);
 }
}
```

III. Branching Statements in Java

Branching statements jump from one statement to another and transfer the execution flow.

1. Break

- Break statement is used to terminate the execution and bypass the remaining code in loop. It is mostly used in loop to stop the execution and comes out of loop. When there are nested loops then break will terminate the innermost loop.

Example:

```
class breakTestDemo
{
 public static void main(String args[])
{
for (int j = 0; j < 5; j++)
 { // come out of loop when i is 4.
if (j == 4)
 break;
System.out.println(j);
 }
System.out.println("After loop");
 }
}
```

2. <u>Continue</u>

Continue statement works same as break but the difference is it only comes out of loop for that iteration and continue to execute the code for next iterations. So it only bypasses the current iteration.

<u>Example:</u>

```
class continueTestDemo
 {
 public static void main(String args[])
{
for (int j = 0; j < 10; j++)
 { // If the number is odd then bypass and continue with next value
 if (j%2 != 0)
continue; // only even numbers will be printed
 System.out.println(j + " ");
 }
 }
 }
```

3. **Return**

Return statement is used to transfer the control back to calling method. Compiler will always bypass any sentences after return statement. So, it must be at the end of any method. They can also return a value to the calling method.

Example:

```
public String getwebURL()
 {
String a= null;
 try {
 a= driver.getCurrentUrl();
}
catch(Exception e)
{
System.out.println("Exception occured while getting the current url : "+e.getStackTrace());
}
return a;
 }
```