CORE JAVA

JAVA:

Java is Object Oriented. However, it is not considered as pure object-oriented as it provides support for primitive data types (like int, char, etc)

The Java codes are first compiled into byte code (machine-independent code). Then the byte code runs on Java Virtual Machine (JVM)

HELLO WORLD PROGRAM:

```
class HelloWorld
{
    // Your program begins with a call to main().
    // Prints "Hello, World" to the terminal window.
    public static void main(String args[])
    {
        System.out.println("Hello, World");
    }
}
```

public static void main(String[] args) definition:

public: So that JVM can execute the method from anywhere.

static: The main method is to be called without an object. The modifiers public and static can be written in either order.

void: The main method doesn't return anything.

main(): Name configured in the JVM. The main method must be inside the class definition. The compiler executes the codes starting always from the main function.

String[]: The main method accepts a single argument, i.e., an array of elements of type String.

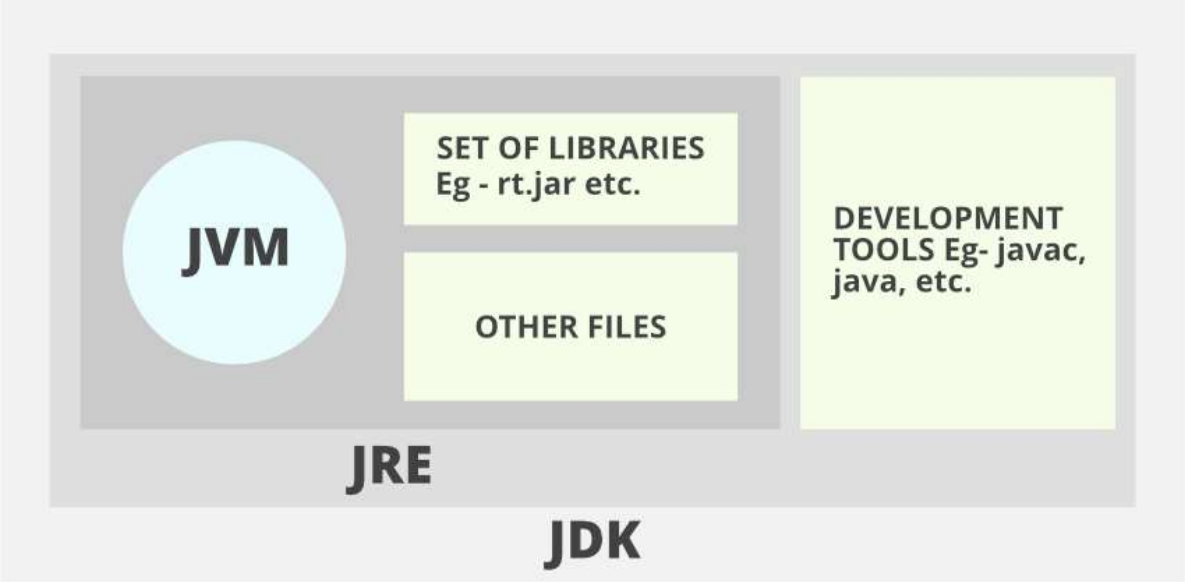
JDK, JRE AND JVM:

1. JDK (Java Development Kit) is a Kit that provides the environment to develop and execute(run) the Java program. JDK is a kit (or package) that includes two things

