History of Java

Java is a **programming language** and a **platform**.

Java is a simple, portable, platform independent, high performance, multithreaded , high level, robust, class based, concurrent, general purpose, 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 name from Oak to Java.

Types of Java Applications

There are mainly 4 types of applications that can be created using Java programming:

1) Standalone Application

2) Web Application

3) Enterprise Application

4) Mobile Application

Java Platforms / Editions

There are 4 platforms or editions of Java:

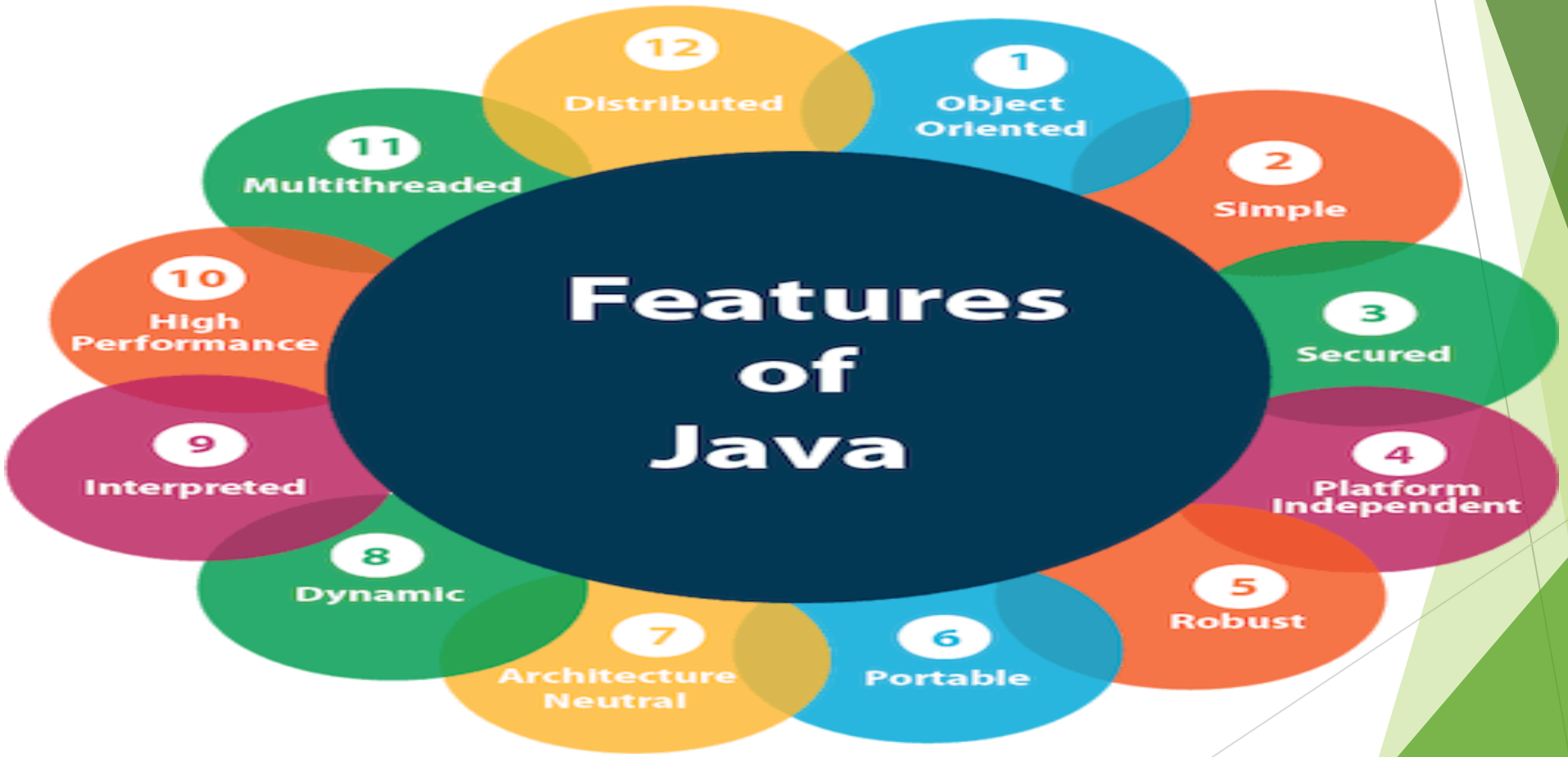
1) Java SE (Java Standard Edition)

