## [🏠\_Home\_](https://themiu.github.io/Notes/)

##### 🔗 [300 Core Java Interview Questions (2023) - javatpoint](https://www.javatpoint.com/corejava-interview-questions)

##### 🔗 [Java Interview Codes](https://docs.google.com/document/d/1YH514z7g347_4qQIBAUg-wmJb0E1VL0eFBBJSiB_iS0)

## How to execute a computer program?

To execute a computer program, you typically need to write or obtain the program code, install the necessary software, compile (if required), and then run the program.

## What is a computer program?

Specific task ekak karaganna Machine ekata deepu Instruction set ekak.

## Why do we need a programming language?

Machine ekath ekka communicate karanna

## Why are so many programming languages available in the world?

Different requirements fulfill karanna

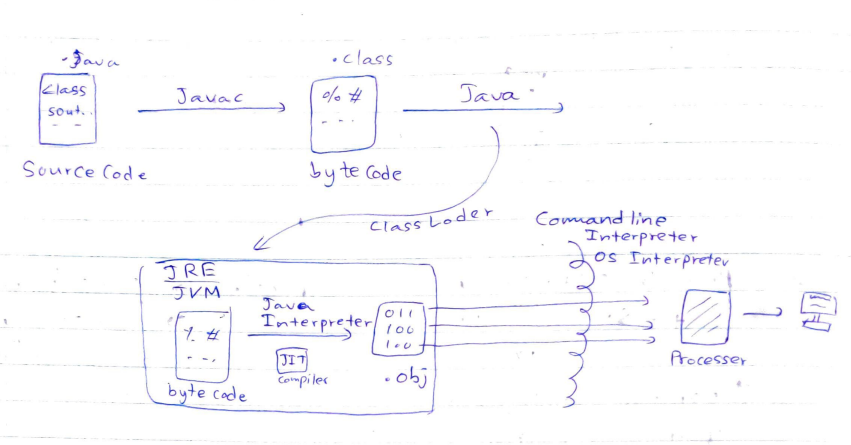
## What are the key benefits of choosing Java rather than other programming languages?

* Platform independence
* Object-oriented programming (OOP)
* Strong memory management
* Large ecosystem and community support

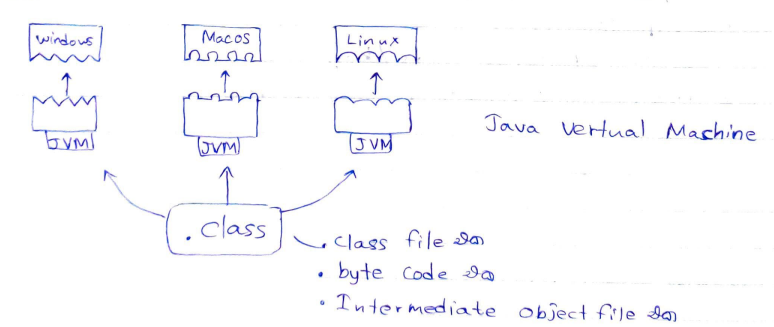
## Who is the founder of Java?

James Gosling at Sun Microsystems

## Draw a diagram and explain how the Java program is running.



## Why is the Java program platform independent? (WORA behavior - Write Once Run Anywhere)

because they are compiled into **bytecode** that can be executed by the Java Virtual Machine (JVM). The JVM provides a consistent runtime environment across different operating systems, allowing Java programs to run without modification on any system that has a compatible JVM.  