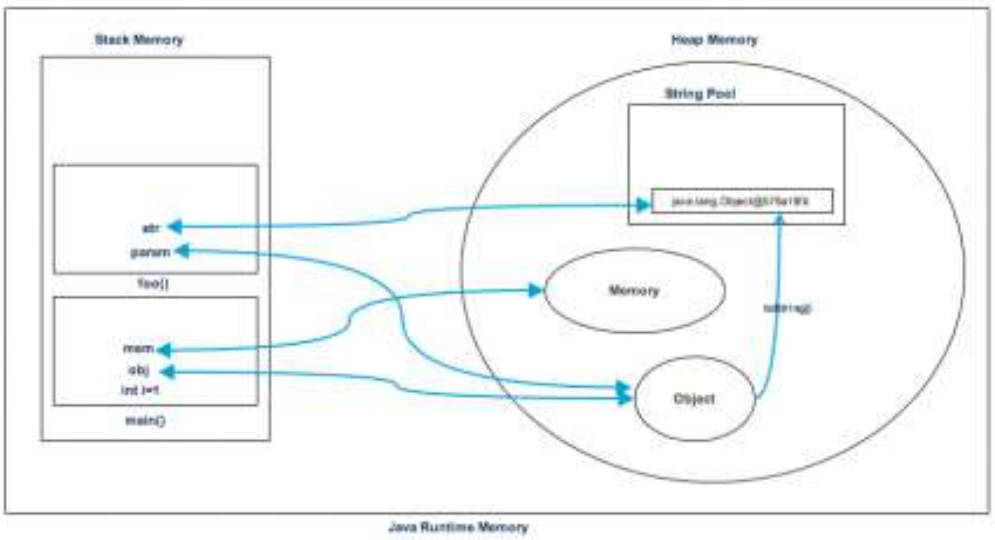
- Development Tools (to provide an environment to develop your java programs)
- JRE (to execute your java program).

2. JRE (Java Runtime Environment) is an installation package that provides an environment to only run (not develop) the java program (or application) onto your machine. JRE is only used by those who only want to run Java programs that are end-users of your system.

3. JVM (Java Virtual Machine) is a very important part of both JDK and JRE because it is contained or inbuilt in both. Whatever Java program you run using JRE or JDK goes into JVM and JVM is responsible for executing the java program line by line, hence it is also known as an interpreter.



JAVA MEMORY MANAGEMENT:



Java Stack Memory

Java Stack memory is used for the execution of a thread.

Stack memory is always referenced in LIFO (Last-In-First-Out) order.

Whenever a method is invoked, a new block is created in the stack memory for the method to hold local primitive values and reference to other objects in the method.

As soon as the method ends, the block becomes unused and becomes available for the next method. S

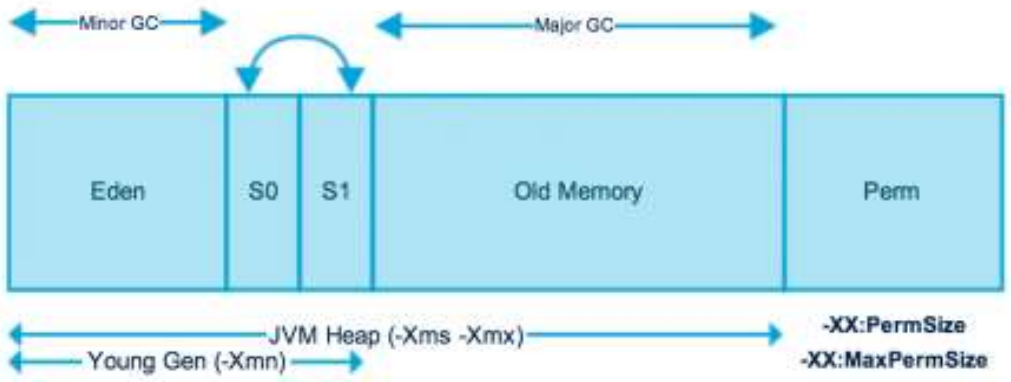
tack memory size is very less compared to Heap memory.

Java Heap Space

Java Heap space is used by java runtime to allocate memory to Objects and JRE classes.

Whenever we create an object, it's always created in the Heap space.

Garbage Collection runs on the heap memory to free the memory used by objects that don't have any reference



JVM Heap memory is physically divided into two parts - Young Generation and Old Generation.

Memory Management in Java - Young Generation

The young generation is the place where all the new objects are created. When the young generation is filled, garbage collection is performed. This garbage collection is called Minor GC. Young Generation is divided into three parts - Eden Memory and two Survivor Memory spaces.