2) Java EE (Java Enterprise Edition)

3) Java ME (Java Micro Edition)

4) JavaFX

Features of Java



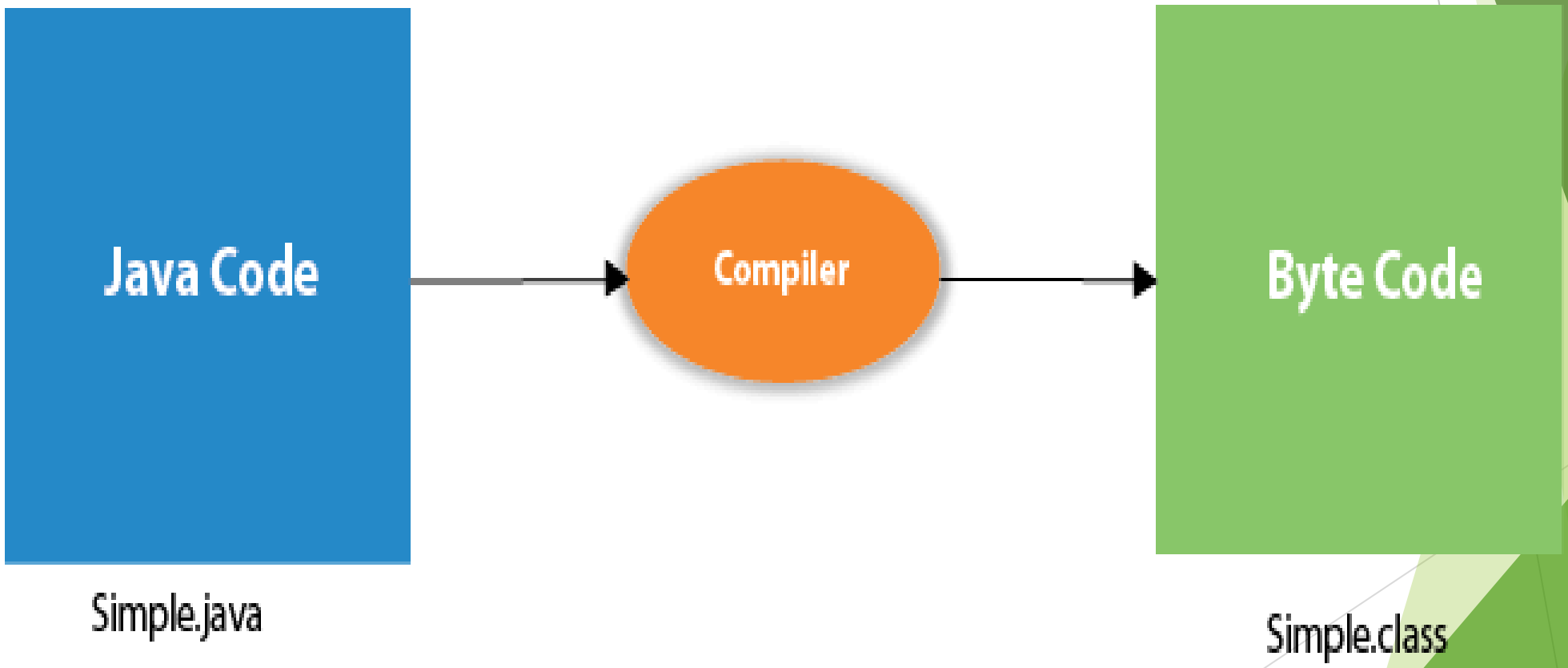
JVM

- JVM (Java Virtual Machine) is an abstract machine.
- The JVM performs the following main tasks:
  - Loads code
  - Verifies code
  - Executes code
  - Provides runtime environment

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 and other files that JVM uses at runtime.