## Why does the JVM become platform dependent?

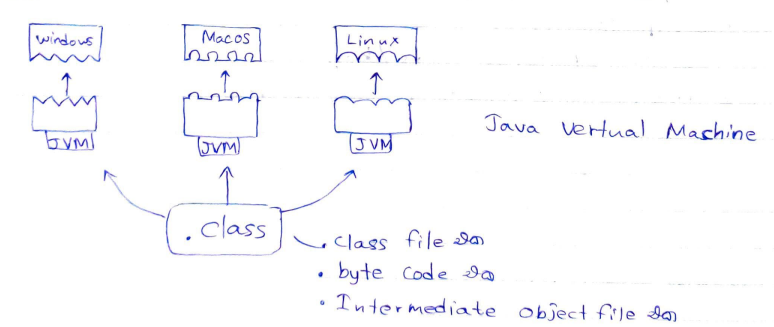
Java software eka OS matha depend weno

## Explain JDK, JRE, JVM

## What is the purpose of the Java compiler?

## What is a Byte/Class/Intermediate code file and why is it so essential in Java?

Bytecode eka nisa Java language eka platform independent



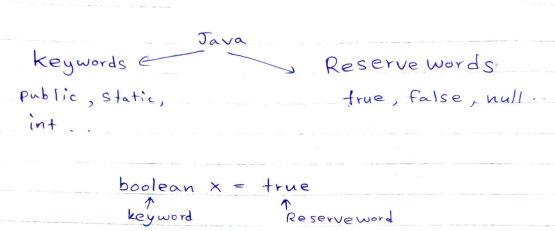
## Explain Java Interpreter Vs. O/s Interpreter/Command line Interpreter

* **Java Interpreter**
  + Executes Java bytecode
  + Runs Java programs on the Java Virtual Machine (JVM)
  + ex: Java Virtual Machine (JVM)
* **Operating System Interpreter/Command Line Interpreter**
  + Interprets and executes OS-specific commands/scripts
  + Executes commands and scripts for OS tasks
  + ex: Command Prompt, Terminal, Bash, PowerShell, etc.

## What is the purpose of the JIT compiler?

Ekama line eka nawta nawata interpret weema walakwai.

## Explain Java keywords and Reserved words.

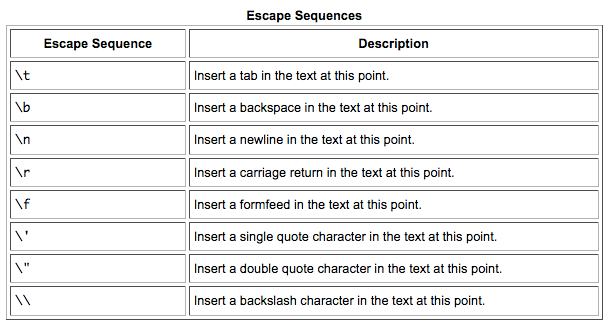


## Explain the Java Literals.

Sout එකක් ඇතුලෙ දාන්න පුලුවන් ඕනම එකක්.

1. Integer Literals
2. Floating point Literals
3. String Literals
4. Character Literals
5. Boolean Literals

## What are special characters?

\t, \n, \b 

## What is a computer variable?

* Data එකක්,
* temporary store කරල තියාගන්න,
* RAM එකෙන් වෙන් කරගත්තු,
* memory location එකක්

## Explain how data are represented in computer memory using One’s complement and Two’s complement.

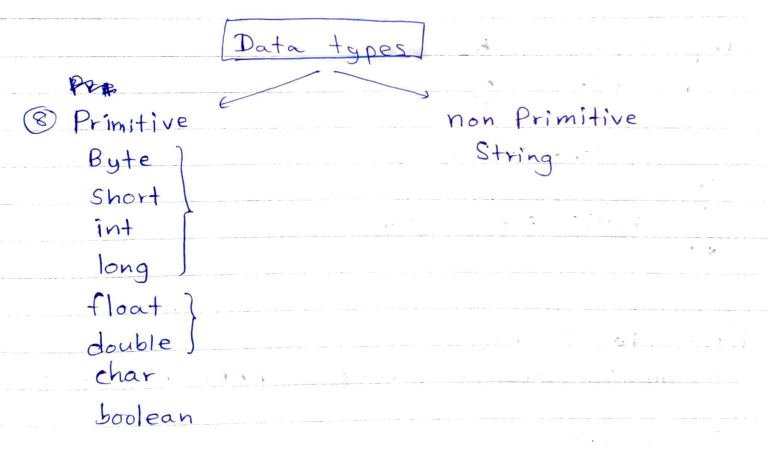
1's complement of 7 (0111) is 8 (1000)

(0 wenuwata 1 , 1 wenuwata 0)