Important Points about Young Generation Spaces:

- Most of the newly created objects are located in the Eden memory space.
- When Eden space is filled with objects, Minor GC is performed and all the survivor objects are moved to one of the survivor spaces.
- Minor GC also checks the survivor objects and move them to the other survivor space. So, at a time, one of the survivor spaces is always empty.
- Objects that are survived after many cycles of GC, are moved to the old generation memory space. Usually, it's done by setting a threshold for the age of the young generation objects before they become eligible to promote to old generation.

Memory Management in Java - Old Generation

Old Generation memory contains the objects that are long-lived and survived after many rounds of Minor GC. Usually, garbage collection is performed in Old Generation memory when it's full. Old Generation Garbage Collection is called Major GC and usually takes a longer time.

Memory Management in Java - Java Garbage Collection

Java Garbage Collection is the process to identify and remove the unused objects from the memory and free space to be allocated to objects created in future processing. One of the basic ways of garbage collection involves three steps:

- **Marking:** This is the first step where garbage collector identifies which objects are in use and which ones are not in use.
- **Normal Deletion:** Garbage Collector removes the unused objects and reclaim the free space to be allocated to other objects.
- **Deletion with Compacting:** For better performance, after deleting unused objects, all the survived objects can be moved to be together. This will increase the performance of allocation of memory to newer objects.

VARIABLES

The variable is the basic unit of storage or a container which holds the value in a Java Program.

A variable is assigned with a data type.

Based on Memory location, Variable is of three types.

1. Local Variable.

    Variable declared inside a method

    This variable cannot be used outside the method where it is declared.

2. Instance Variable.

    Variable declared outside the method but inside the class.

    This variable can be used only within the class.

3. Static Variable.

    A variable that is declared as static.

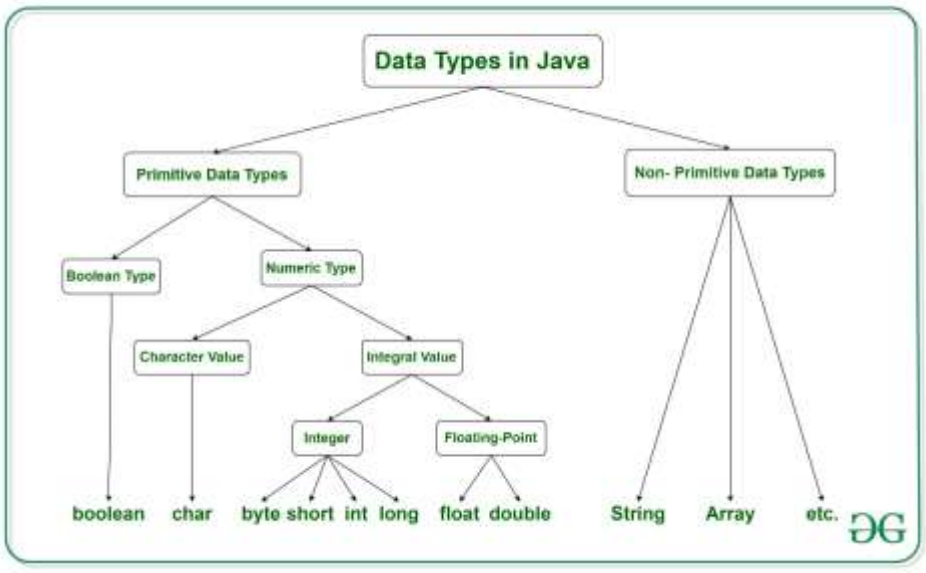
    This variable can be shared across different instance or class.

DATATYPES:

A data type in Java represents the size and different values that can be stored in a variable.

Java has two categories in which data types are segregated

1. Primitive Data Type: such as Boolean, char, int, short, byte, long, float, and double
2. Non-Primitive Data Type or Object Data type: such as String, Array, etc.



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

OPERATORS:

Operators are keywords used to perform some operations on the java code.

1. **Arithmetic Operators:** They are used to perform simple arithmetic operations on primitive data types.