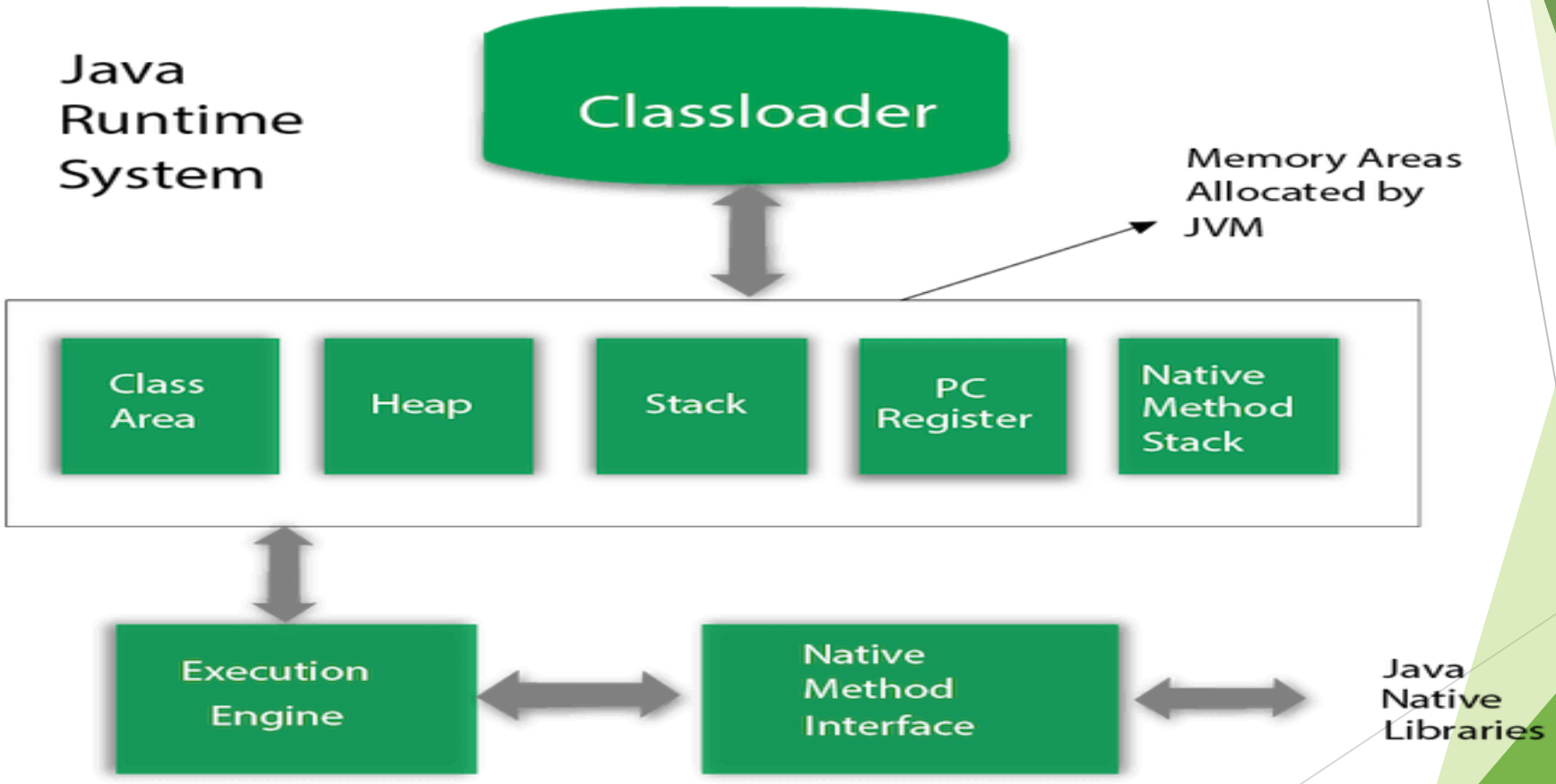
What happens at compile time?