2's complement of 7 (0111) is 9 (1001)

(1’s complement walata 1k ekathu karala)

## How many primitive types are available and what are they?



## Is the String data type a primitive or non-primitive?

Non-primitive

## Explain Scanner and Random.

Scanner - for reading inputs

Random - for generating random numbers and values

## What are the flow controls that you can use in Java?

if-else: Executes code conditionally based on a specified condition.

switch: Allows choosing between multiple alternatives based on the value of an expression.

for: Repeats a block of code a fixed number of times.

while: Repeats a block of code as long as a specified condition is true.

do-while: Repeats a block of code at least once, and then continues as long as a specified condition is true.

break: Terminates the loop or switch statement and transfers control to the next statement after the loop or switch.

continue: Skips the current iteration of a loop and proceeds to the next iteration.

return: Exits a method and returns a value (if specified) to the calling code.

**<Practical>**

## Create a program to perform a grading system that can input average marks from the keyboard and give respective grades for it using if, else, else-if.

## Perform the above program using ternary operators.

## What are Operators in Java?

Operators in Java are special symbols or characters used to perform operations on operands.

They include

* arithmetic operators (+, -, \*, /, %),
* assignment operators (=, +=, -=, \*=, /=, %=),
* comparison operators (==, !=, >, <, >=, <=),
* logical operators (&&, ||, !),
* bitwise operators (&, |, ^, ~, <<, >>, >>>),
* increment/decrement operators (++, --),
* conditional operator (ternary operator ?:),
* instanceof operator,
* and string concatenation operator (+).

## How do postfix Vs. prefix increment and decrement operators perform?

Postfix increment/decrement operators (i++, i--)

1. Use current value
2. Then increment/decrement the variable

Prefix increment/decrement operators (++i, --i)

1. increment/decrement the variable
2. Then use its updated value

## In what kind of scenario you can use the Java switch and create a program to demonstrate it?

when you have multiple cases to evaluate against a single variable or expression. It allows you to efficiently select and execute different blocks of code based on the matching case.

## What does iteration mean?

waara keepayak Actions set ekak repeat karano. (loop ekaka i)

## How do you execute a program if you have an Iteration process in your program?

## Explain the difference between for-loop Vs. while-loop

for: Repeats a block of code a fixed number of times.

while: Repeats a block of code as long as a specified condition is true.

## Explain the difference between while-loop Vs. do/while-loop.

while: Repeats a block of code as long as a specified condition is true.

do-while: Repeats a block of code at least once, and then continues as long as a specified condition is true.

## Explain break, continue, and return keywords in Java.

break: Terminates the loop or switch statement and transfers control to the next statement after the loop or switch.

continue: Skips the current iteration of a loop and proceeds to the next iteration.

return: Exits a method and returns a value (if specified) to the calling code.

## For which purpose do we need to use Java Labels?

used to control the flow of execution in nested loops or switch statements. They allow you to jump to a specific statement, break out of nested loops, or continue to the next iteration.

| public class LabelExample {  public static void main(String[] args) {  outerLoop: for (int i = 1; i <= 3; i++) {  innerLoop: for (int j = 1; j <= 3; j++) {  System.out.println("i = " + i + ", j = " + j);  if (i == 2 && j == 2) {  break outerLoop; *// Breaks out of both loops when i=2 and j=2*  }  }  }  } } |
| --- |

## What is the starting point of the Java program?

Main method

## Will the Java program still run after you remove the main method from your program? Will it be a compile-time error, a Runtime error, or nothing happens?

No. a compile-time error

## What is the method/function and explain the advantages of using it.

block of code that performs a specific task or set of actions

## How many method types are in Java and demonstrate the example program for each and every type.

General , parameterized, return type

| public class GeneralMethodExample {  public static void sayHello() {  System.out.println("Hello, World!");  }   public static void main(String[] args) {  sayHello(); *// Call the general method*  } } |
| --- |

| public class ParameterizedMethodExample {  public static void greet(String name) {  System.out.println("Hello, " + name + "!");  }   public static void main(String[] args) {  String personName = "John";  greet(personName); *// Call the parameterized method*  } } |
| --- |

| public class ReturnTypeMethodExample {  public static int addNumbers(int a, int b) {  return a + b;  }   public static void main(String[] args) {  int num1 = 5;  int num2 = 3;  int sum = addNumbers(num1, num2); *// Call the return type method and store the result*  System.out.println("Sum: " + sum);  } } |
| --- |

## What is the return type of the main method?

void

## What does the void type mean in a method?

that the method does not return any value.