    * : Multiplication

    / : Division

    % : Modulo

    + : Addition

    – : Subtraction

2**. Unary Operators:** Unary operators need only one operand. They are used to increment, decrement or negate a value.

- ++ : Increment operator, used for incrementing the value by 1. There are two varieties of increment operators.
    - Post-Increment: Value is first used for computing the result and then incremented.
    - Pre-Increment: Value is incremented first, and then the result is computed.
- -- : Decrement operator, used for decrementing the value by 1. There are two varieties of decrement operators.
    - Post-decrement: Value is first used for computing the result and then decremented.
    - Pre-Decrement: Value is decremented first, and then the result is computed.
- ! : Logical not operator, used for inverting a boolean value.

3. **Assignment Operator:** '=' Assignment operator is used to assigning a value to any variable. It has a right to left associativity, i.e. value given on the right-hand side of the operator is assigned to the variable on the left, and therefore right-hand side value must be declared before using it or should be a constant.

The general format of the assignment operator is:

variable = value;

In many cases, the assignment operator can be combined with other operators to build a shorter version of the statement called a Compound Statement. For example, instead of a = a+5, we can write a += 5.

+=, for adding left operand with right operand and then assigning it to the variable on the left.

-=, for subtracting right operand from left operand and then assigning it to the variable on the left.

*=, for multiplying left operand with right operand and then assigning it to the variable on the left.

/=, for dividing left operand by right operand and then assigning it to the variable on the left.

%=, for assigning modulo of left operand by right operand and then assigning it to the variable on the left.

4**. Relational Operators**: These operators are used to check for relations like equality, greater than, and less than. They return boolean results after the comparison and are extensively used in looping statements as well as conditional if-else statements.

Some of the relational operators are-

==, Equal to returns true if the left-hand side is equal to the right-hand side.

!=, Not Equal to returns true if the left-hand side is not equal to the right-hand side.

<, less than: returns true if the left-hand side is less than the right-hand side.

<=, less than or equal to returns true if the left-hand side is less than or equal to the right-hand side.

>, Greater than: returns true if the left-hand side is greater than the right-hand side.

>=, Greater than or equal to returns true if the left-hand side is greater than or equal to the right-hand side.

5. **Logical Operators:** These operators are used to perform "logical AND" and "logical OR" operations

&&, Logical AND: returns true when both conditions are true.

||, Logical OR: returns true if at least one condition is true.

!, Logical NOT: returns true when a condition is false and vice-versa

6. **Ternary operator:** Ternary operator is a shorthand version of the if-else statement. It has three operands and hence the name ternary.

The general format is:

condition ? if true : if false

The above statement means that if the condition evaluates to true, then execute the statements after the '?' else execute the statements after the ':'.

CONTROL STATEMENTS:

Java uses control statements to determine the flow of code execution.

Java control statements are classified into 3 types:

1. Selection statements
2. Iteration statements
3. Jump statements.

1. Selection Statements:

1. if

The if statement is Java's conditional branch statement. It can be used to route program execution through two different paths

if, if-else, if-else if- else.

2. switch

The switch statement is Java's multiway branch statement.

The switch statement contains multiple blocks of code called cases and a single case is executed based on the variable

2. Iteration statements (loops)

Java's iteration statements are for, while, and do-while.

A loop repeatedly executes the same set of instructions until a termination condition is met.

1. for

The Java for loop is a control flow statement that iterates a part of the programs multiple times.

```
for (initialization;relative operator;Increment){
}
```

2. while

It repeats a statement or block while its controlling expression is true

```
while(condition) {
// body of loop
}
```

3. do-while

The do-while loop always executes its body at least once, because its conditional expression is at the bottom of the loop.

```
do {
// body of loop
} while (condition);
```

3. Jump statements.

1. break

The Java break statement is used to break loop or switch statement.

It breaks the current flow of the program at specified condition. In case of inner loop, it breaks only inner loop.

2. continue

The continue statement is used in loop control structure when you need to jump to the next iteration of the loop immediately