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.
- JDK is an implementation of any one of the below given Java Platforms released by Oracle Corporation:
  - Standard Edition Java Platform
  - Enterprise Edition Java Platform
  - Micro Edition Java Platform

JVM Architecture



Java Variables

Types of Variables

- There are three types of variables in Java.
  - local variable
  - instance variable
  - static variable

Local Variable

- A variable declared inside the body of the method is called local variable. 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.

Instance Variable

- A variable declared inside the class but outside the body of the method, is called an instance variable. It is not declared as static. It is called an instance variable because its value is instance-specific and is not shared among instances.

Static variable

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

Data Types in Java

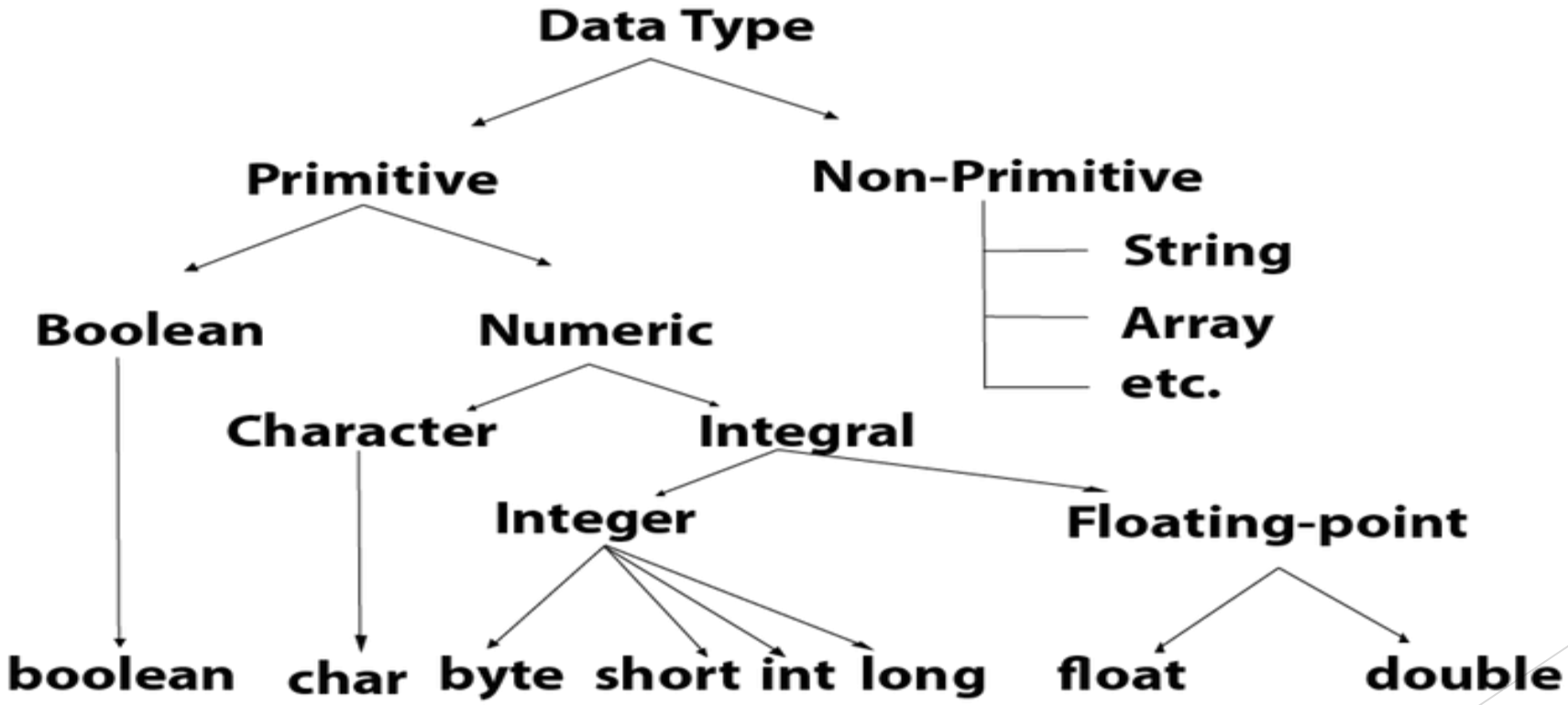
- Data types specify the different sizes and values that can be stored in the variable. There are two types of data types in Java:

1. **Primitive data types:** The primitive data types include boolean, char,

byte, short, int, long, float and double.

2. Non-primitive data types: The non-primitive data types include Classes, Interfaces and Arrays.

Data Types in Java



Java Primitive Data Types

- In Java language, primitive data types are the building blocks of data manipulation. These are the most basic data types available in Java language. 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.

Operators in Java

- **Operator** in Java is a symbol that is used to perform operations. There are many types of operators in Java which are given below:
  - Unary Operator, Arithmetic Operator,
  - Shift Operator, Relational Operator,
  - Bitwise Operator, Logical Operator,
  - Ternary Operator and Assignment Operator.

Java Operator Precedence

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	= += -= *= /= %= &= ^= |= <<= >>= >>>=

Java Keywords

- Java keywords are also known as reserved words. Keywords are particular words that act as a key to a code. These are predefined words by Java so they cannot be used as a variable or object name or class name.

List of Java Keywords

- abstract, boolean,break, byte, case, catch, char, class,
- continue, default, do, double,if, else,enum,extends,
- final, finally, float, for, implements, import,
- instanceof, int, interface, long, native, new, null,
- package, private, protected,

Java Control Statements in Java

Decision Making statements

- if statements
- switch statement

Looping statements

- Do-while loop
- while loop
- for loop
- for-each loop

- **Jump statements**
- break statement
- continue statement

Java If-else Statement

- The Java *if statement* is used to test the condition. It checks boolean condition: *true* or *false*. There are various types of if statement in Java.
  - **if statement**
  - **if-else statement**
  - **if-else-if ladder**
  - **nested if statement**

Java Switch Statement

- The Java *switch statement* executes one statement from multiple conditions.
- It is like if-else-if ladder statement. T
- he switch statement works with byte, short, int, long, enum types, String and some wrapper types like Byte, Short, Int, and Long. Since Java 7, you can use strings in the switch statement.
- In other words, the switch statement tests the equality of a variable against multiple values.

For loop

The Java *for loop* is used to iterate a part of the program several times. If the number of iteration is **fixed**, it is recommended to use for loop. There are three types of for loops in Java.

- Simple for Loop
- For-each or Enhanced for Loop
- Labeled for Loop

Java Simple for Loop

A simple for loop is the same as C/C++. We can initialize the variable, check condition and increment/decrement value.

Syntax:

```
for(initialization; condition; increment/decrement)
{
//statement or code to be executed
}
```

Java Nested for Loop

- If we have a for loop inside the another loop, it is known as nested for loop. The inner loop executes completely whenever outer loop executes.

Java for-each Loop

The for-each loop is used to traverse array or collection in Java. It is easier to use than simple for loop because we don't need to increment value and use subscript notation. It works on the basis of elements and not the index. It returns element one by one in the defined variable. **Syntax:**

```
for(data_type variable : array_name)
{
//code to be executed
}
```

Java Labeled For Loop

We can have a name of each Java for loop. To do so, we use label before the for loop. It is useful while using the nested for loop as we can break/continue specific for loop.