## Explain the arguments and method parameters.

| public class Example {  public static void main(String[] args) {  int num1 = 5;  int num2 = 3;  int result = addNumbers(num1, num2); *// num1 and num2 are the arguments*  System.out.println("Sum: " + result);  }   public static int addNumbers(int a, int b) { *// a and b are the parameters*  return a + b;  } } |
| --- |

Arguments - pass karana actual values

Parameters - arguments daganna declare karapu variables

## What is method recursion?

Method recursion is a technique where a method calls itself.

| public class RecursionExample {  public static void countDown(int n) {  if (n <= 0) {  System.out.println("Done!");  } else {  System.out.println(n);  countDown(n - 1); *// Recursive call with a smaller value*  }  }   public static void main(String[] args) {  countDown(5);  } } |
| --- |

## What will happen if you declare 2 methods with the same method name with the same parameter types and count inside your class?

compilation error.

## So if you wanna declare 2 methods with the same name, what would you do then?

Change parameters (count or type)

## What does Static polymorphism/ Compile time polymorphism/ Method overloading means?

\* එකම class එකක් ඇතුලෙ

\* same name - different parameter count

\* same name - different parameter type

\* method හදන්න පුලුවන්.

## Can we overload our main method? If so, how?

Yes

| public class MainMethodOverloading {  public static void main(String[] args) {  System.out.println("This is the default main method");  main(10); *// Calling the overloaded main method with int num*  main("Hello", "World"); *// Calling the overloaded main method with two String arguments*  }   public static void main(int num) {  System.out.println("This is the main method with int num");  }   public static void main(String arg1, String arg2) {  System.out.println("This is the main method with two String arguments");  } } |
| --- |

## Does giving any effect to the method overloading depend on the method’s return type? Explain your answer.

No, the method's return type does not have any effect on method overloading. Method overloading is determined solely by the method name and its parameter types or counts.

## In which situation can we put the return statement in Java?

* Return a value from a method.

| public int calculateSum(int a, int b) {  int sum = a + b;  return sum; *// Returning the sum as the result of the method* } |
| --- |

* Exit the method prematurely based on conditions.

| public boolean isPositive(int number) {  if (number > 0) {  return true; *// Return true if the number is positive*  } else {  return false; *// Return false for non-positive numbers*  } } |
| --- |

* Control the flow of a method.

| public void printNumbers(int limit) {  for (int i = 1; i <= limit; i++) {  if (i == 5) {  return; *// Exit the method when i equals 5*  }  System.out.println(i);  } } |
| --- |

* Handle exceptional cases.

| public int divide(int dividend, int divisor) {  if (divisor == 0) {  System.out.println("Error: Division by zero is not allowed.");  return -1; *// Return a specific value to indicate an error*  }  return dividend / divisor; *// Return the result of the division* } |
| --- |

* In void methods, it can be used to simply exit the method.

| public void greet(String name) {  if (name == null) {  System.out.println("Please provide a name.");  return; *// Exit the method without performing further actions*  }  System.out.println("Hello, " + name + "!"); } |
| --- |

## Can we put statements under the return statement? Explain your answer.

no.

## If we create a variable inside the methodA() and then try to access that variable inside the main method at the end of that methodA() execution process, is this legal or not?

## Explain your answer by drawing with the method stack calling diagram.

It is illegal to access a local variable from a different scope.

| public class Main {  public static void main(String[] args) {  methodA();  *// Attempting to access the variable declared in methodA() here*  }   public static void methodA() {  int x = 10; *// Local variable declared inside methodA()*  *// Some code...*   *// MethodA execution finishes here*  } } |
| --- |