Syntax:

```
labelname: for(initialization; condition; increment/decrement)
{
//code to be executed
}
```

Java for Loop vs while Loop vs do-while Loop

Comparison	for loop	while loop	do-while loop
Introduction	The Java for loop is a control flow statement that iterates a part of the programs multiple times.	The Java while loop is a control flow statement that executes a part of the programs repeatedly on the basis of given boolean condition.	The Java do while loop is a control flow statement that executes a part of the programs at least once and the further execution depends upon the given boolean condition.
When to use	If the number of iteration is fixed, it is recommended to use for loop.	If the number of iteration is not fixed, it is recommended to use while loop.	If the number of iteration is not fixed and you must have to execute the loop at least once, it is recommended to use the do-while loop.

Syntax	for(init;condition;incr/decr){
	//code to be executed
	}
		while(condition){
		//code to be executed
		}
			do{
			//code to be executed
			}while(condition);

Example	//for loop
	for(int i=1;i<=10;i++){
	System.out.println(i);
	}
		//while loop
		int i=1;
		while(i<=10){
		System.out.println(i);
		i++;
		}
			//do-while loop
			int i=1;
			do{
			System.out.println(i);
			i++;
			}while(i<=10);

Syntax for infinitive loop	for(;;){
	//code to be executed
	}
		while(true){
		//code to be executed
		}
			do{
			//code to be executed
			}while(true);

Java While Loop

The Java *while loop* is used to iterate a part of the program repeatedly until the specified Boolean condition is true. As soon as the Boolean condition becomes false, the loop automatically stops. The while loop is considered as a repeating if statement. If the number of iteration is not fixed, it is recommended to use the while loop.

Syntax:

```
while (condition)
{
//code to be executed

Increment / decrement statement
}
```

Java do-while Loop

The Java *do-while loop* is used to iterate a part of the program repeatedly, until the specified condition is true. If the number of iteration is not fixed and you must have to execute the loop at least once, it is recommended to use a do-while loop. Java do-while loop is called an **exit control loop**. Therefore, unlike while loop and for loop, the do-while check the condition at the end of loop body. The Java *do-while loop* is executed at least once because condition is checked after loop body.

Syntax:

```
Do
{
//code to be executed / loop body

//update statement
}while (condition);
```

Java Continue Statement

The continue statement is used in loop control structure when you need to jump to the next iteration of the loop immediately. It can be used with for loop or while loop. The Java *continue statement* is used to continue the loop. It continues the current flow of the program and skips the remaining code at the specified condition. In case of an inner loop, it continues the inner loop only. We can use Java continue statement in all types of loops such as for loop, while loop and do-while loop.

Syntax:

jump-statement;

continue;

Java Comments

The Java comments are the statements in a program that are not executed by the compiler and interpreter.

Types of Java Comments

Single Line Comment

Multi Line Comment

Documentation Comment

Why do we use comments in a code?

- Comments are used to make the program more readable by adding the details of the code.
- It makes easy to maintain the code and to find the errors easily.
- The comments can be used to provide information or explanation about the variable, method, class, or any statement.
- It can also be used to prevent the execution of program code while testing the alternative code.

Java Single Line Comment

- The single-line comment is used to comment only one line of the code. It is the widely used and easiest way of commenting the statements. Single line comments starts with two forward slashes **(//)**. Any text in front of // is not executed by Java.
- **Syntax:**
- //This is single line comment

Java Multi Line Comment

The multi-line comment is used to comment multiple lines of code. It can be used to explain a complex code snippet or to comment multiple lines of code at a time (as it will be difficult to use single-line comments there). Multi-line comments are placed between /* and */. Any text between /* and */ is not executed by Java.

Syntax:

```
/*
This
is
multi line
comment
*/
```

Java Documentation Comment

Documentation comments are usually used to write large programs for a project or software application as it helps to create documentation API. These APIs are needed for reference, i.e., which classes, methods, arguments, etc., are used in the code. To create documentation API, we need to use the **javadoc tool**. The documentation comments are placed between /** and */.

Syntax:

```
/**
**We can use various tags to depict the parameter
*or heading or author name
*We can also use HTML tags
*
*/
```